Sistema Administrador de Competencias de Programación Competitiva “ICPC-UMSS”

Manual técnico

Consultor TIS: Lic. Maria Leticia Blanco Coca
Convocatoria Pública: PETIS-1408-2023
Razón Social: INFINITY CODE Ltda.

INFINITY
C O D E

Contenido

1. Introducción

Este manual técnico tiene como objetivo detallar los recursos utilizados en la creación del Sistema administrador de competencias de programación competitiva, denominado “ICPC-UMSS”. La implementación de este proyecto se fundamenta en el uso de Laravel, un framework de código abierto diseñado para el desarrollo eficiente de aplicaciones y servicios web utilizando el lenguaje de programación PHP. A lo largo de este documento, se proporcionará información detallada sobre la arquitectura, funcionalidades y procesos clave que conforman el ICPC-UMSS, permitiendo una comprensión exhaustiva del sistema y facilitando su mantenimiento y evolución.

2. Objetivo de este manual

El objetivo principal de este Manual Técnico es proporcionar una guía exhaustiva y accesible para todos los aspectos del diseño, desarrollo, implementación y mantenimiento de la página web “ICPC-UMSS”.
Este documento está diseñado para servir como un recurso central que capacita a desarrolladores, ingenieros y personal técnico para entender la arquitectura del sistema, implementar mejores prácticas de desarrollo, y abordar eficazmente los desafíos inherentes al ciclo de vida del proyecto web.

3. Dirigido

Este manual está pensado para ser utilizado por profesionales de distintas áreas, como desarrolladores, ingenieros, personal técnico y administradores. Proporciona información técnica esencial sobre la estructura del sistema y prácticas recomendadas para el mantenimiento eficiente. Su enfoque general y adaptable lo hace relevante para una variedad de roles dentro del contexto del proyecto "ICPC-UMSS".

4. Conocimientos previos

Antes de comenzar a operar las páginas y utilizar este manual, es importante contar con ciertos conocimientos mínimos. Las personas que interactúan con la información proporcionada en este documento deben tener:

- Conocimientos básicos acerca de Programas Utilitarios.
- Conocimiento básico de Internet.
- Conocimiento básico de Windows.

5. Especificaciones de requerimientos de software

Con el fin de garantizar el éxito en la implementación del Sistema de Gestión de Eventos "ICPC-UMSS", diseñado para la organización y registro eficiente de eventos, se detallan los

requisitos de software esenciales que constituirán la base sólida para asegurar una implementación efectiva y un rendimiento óptimo del sistema.

Sistema Operativo:

Windows 7 o superior.
GNU/Linux (Ubuntu 14.04 o superiores).

Base de Datos:

MariaDB 10.6.3

Servidor Web:

Apache v.2+.

Entorno de Desarrollo:

Laravel 8.75

Lenguajes de Programación:

PHP 5.6.
Javascript.

Manejador de Dependencias:

Composer 2.6.3

Librerías:

Bootstrap
Axios
FontAwesome

Navegadores Web Compatibles:

Google Chrome.
Microsoft Edge.
Mozilla Firefox.
Opera.

Editor de Texto:

Visual Studio Code

Repositorio del Proyecto:

Github: https://github.com/Jorge2610/tis-icpc.git

Características Clave:

Gestión de Competencias: Permite la creación, programación y gestión de competencias de programación, incluyendo detalles como fechas, horarios, y ubicaciones.

Registro de Equipos y Participantes: Facilita el registro de equipos y participantes para cada competencia.

Menú de información:

- Eventos

Menú lateral:

- TIPO DE EVENTO
  - Ver tipos de evento
  - Crear tipo de evento
  - Editar tipo de evento
  - Eliminar tipo de evento
- EVENTO
  - Crear evento
  - Editar evento
  - Cancelar evento
  - Anular evento
  - Notificar evento
- ACTIVIDAD
  - Crear actividad
  - Editar actividad
  - Eliminar actividad
- AFICHE
  - Asignar afiche
  - Cambiar afiche
  - Eliminar afiche
- PATROCINADOR
  - Crear patrocinador
  - Editar patrocinador
  - Eliminar patrocinador
  - Asignar patrocinador
  - Quitar patrocinador
- SITIOS DE INTERÉS
  - Registrar sitio de interés
  - Quitar sitio de interés

6. Documentación de diseño

6.1. Diseño de base de datos:

6.1.1. Diagrama de Entidad-Relación (DER)

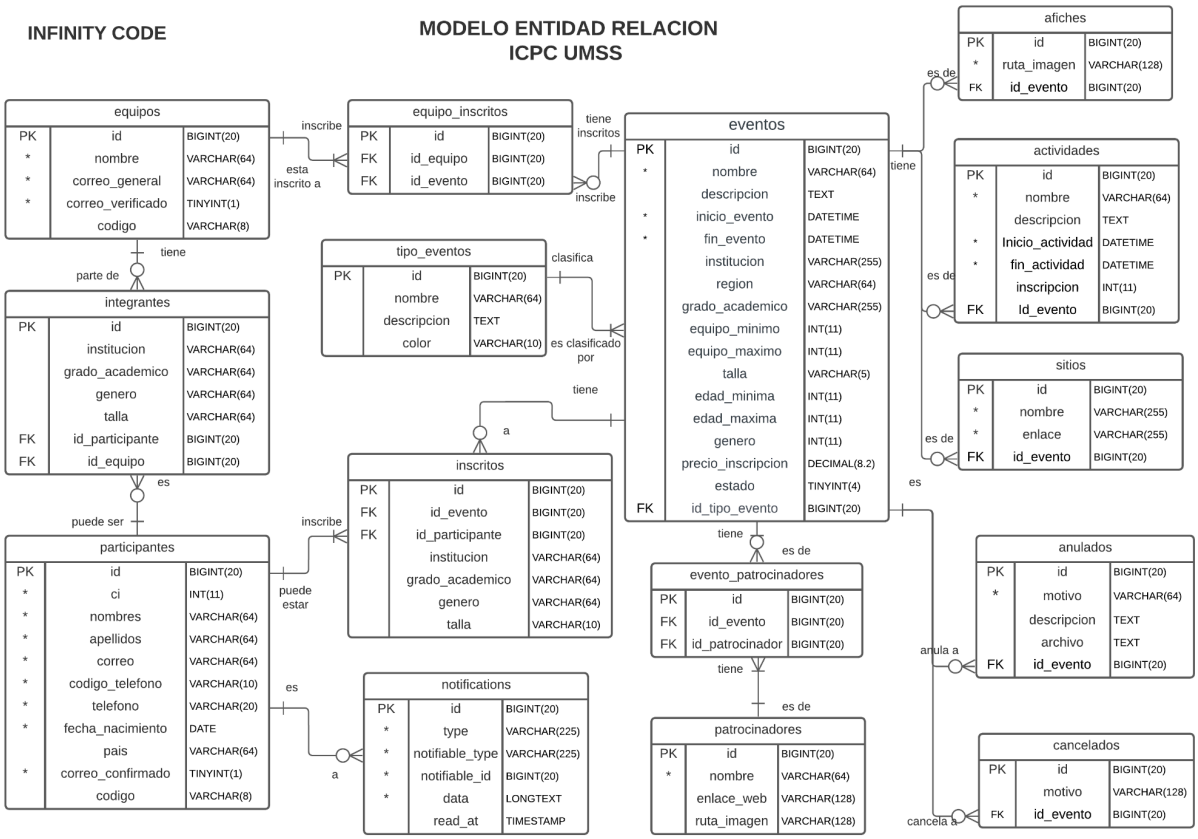


Figura 1. Diagrama de Entidad-Relación de ICPC-UMSS

6.1.2. Análisis de datos para la migración en base de datos

Para este objetivo se diseñó un modelo de E-R (Entidad Relación), que nos permitió hacer el análisis de datos correspondientes a las migraciones que se fueron creando.

6.1.3. Migraciones creadas

Las migraciones creadas son:
2023_11_16_144902_create_actividads_table.php .- Crea una tabla llamada actividades con atributos específicos para almacenar información relacionada con actividades, como nombre, fechas de inicio y fin, descripción, número de

inscripciones, y una clave foránea que conecta la actividad con un evento asociado en la tabla eventos.

2023_11_14_194931_create_evento_patrocinadors_table.php .- Está diseñada para crear una tabla de relación muchos a muchos entre eventos y patrocinadores. La tabla resultante permite asociar múltiples patrocinadores a un evento y viceversa, lo que es útil en situaciones en las que un evento puede tener varios patrocinadores y un patrocinador puede estar asociado a varios eventos.

2023_11_06_002625_create_anulados_table.php .- Almacena información sobre eventos anulados, proporcionando detalles sobre el motivo de la anulación, una descripción opcional y la posibilidad de adjuntar archivos relacionados. La clave foránea id_evento vincula esta información específica de anulación con un evento en particular de la tabla eventos.

2023_11_06_002421_create_cancelados_table.php .- Almacena información sobre eventos cancelados, proporcionando detalles sobre el motivo de la cancelación. La clave foránea id_evento vincula esta información específica de cancelación con un evento en particular de la tabla eventos. El motivo de la cancelación es un campo opcional que permite registrar detalles adicionales sobre la razón de la cancelación, y la tabla mantiene un registro de las fechas y horas de creación y actualización de cada entrada.

2023_11_03_050436_create_sitios_table.php .- Almacena información sobre sitios o enlaces relacionados con eventos. La clave foránea id_evento vincula esta información específica del sitio con un evento en particular de la tabla eventos. La tabla registra detalles como el título, la URL y las fechas y horas de creación y actualización de cada entrada.

2023_10_25_162843_create_afiches_table.php .- Almacena información sobre afiches asociados a eventos. La clave foránea id_evento vincula esta información específica del afiche con un evento en particular de la tabla eventos. La tabla registra detalles como la ruta o nombre de archivo de la imagen y las fechas y horas de creación y actualización de cada entrada.

2023_10_03_224904_create_patrocinadores_table.php .- Almacena información sobre patrocinadores de eventos. La clave única en el campo nombre asegura que no haya patrocinadores duplicados, y se proporciona la opción de almacenar la imagen y el enlace web asociados con cada patrocinador. El borrado suave (softDeletes) permite la posibilidad de marcar registros como eliminados en lugar de borrarlos físicamente. La tabla también registra las fechas y horas de creación y actualización de cada entrada.

2023_10_03_222128_create_eventos_table.php .- Almacena información detallada sobre eventos, incluyendo nombre, descripción, fechas, requisitos de participación, precios y más. La clave foránea id_tipo_evento vincula cada evento

con un tipo específico de evento, y el campo estado puede utilizarse para indicar el estado actual del evento.

2023_10_03_221248_create_tipo_eventos_table.php .- Almacena información sobre tipos de eventos. La clave única en el campo nombre asegura que no haya tipos de eventos duplicados, y se proporciona la opción de almacenar una descripción y un color asociado con cada tipo de evento. El borrado suave (softDeletes) permite la posibilidad de marcar registros como eliminados en lugar de borrarlos físicamente. La tabla también registra las fechas y horas de creación y actualización de cada entrada.

2019_12_14_000001_create_personal_access_tokens_table.php .- Almacena información sobre tokens de acceso personal utilizados en la autenticación de la API. La relación polimórfica tokenable permite asociar el token con diferentes modelos en la aplicación. El campo abilities puede contener información sobre los permisos asociados al token, y last_used_at registra la última vez que se utilizó el token. La tabla también registra las fechas y horas de creación y actualización de cada entrada.

2019_08_19_000000_create_failed_jobs_table.php .- Almacena información sobre trabajos que fallaron durante la ejecución de la cola de trabajos en Laravel. La tabla registra detalles como la conexión de la cola, el nombre de la cola, los datos del trabajo y la excepción que causó el fallo, junto con la fecha y hora del fallo.

2014_10_12_100000_create_password_resets_table.php .- Almacena información sobre restablecimientos de contraseñas. La tabla registra la dirección de correo electrónico del usuario que solicitó el restablecimiento, el token asociado y la fecha y hora de creación del registro.

2014_10_12_000000_create_users_table.php .- Almacena información sobre usuarios. La tabla users registra detalles como el nombre, la dirección de correo electrónico, la verificación de correo electrónico, la contraseña cifrada y los detalles de marca de tiempo de creación y actualización de cada usuario.

2023_11_30_194151_create_inscritos_table.php .- Almacena información sobre los participantes inscritos en eventos, incluyendo detalles como la institución, el grado académico, el género y la talla de cada participante.

2023_12_08_105038_create_participantes_table.php .- Almacena información sobre los participantes, incluyendo detalles como la cédula de identidad, nombres, apellidos, dirección de correo electrónico, número de teléfono, fecha de nacimiento, país de residencia, estado de confirmación del correo electrónico y un código asociado al participante.

2023_12_08_120502_create_notifications_table.php .- Almacena notificaciones con información sobre el tipo de notificación, el modelo asociado (polimorfismo), los datos específicos de la notificación y el estado de lectura.
2023_12_08_194250_create_equipos_table.php .- Almacena información sobre los equipos participantes en eventos, incluyendo detalles como el nombre del equipo, la dirección de correo electrónico general, el estado de verificación del correo electrónico y un código asociado al equipo.
2023_12_08_194529_create_integrantes_table.php .- Almacena información sobre los integrantes de un equipo en eventos, incluyendo detalles como la institución, grado académico, género y talla de cada integrante.
2023_12_08_213708_create_equipo_inscrito.php .- Registra la inscripción de equipos en eventos, estableciendo relaciones entre la tabla equipos y la tabla eventos.
2023_12_15_160042_create_jobs_table.php .- Proporciona la estructura necesaria para que Laravel maneje trabajos en segundo plano y los gestione mediante colas de trabajos.

Las migraciones están en la carpeta de database, migrations

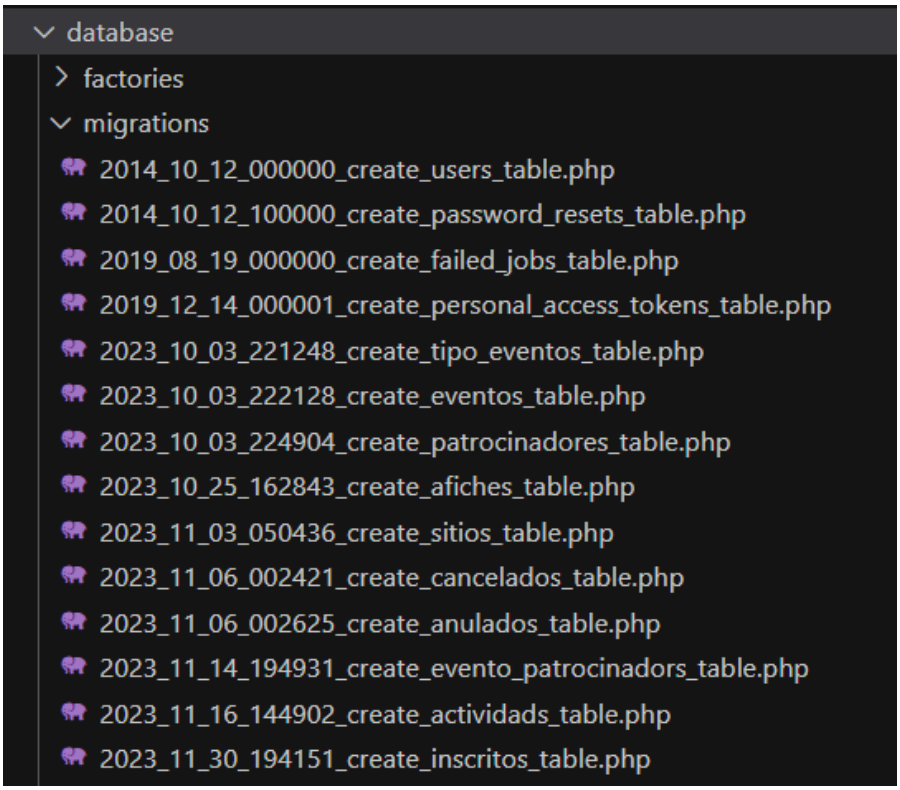


Figura 2. Migraciones del proyecto ICPC-UMSS

6.1.4. Modelos creados

Se creo los siguientes modelos:
Actividad.php
Afiche.php
Anulado.php
Cancelado.php
Equipo.php
EquipoInscrito.php
Evento.php
EventoPatrocinador.php
Inscrito.php
Integrante.php
Participante.php
Patrocinador.php
Sitio.php
TipoEvento.php
User.php

Estos modelos están ubicados en la carpeta app

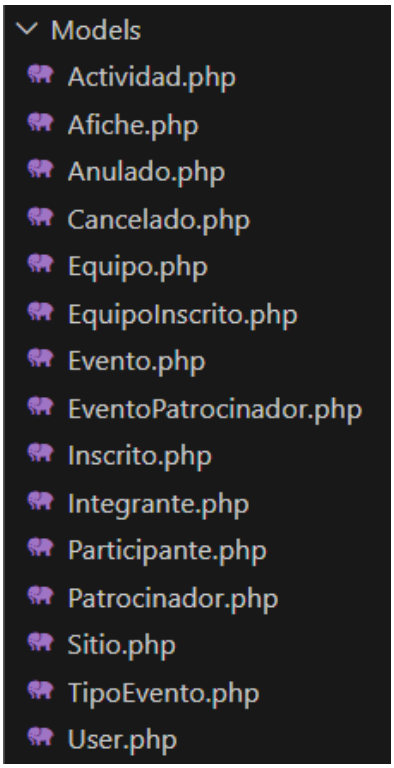


Figura 3. Modelos del proyecto ICPC-UMSS

6.1.5. Campos y tipo de datos de las migraciones

Se puede apreciar los capos,tipos de dato y el nombre de las tablas.

2023_11_16_144902_create_actividads_table.php Figura 4

```php
Schema::create('actividades', function (Blueprint $table) {
    $table->id();
    $table->string('nombre', 64);
    $table->dateTime('inicio_actividad')->format('d-m-Y');
    $table->dateTime('fin_actividad')->format('d-m-Y');
    $table->text('descripcion', 1000)->nullable();
    $table->integer('inscripcion')->nullable();
    $table->foreignId('id_evento')->constrained('eventos')->restrictOnDelete()->cascadeOnUpdate();
    $table->timestamps();
});
```

Figura 4.

2023_11_14_194931_create_evento_patrocinadors_table.php Figura 5

```php
Schema::create('evento_patrocinadores', function (Blueprint $table) {
    $table->id();
    $table->foreignId('id_evento')->constrained('eventos')->restrictOnDelete()->cascadeOnUpdate();
    $table->foreignId('id_patrocinador')->constrained('patrocinadores')->restrictOnDelete()->cascadeOnUpdate();
    $table->timestamps();
});
```

Figura 5.

2023_11_06_002625_create_anulados_table.php Figura 6

```php
Schema::create('anulados', function (Blueprint $table) {
    $table->id();
    $table->string("motivo", 64);
    $table->text("descripcion", 2048)->nullable();
    $table->text("archivos")->nullable();
    $table->foreignId('id_evento')->constrained('eventos')->restrictOnDelete()->cascadeOnUpdate();
    $table->timestamps();
});
```

Figura 6.

2023_11_06_002421_create_cancelados_table.php Figura 7

```php
Schema::create('cancelados', function (Blueprint $table) {
    $table->id();
    $table->string("motivo", 128)->nullable();
    $table->foreignId('id_evento')->constrained('eventos')->restrictOnDelete()->cascadeOnUpdate();
    $table->timestamps();
});
```

Figura 7.

2023_11_03_050436_create_sitios_table.php Figura 8

```php
Schema::create('sitios', function (Blueprint $table) {
    $table->id();
    $table->string('titulo');
    $table->string('enlace');
    $table->foreignId('id_evento')->constrained('eventos')->restrictOnDelete()->cascadeOnUpdate();
    $table->timestamps();
});
```

Figura 8.

2023_10_25_162843_create_afiches_table.php Figura 9

```php
Schema::create('afiches', function (Blueprint $table) {
    $table->id();
    $table->string('ruta_imagen', 128);
    $table->foreignId('id_evento')->constrained('eventos')->restrictOnDelete()->cascadeOnUpdate();
    $table->timestamps();
});
```

Figura 9.

2023_10_03_224904_create_patrocinadores_table.php Figura 10

```php
Schema::create('patrocinadores', function (Blueprint $table) {
    $table->id();
    $table->string('nombre', 64)->unique();
    $table->string('ruta_imagen', 128)->nullable();
    $table->string('enlace_web', 128)->nullable();
    $table->softDeletes();
    $table->timestamps();
});
```

Figura 10.

2023_10_03_222128_create_eventos_table.php Figura 11

```php
Schema::enableForeignKeyConstraints();
Schema::create('eventos', function (Blueprint $table) {
    $table->id();
    $table->string('nombre', 64)->unique();
    $table->text('descripcion', 2048)->nullable();
    $table->dateTime('inicio_evento')->format('d-m-Y');
    $table->dateTime('fin_evento')->format('d-m-Y');
    $table->string('institucion')->nullable();
    $table->string('region', 64)->nullable();
    $table->string('grado_academico')->nullable();
    $table->integer('equipo_minimo')->nullable();
    $table->integer('equipo_maximo')->nullable();
    $table->string('talla', 5)->nullable();
    $table->integer('edad_minima')->nullable();
    $table->integer('edad_maxima')->nullable();
    $table->string('genero', 10)->nullable();
    $table->decimal('precio_inscripcion')->nullable();
    $table->tinyInteger('estado')->nullable()->default(0);
    $table->foreignId('id_tipo_evento')->constrained('tipo_eventos')->restrictOnDelete()->cascadeOnUpdate();
    $table->timestamps();
});
```

Figura 11.

2023_10_03_221248_create_tipo_eventos_table.php Figura 12

```php
Schema::create('tipo_eventos', function (Blueprint $table) {
    $table->id();
    $table->string('nombre', 64)->unique();
    $table->text('descripcion', 500)->nullable();
    $table->string('color', 10)->default('#000000');
    $table->softDeletes();
    $table->timestamps();
});
```

Figura 12

2019_12_14_000001_create_personal_access_tokens_table.php Figura 13

```php
Schema::create('personal_access_tokens', function (Blueprint $table) {
    $table->id();
    $table->morphs('tokenable');
    $table->string('name');
    $table->string('token', 64)->unique();
    $table->text('abilities')->nullable();
    $table->timestamp('last_used_at')->nullable();
    $table->timestamps();
});
```

Figura 13.

2019_08_19_000000_create_failed_jobs_table.php Figura 14

```php
Schema::create('failed_jobs', function (Blueprint $table) {
    $table->id();
    $table->string('uuid')->unique();
    $table->text('connection');
    $table->text('queue');
    $table->longText('payload');
    $table->longText('exception');
    $table->timestamp('failed_at')->useCurrent();
});
```

Figura 14.

2014_10_12_100000_create_password_resets_table.php Figura 15

```php
Schema::create('password_resets', function (Blueprint $table) {
    $table->string('email')->index();
    $table->string('token');
    $table->timestamp('created_at')->nullable();
});
```

Figura 15.

2014_10_12_000000_create_users_table.php Figura 16

```php
Schema::create('users', function (Blueprint $table) {
    $table->id();
    $table->string('name');
    $table->string('email')->unique();
    $table->timestamp('email_verified_at')->nullable();
    $table->string('password');
    $table->rememberToken();
    $table->timestamps();
});
```

Figura 16.

2023_12_30_194151_create_inscritos_table.php Figura 17

```php
Schema::create('inscritos', function (Blueprint $table) {
    $table->id();
    $table->foreignId('id_evento')->constrained('eventos')->onDelete('cascade');
    $table->foreignId('id_participante')->constrained('participantes')->onDelete('cascade');
    $table->string('institucion', 64)->nullable();
    $table->string('grado_academico', 64)->nullable();
    $table->string('genero', 64)->nullable();
    $table->string('talla', 10)->nullable();
    $table->timestamps();
});
```

Figura 17.

2023_12_08_105038_create_participantes_table.php Figura 18

```php
Schema::create('participantes', function (Blueprint $table) {
    $table->id();
    $table->integer('ci');
    $table->string('nombres', 64);
    $table->string('apellidos', 64);
    $table->string('correo', 64);
    $table->string('codigo_telefono', 10);
    $table->string('telefono', 64);
    $table->date('fecha_nacimiento');
    $table->string('pais', 64)->nullable();
    $table->boolean('correo_confirmado')->default(0);
    $table->string('codigo',8)->nullable();
    $table->timestamps();
});
```

Figura 18.

2023_12_08_120502_create_notifications_table.php Figura 19

```php
Schema::create('notifications', function (Blueprint $table) {
    $table->uuid('id')->primary();
    $table->string('type');
    $table->morphs('notifiable');
    $table->json('data');
    $table->timestamp('read_at')->nullable();
    $table->timestamps();
});
```

Figura 19.

2023_12_08_194250_create_equipos_table.php Figura 20

```php
Schema::create('equipos', function (Blueprint $table) {
    $table->id();
    $table->string('nombre', 64)->nullable();
    $table->string('correo_general', 64);
    $table->boolean('correo_verificado')->default(0);
    $table->string('codigo')->nullable();
    $table->timestamps();
});
```

Figura 20.

2023_12_08_194529_create_integrantes_table.php Figura 21

```php
Schema::create('integrantes', function (Blueprint $table) {
    $table->id();
    $table->string('institucion', 64)->nullable();
    $table->string('grado_academico', 64)->nullable();
    $table->string('genero', 64)->nullable();
    $table->string('talla', 10)->nullable();
    $table->foreignId('id_participante')->constrained('participantes')->restrictOnDelete()->cascadeOnUpdate();
    $table->foreignId('id_equipo')->constrained('equipos')->restrictOnDelete()->cascadeOnUpdate();
    $table->timestamps();
});
```

Figura 21.

2023_12_08_213708_create_equipo_inscrito.php Figura 22

```php
Schema::create('equipo_inscritos', function (Blueprint $table) {
    $table->id();
    $table->foreignId('id_equipo')->constrained('equipos')->restrictOnDelete()->cascadeOnUpdate();
    $table->foreignId('id_evento')->constrained('eventos')->restrictOnDelete()->cascadeOnUpdate();
    $table->timestamps();
});
```

Figura 22.

2023_12_15_160042_create_jobs_table.php Figura 23

```php
Schema::create('jobs', function (Blueprint $table) {
    $table->bigIncrements('id');
    $table->string('queue')->index();
    $table->longText('payload');
    $table->unsignedTinyInteger('attempts');
    $table->unsignedInteger('reserved_at')->nullable();
    $table->unsignedInteger('available_at');
    $table->unsignedInteger('created_at');
});
```

Figura 23.

6.2. Diseño de interfaces de usuario

En el desarrollo de las interfaces de usuario, se implementan principios fundamentales de Bootstrap. Las directrices a seguir se detallan a continuación, enfocándose en prácticas específicas para garantizar una implementación eficiente y coherente en el sistema:

Nombrado Conciso de Elementos: Se prioriza la brevedad en la nomenclatura de etiquetas, asegurando una comunicación clara y eficiente. Los botones se etiquetaron con

verbos en modo infinitivo, describiendo de manera precisa la acción que el usuario llevará a cabo al interactuar con ellos.

Explotación de la Cuadrícula Flexible: Se aprovecha la versatilidad de la cuadrícula flexible proporcionada por Bootstrap para lograr un diseño consistente y adaptativo en todas las vistas del sistema. Esto garantiza la presentación ordenada de contenido y mejora la respuesta en diversos dispositivos.

Adaptación de Componentes: Aunque Bootstrap ofrece componentes predefinidos, se realizaron ajustes y personalizaciones para satisfacer las necesidades específicas del sistema. Esto asegura que el diseño general siga las pautas de Bootstrap y, al mismo tiempo, se ajuste con precisión al contexto del proyecto.

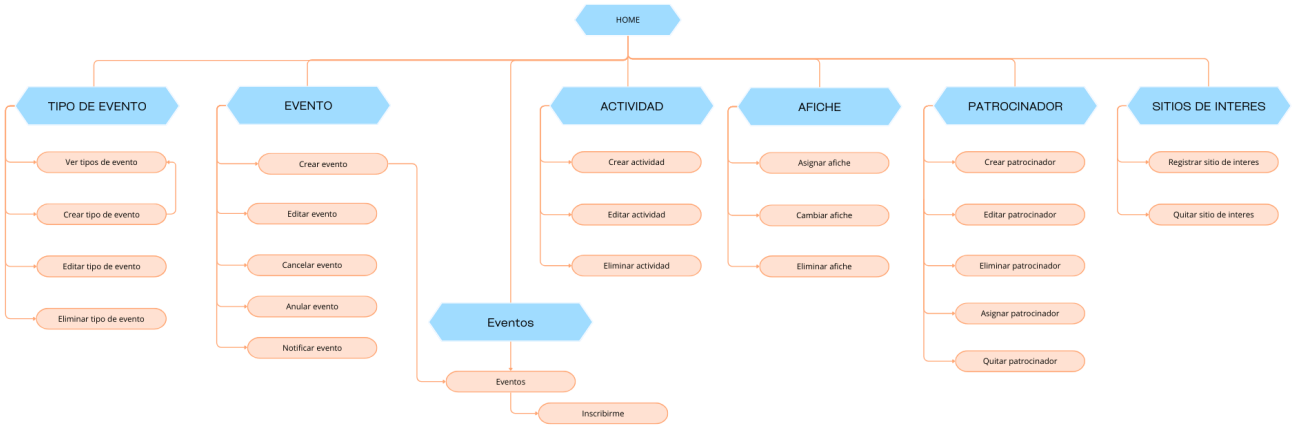


Figura 24. Diagrama de flujos de acceso entre las vistas (sitemap)

6.3. Arquitectura de sistema ICPC-UMSS

El Sistema Administrador de Competencias de Programación Competitiva (ICPC-UMSS) está diseñado siguiendo una arquitectura de tres capas, basada en el patrón Modelo-Vista-Controlador (MVC). Esta elección arquitectónica proporciona modularidad, flexibilidad y facilita el mantenimiento del sistema a medida que evoluciona.

6.3.1. Componentes Principales:

Capa de Presentación (Vista):

- Responsable de la interfaz de usuario y la interacción con el usuario.
- Desarrollada utilizando las plantillas Blade de Laravel y estilos de Bootstrap para una apariencia moderna y responsiva.

Capa de Lógica de Negocios (Controlador):

- Contiene la lógica de aplicación y la gestión de datos.
- Implementada en Laravel, un framework PHP que sigue el patrón MVC.

- Utiliza controladores para procesar las solicitudes del usuario y gestionar la lógica de negocio.

Capa de Datos (Modelo):

- Responsable de la interacción con la base de datos.
- Utiliza el ORM Eloquent de Laravel para simplificar las operaciones de base de datos.
- Define modelos que representan las entidades del sistema, como Competencias y Participantes.

6.3.2. Cuadro de estructura

La siguiente imagen explica la interacción de los componentes y el funcionamiento del Framework.

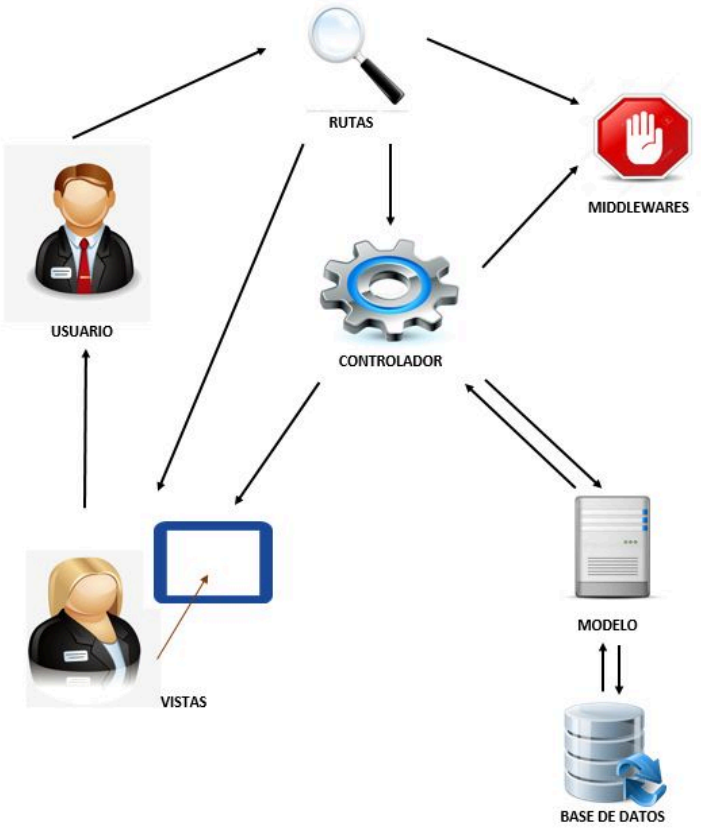


Figura 25. Estructura de Laravel.

El proyecto presenta la siguiente estructura de folders y archivos:

- app/
- bootstrap/
- config/
- database/
- public/

- resources/
- routes/
- storage/
- tests/
- vendor/
- .editorconfig
- .env
- .env.example
- .gitattributes
- .gitignore
- .styleci.yml
- artisan
- composer.json
- composer.lock
- LICENSE
- package-lock.json
- package.json
- phpunit.xml
- README.md
- server.php
- .webpack.mix.js

A continuación, se describen los directorios y archivos más importantes para que nos ayuden a entender más el funcionamiento del framework.

El directorio app

El directorio "app/" concentra la mayoría de la lógica de la aplicación. Aquí se encuentran los controladores, modelos, políticas, observadores y otros archivos cruciales que definen el comportamiento y la interacción de la aplicación. Cada subdirectorio desempeña un papel específico:

Console:

Almacena comandos personalizados de Artisan. Artisan es la interfaz de línea de comandos de Laravel, y este subdirectorio permite definir comandos personalizados para realizar tareas específicas, como la generación de código o la ejecución de tareas programadas.

Exceptions:

Contiene clases relacionadas con el manejo de excepciones en la aplicación. Aquí se pueden personalizar las respuestas ante errores específicos, proporcionando un control más preciso sobre cómo la aplicación responde a situaciones inesperadas.

Http:

Fundamental para el manejo de solicitudes y respuestas HTTP. El subdirectorio "Http" incluye controladores que gestionan las rutas de la aplicación y facilitan la interacción con el usuario a través de la web.

Mail:

Almacena clases relacionadas con el envío de correos electrónicos. Aquí se definen mailables que representan los diferentes tipos de correos electrónicos que la aplicación puede enviar, permitiendo una gestión eficiente de las comunicaciones por correo.

Models:

Es el lugar para almacenar modelos de datos. Cada modelo representa una tabla en la base de datos y facilita la interacción con dichos datos mediante consultas y manipulación.

Notifications:

Contiene clases relacionadas con el sistema de notificaciones de Laravel. Aquí se definen las notificaciones que pueden ser enviadas por correo electrónico, mensajes de texto, bases de datos, etc.

Providers:

Almacena clases que actúan como proveedores de servicios. Estos proveedores registran servicios en la aplicación, como enlaces, encargados de eventos y más, permitiendo una fácil extensibilidad y modularidad del sistema.

View:

Contiene archivos relacionados con las vistas de la aplicación. Las vistas definen la presentación y la interfaz de usuario, separando la lógica del controlador de la presentación visual.

El directorio config

El directorio "config/" en Laravel es esencial para la configuración de la aplicación. Aquí se encuentran archivos que definen una variedad de configuraciones para diferentes componentes del sistema. A continuación, se proporciona una descripción detallada de los contenidos de este directorio:

app.php:

Este archivo contiene configuraciones generales de la aplicación, como el nombre, el entorno de la aplicación y otros parámetros clave.

auth.php:

Define configuraciones relacionadas con la autenticación de usuarios, como los controladores de autenticación, las rutas de redirección y las vistas asociadas.

broadcasting.php:

Contiene configuraciones para la transmisión en tiempo real. Se puede definir aquí los controladores y conexiones utilizados para la transmisión de eventos en tiempo real.

cache.php:
Define configuraciones para el sistema de almacenamiento en caché, permitiendo especificar el controlador de caché, la duración y otras opciones relevantes.
database.php:
Contiene configuraciones relacionadas con la conexión a la base de datos, como el tipo de conexión, el nombre de la base de datos, el nombre de usuario y la contraseña.
filesystem.php:
Define configuraciones para la manipulación de archivos y sistemas de archivos. Aquí se puede especificar los discos de almacenamiento utilizados por la aplicación.
mail.php:
Contiene configuraciones para el envío de correos electrónicos, incluyendo los controladores de correo, las direcciones de correo electrónico predeterminadas y otras opciones relacionadas con el correo.
queue.php:
Define configuraciones para la gestión de colas de trabajos (queues). Aquí se puede configurar diferentes controladores de colas y opciones asociadas.
session.php:
Contiene configuraciones para el manejo de sesiones en la aplicación, incluyendo el controlador de sesión, la duración de la sesión y otros parámetros relacionados.
view.php:
Define configuraciones para el sistema de vistas de la aplicación. Puede especificar el motor de plantillas, ubicaciones de vistas y otras opciones relevantes.

El directorio database
La carpeta "database/" en Laravel juega un papel esencial en la gestión de la base de datos de la aplicación. Aquí hay una descripción de los subdirectorios y archivos comunes dentro de esta carpeta:
Factories:
Esta carpeta contiene archivos de factories que definen la estructura de los datos de prueba utilizados en la aplicación. Las factories facilitan la creación de registros ficticios para pruebas y desarrollo.
Migrations:
En esta carpeta se encuentran los archivos de migración. Las migraciones son archivos que describen las modificaciones en la estructura de la base de datos. Permiten realizar cambios en la base de datos de manera controlada y versionada.
Seeders:

Aquí están los archivos de seeders, los cuales permiten poblar la base de datos con datos de prueba. Son útiles para generar conjuntos de datos iniciales o de prueba durante el desarrollo.

El directorio public

La carpeta "public/" es fundamental en Laravel, ya que almacena los archivos accesibles públicamente desde la web. Aquí está una descripción de los elementos comunes dentro de esta carpeta:

css:

Esta carpeta contiene archivos CSS que se utilizan para definir el estilo visual de las páginas web de la aplicación.

images:

Almacena archivos de imágenes utilizados en la aplicación. Las imágenes aquí se pueden referenciar directamente desde las vistas.

js:

Contiene archivos JavaScript utilizados para proporcionar funcionalidades interactivas en el lado del cliente. Estos archivos pueden ser referenciados desde las vistas.

storage:

Enlaces simbólicos a la carpeta "storage/app/public". Laravel usa este enlace para acceder a los archivos almacenados en el sistema de almacenamiento. Es necesario ejecutar el comando php artisan storage:link para crear este enlace simbólico.

.htaccess:

El archivo .htaccess contiene configuraciones para el servidor web Apache. Puede incluir reglas de reescritura y otras configuraciones específicas para Laravel.

favicon.ico:

Este archivo es el ícono que se muestra en las pestañas del navegador. Se puede personalizar para representar la identidad visual de la aplicación.

index.php:

El archivo index.php es el punto de entrada principal para solicitudes HTTP. Inicia la aplicación Laravel y maneja la ejecución de las solicitudes web.

mix-manifest.json:

Este archivo se genera automáticamente cuando se compilan los activos con Laravel Mix. Contiene un mapeo de activos compilados a sus rutas correspondientes y se utiliza para versionar y referenciar activos compilados en la aplicación.

El directorio resources

La carpeta "resources/" es un componente clave en la estructura de un proyecto Laravel, ya que almacena recursos y vistas que son esenciales para la construcción y presentación de la aplicación. A continuación, se describe el propósito de los subdirectorios y archivos:

css:
Aquí se encuentran archivos CSS utilizados para definir estilos visuales en la interfaz de usuario de la aplicación. Estos archivos se pueden compilar y optimizar con herramientas como Laravel Mix.

js:
Contiene archivos JavaScript que proporcionan funcionalidades interactivas en el lado del cliente.

lang:
Esta carpeta almacena archivos de traducción para admitir la internacionalización (i18n) y localización (l10n) de la aplicación.

sass:
Similar a la carpeta "css", pero específicamente diseñada para archivos Sass. Sass es un preprocesador de CSS que permite una escritura más eficiente y modular del código CSS.

views:
Aquí es donde se encuentran las vistas de la aplicación. Las vistas son archivos que contienen el HTML y incorporan elementos dinámicos utilizando el motor de plantillas de Blade. Las subcarpetas pueden organizarse por controlador o por sección de la aplicación.

La carpeta routes:
La carpeta "routes/" es fundamental para definir las rutas de tu aplicación, determinando cómo responden a las solicitudes HTTP. A continuación, se describen los archivos de esta carpeta:

api.php:
Este archivo se utiliza para definir las rutas de la API de la aplicación. Aquí se especifican las rutas que manejan solicitudes HTTP para recursos de la API, como controladores y recursos de Eloquent.

channels.php:
Este archivo contiene rutas de canales que definen cómo los eventos son transmitidos a los clientes conectados a través de WebSockets.

console.php:
En este archivo, se puede definir comandos personalizados de Artisan. Estos comandos permiten realizar tareas de consola específicas de la aplicación y se ejecutan mediante la interfaz de línea de comandos de Artisan.

web.php:

Este archivo se utiliza para definir las rutas web de la aplicación. Aquí es donde se especifica las rutas que responden a solicitudes HTTP desde navegadores web.

La carpeta storage
La carpeta "storage/" en Laravel es crucial para almacenar archivos generados y gestionar recursos que no deberían ser accesibles directamente desde la web. Aquí hay una descripción de los subdirectorios:
app:
Esta carpeta almacena archivos generados por la aplicación. Se pueden incluir archivos de usuario, archivos de carga, y otros recursos generados dinámicamente que no deben ser accesibles directamente desde la web.
framework:
Contiene subdirectorios adicionales que almacenan archivos generados por Laravel. Por ejemplo, las carpetas "cache/", "sessions/", y "testing/" pueden encontrarse aquí. Laravel utiliza estos directorios para almacenar información temporal y optimizar el rendimiento de la aplicación.
logs:
Aquí es donde Laravel almacena sus registros de eventos y errores. Los archivos de registro proporcionan información valiosa para la depuración y el monitoreo del rendimiento de la aplicación.

7. Implementación

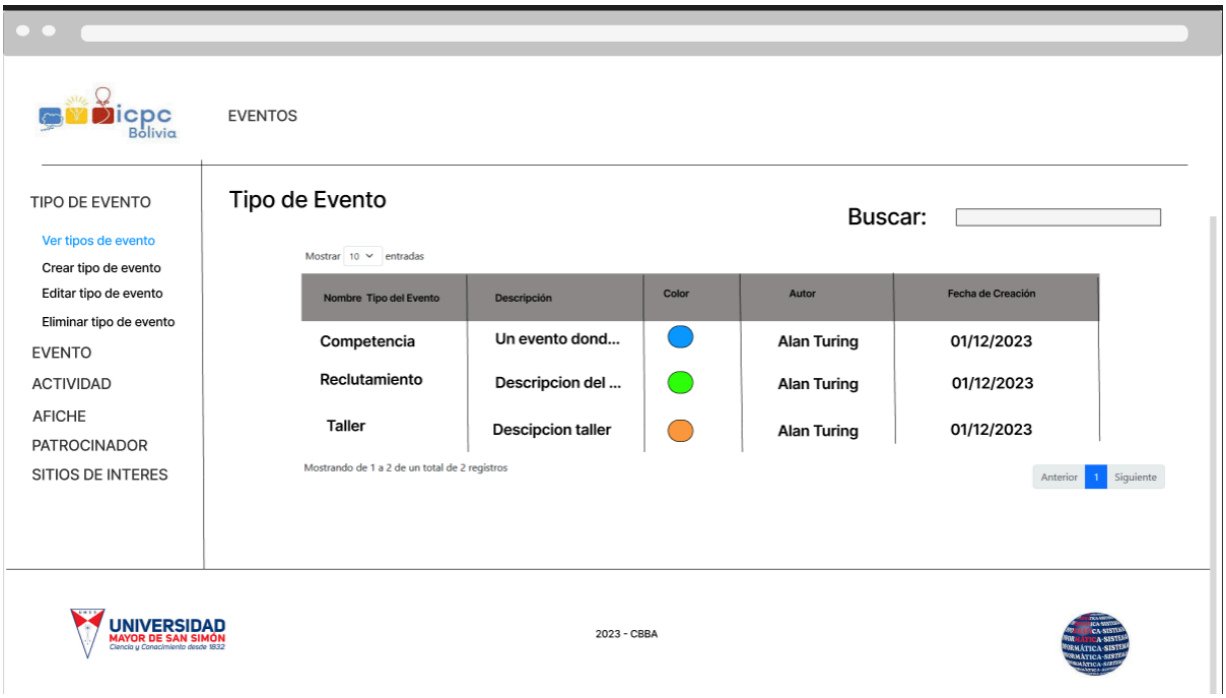
7.1. TIPO DE EVENTO

7.1.1. Ver tipos de evento

7.1.1.1. Historia de usuario de ver tipos de evento

Historia de Usuario	
Título: Mostrar tipo de evento	**ID: 2**
Estimación: 1	**Importancia: Alta**
Descripción Yo como usuario con privilegios para crear tipos de eventos, quiero visualizar los distintos tipos de evento que haya creado previamente con los campos de “Nombre del tipo de evento”, “Descripción”, “Color de referencia”, “Autor” y “Fecha de creación” para saber los datos de tipos de evento que existen.	

MockUps



Criterios de aceptación:

1. Al hacer clic en la sección “TIPO DE EVENTO” en el menú lateral y seleccionar la opción “Ver tipos de evento” se muestra una tabla con los tipos de eventos creados.
2. La tabla tiene las columnas “Nombre del tipo de evento”, “Descripción”, “Color de referencia”, “Autor” y “Fecha de creación”.
3. Si el tipo de evento no tiene una descripción registrada, el campo en la columna “Descripción” está vacío.
4. Cuando la descripción es muy larga se trunca el texto con “…”
5. Se muestra una descripción emergente para el campo descripción.

7.1.1.2. Código fuente de ver tipos de evento

Controlador de ver tipos de evento

Este método se encarga de obtener todos los registros de la tabla 'TipoEvento' utilizando el modelo correspondiente. Luego, devuelve una vista llamada 'tipos-de-evento/tiposDeEvento', proporcionando los tipos de eventos como datos para ser utilizados en la vista. La vista puede acceder a los tipos de eventos a través de la variable $tiposDeEventos.

Ruta del archivo: app\Http\Controllers\TipoEventoController.php

```php
/**
 * Muestra la vista que presenta todos los tipos de eventos.
 *
 * @return \Illuminate\Http\Response
 */
public function mostrarVistaTipoEvento()
{
    // Recupera todos los tipos de eventos de la base de datos
    $tiposDeEventos = TipoEvento::all();

    // Devuelve la vista 'tipos-de-evento/tiposDeEvento' con los tipos de eventos
    // pasados como datos para ser utilizados en la vista
    return view('tipos-de-evento/tiposDeEvento', ['tiposDeEventos' => $tiposDeEventos]);
}
```

Codigo JavaScript de ver tipos de evento

Se encarga de la inicialización y gestión de una tabla DataTable para la visualización de tipos de eventos. Se declaran variables globales, como tablaDeTipos para almacenar la instancia DataTable, y tablaInicializada para verificar si la tabla ya ha sido inicializada.

Las opciones de DataTable, definidas en dataTableOptions, incluyen configuraciones como el orden inicial, la cantidad de entradas por página y el idioma. La función initDataTable se encarga de inicializar la DataTable con estas opciones, destruyendo previamente la tabla si ya ha sido inicializada.

Adicionalmente, hay una función llamada cargarTiposDeEvento que simula un breve retraso antes de redirigir a la página de administración de tipos de evento (/admin/tipos-de-evento), proporcionando una transición suave para el usuario.

Ruta del archivo: public/js/TipoDeEvento/tipoDeEvento.js

```
// Declaración de variables globales
let tablaDeTipos;
let tablaInicializada = false;

// Opciones de DataTable
const dataTableOptions = {
    order: [[0, "desc"]],
    pageLength: 10,
    lengthMenu: [5, 10, 15, 20],
    destroy: true,
    order: [[5, "desc"]],
    language: {
        lengthMenu: "Mostrar _MENU_ entradas",
        zeroRecords: "Ningún tipo de evento encontrado",
        info: "Mostrando de _START_ a _END_ de un total de _TOTAL_ registros",
        infoEmpty: "Ningún tipo de evento encontrado",
        infoFiltered: "(filtrados desde _MAX_ registros totales)",
        search: "Buscar:",
        loadingRecords: "Cargando...",
        paginate: {
            first: "Primero",
            last: "Último",
            next: "Siguiente",
            previous: "Anterior",
        },
    },
};

// Inicialización de DataTable
const initDataTable = async () => {
    // Destruir la tabla si ya está inicializada
    if (tablaInicializada) {
        tablaDeTipos.destroy();
    }

    // Inicializar DataTable con opciones
    DataTable.datetime("DD-MM-YYYY");
    tablaDeTipos = $("#tablaTipoDeEvento").DataTable(dataTableOptions);
```

```
    // Actualizar el estado de inicialización
    tablaInicializada = true;
};

// Función para cargar tipos de evento y redirigir a la página de administración
const cargarTiposDeEvento = async () => {
    // Retraso simulado para mostrar un mensaje antes de redirigir
    setTimeout(() => {
        // Redirigir a la página de administración de tipos de evento
        window.location.href = "/admin/tipos-de-evento";
    }, 1750);
};

// Evento de carga de la ventana
window.addEventListener("load", async () => {
    // Inicializar DataTable al cargar la página
    initDataTable();
});
```

Vista de ver tipos de evento

Este código Blade forma parte de la interfaz de usuario de una aplicación web desarrollada en Laravel, un framework de PHP. Su propósito principal es mostrar una lista de tipos de eventos en una interfaz clara y estructurada, permitiendo la visualización y gestión eficiente de la información.

Ruta del archivo: resources/views/tipos-de-evento/tiposDeEvento.blade.php

```
@extends('layouts.app')

@section('content')
    <!-- Contenedor principal -->
    <div class="container">
        <!-- Encabezado de la página -->
        <div class="row align-items-end">
            <!-- Título de la página -->
            <div class="col-md-6">
                <h2>Tipos de evento</h2>
            </div>
            <!-- Botón para abrir el modal de creación de tipo de evento -->
            <div class="col-md-1">
                <x-ModalCrearTipoEvento />
            </div>
            <!-- Espacio adicional en el encabezado -->
            <div class="col-md-9"></div>
        </div>

        <!-- Cuerpo de la página con la tabla de tipos de evento -->
        <div class="row mt-3">
            <table class="table table-striped text-secondary" id="tablaTipoDeEvento">
                <caption>Tipos de evento</caption>
                <thead>
                    <!-- Encabezados de la tabla -->
                    <tr>
                        <th scope="col" class="col-md-3" data-visible="false">Ordenador</th>
                        <th scope="col" class="col-md-3">Nombre del tipo de evento</th>
                        <th scope="col" class="col-md-2">Descripción</th>
                        <th scope="col" class="col-md-2 text-center">Color de referencia</th>
                        <th scope="col" class="col-md-2 text-center">Autor</th>
                        <th scope="col" class="col-md-3 text-center">Fecha de creación</th>
                    </tr>
                </thead>
                <tbody id="datosTabla">
                    <!-- Iteración sobre los tipos de eventos para llenar la tabla -->
                    @foreach ($tiposDeEventos as $tipoDeEvento)
                        <tr>
                            <td>{{ $tipoDeEvento->created_at }}</td>
                            <td>{{ $tipoDeEvento->nombre }}</td>
                            <td title="{{ $tipoDeEvento->descripcion }}">
                                <!-- Descripción truncada para visualización -->
                                <span class="d-inline-block text-truncate" style="max-width: 150px;">
                                    {{ $tipoDeEvento->descripcion }}
                                </span>
                            </td>
                            <td class="container-color">
                                <!-- Celda de color de referencia -->
                                <div class="color-cell" style="background-color:{{ $tipoDeEvento->color }};"></div>
                            </td>
                            <td class="text-center">Yo</td>
                            <td class="text-center">{{ date('d-m-Y', strtotime($tipoDeEvento->created_at)) }}</td>
                        </tr>
                    @endforeach
                </tbody>
            </table>
        </div>
    </div>
```

```
    <!-- Enlaces y scripts adicionales -->
    <link href="https://cdn.datatables.net/1.13.6/css/dataTables.bootstrap5.min.css" rel="stylesheet">
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.11.1/font/bootstrap-icons.css">
    <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.7.1/jquery.min.js"></script>
    <script src="https://cdn.datatables.net/1.13.6/js/jquery.dataTables.min.js"></script>
    <script src="https://cdn.datatables.net/1.13.6/js/dataTables.bootstrap5.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/moment.js/2.29.2/moment.min.js"></script>
    <script src="{{ asset('js/TipoDeEvento/tipoDeEvento.js') }}" defer></script>
@endsection
```

7.1.1.3. Interfaz de usuario de ver tipos de evento

La interfaz de usuario para la vista "Ver Tipos de Evento" presenta una tabla organizada con la siguiente información:

Nombre de Tipo de Evento:
En la primera columna, se muestra el nombre de cada tipo de evento, permitiendo una rápida identificación.
Descripción:
La columna de descripción proporciona información adicional sobre cada tipo de evento, destacando características distintivas o detalles relevantes.
Color de Referencia:
Cada tipo de evento se identifica visualmente mediante un color de referencia, facilitando la diferenciación entre ellos.
Autor:
La columna "Autor" indica quién creó o registró inicialmente el tipo de evento.
Fecha de Creación:
La fecha de creación muestra cuándo se agregó inicialmente cada tipo de evento al sistema.

7.1.2. Crear tipo de evento

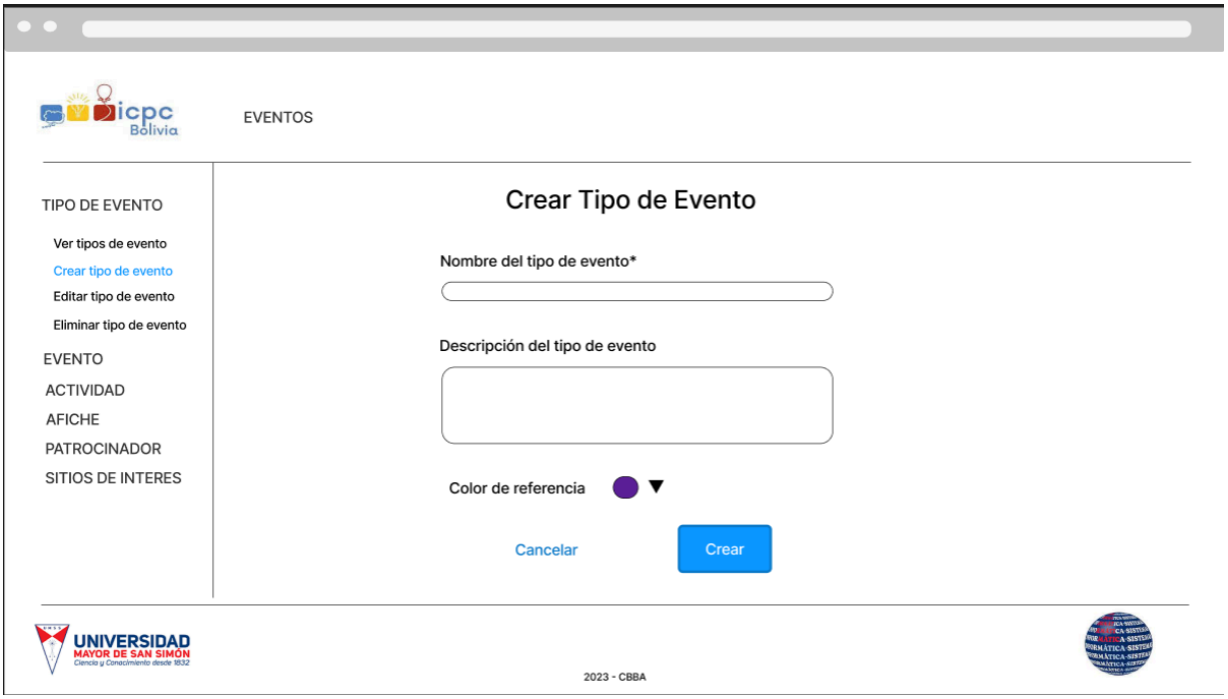
7.1.2.1. Historia de usuario de crear tipo de evento

Historia de Usuario	
Título: Crear un tipo de evento	**ID: 1**
Estimación: 2	**Importancia: Alta**

Descripción:

Yo como usuario con privilegios de crear un tipo de evento, quiero crear un tipo de evento, para poder asignar estos tipos creados a mis eventos.

MockUps



Criterios de aceptación

1. Al hacer clic en la sección de “TIPO DE EVENTO” en el menú lateral y seleccionar la opción “Crear tipo de evento”, se abre un formulario con los siguientes campos: “Nombre del tipo de evento *”, “Descripción del tipo de evento", “Color de referencia” y los botones "Cancelar " y "Crear ".
2. El campo “Nombre del tipo de evento *” es obligatorio.
3. El campo “Nombre del tipo de evento *” es único.
4. El campo “Nombre del tipo de evento *” tiene 64 caracteres como máximo.

5. El campo “Descripción del tipo de evento” es opcional.
6. El campo “Descripción del tipo de evento” tiene un máximo de 500 caracteres.
7. Al hacer clic en el campo “Color de referencia” se despliega un selector de color, específico del navegador.
8. El campo “Color de referencia” tendrá de color por defecto “morado”.
9. Al seleccionar un color de la paleta de colores, el campo “Color de referencia” cambia su tono al color seleccionado.
10. Al hacer clic en el botón “Crear” en el formulario se controla que el nombre del tipo de evento no sea nulo.
11. Si el nombre del tipo de evento está vacío, al hacer clic en el botón “Crear”, aparece un mensaje de validación “El nombre no puede estar vacío.” debajo del campo “Nombre del tipo de evento *”.
12. Si el nombre del tipo de evento ya fue registrado, al hacer clic en el botón “Crear”, aparece una alerta temporal con el mensaje “El tipo de evento ya existe”.
13. Si el nombre del tipo de evento ya fue registrado, al hacer clic en el botón “Crear”, aparece una mensaje de validación “El tipo de evento ya existe” debajo del campo “Nombre del tipo de evento”.
14. Al hacer clic en el botón “Cancelar” se limpian los campos.
15. Si los campos cumplen lo requerido, al hacer clic en el botón “Crear” aparece una alerta temporal con el mensaje de éxito “Creado exitosamente”.
16. Si el nombre del tipo de evento corresponde con el nombre de un tipo de evento eliminado, al hacer clic en el botón “Crear”, aparece un modal con el mensaje “El tipo de evento que intentó crear ya existe, ¿Desea habilitarlo nuevamente? ”, con dos botones, “No” y “Sí”.
17. Al hacer clic en “Sí”, se abre un modal con el título “¿Desea actualizar los datos del tipo de evento?”, en el cual se muestra tanto la información previa como la nueva del tipo de evento.
18. Al hacer clic en “Sí”, en el modal con título “¿Desea actualizar los datos del tipo de evento?”, el tipo de evento es restaurado con la información nueva.
19. Al hacer clic en “No”, en el modal con título “¿Desea actualizar los datos del tipo de evento?”, el tipo de evento es restaurado con la información previa.
20. Al hacer clic en “No” o “Sí”, en el modal con título “¿Desea actualizar los datos del tipo de evento?”, aparece una alerta temporal con el mensaje de éxito “Restaurado exitosamente”.
21. Al finalizar el tiempo de la alerta temporal, se redirige a la sección “Ver tipos de evento”.

7.1.2.2. Código fuente de crear tipo de evento

Controlador tipo de evento

Este método crea un nuevo tipo de evento usando los datos proporcionados en la solicitud y lo almacena en la base de datos. Devuelve una respuesta JSON exitosa si la operación

es exitosa. Maneja posibles errores, como la duplicación de tipos de eventos o campos demasiado grandes, proporcionando mensajes de error específicos en formato JSON.

Ruta del archivo: app/Http/Controllers/TipoEventoController.php

```php
/**
 * Almacena un nuevo tipo de evento en la base de datos.
 *
 * @return \Illuminate\Http\Response
 */
public function store(Request $request)
{
    try {
        // Crear una nueva instancia del modelo TipoEvento
        $tipo_evento = new TipoEvento();
        // Asignar valores a las propiedades del tipo de evento desde los datos proporcionados en la
        //solicitud
        $tipo_evento->nombre = $request->nombre;
        $tipo_evento->descripcion = $request->descripcion;
        $tipo_evento->color = $request->color;
        // Guardar el nuevo tipo de evento en la base de datos
        $tipo_evento->save();
        // Devolver una respuesta JSON exitosa
        return response()->json(['mensaje' => 'Creado exitosamente', 'error' => false]);
    } catch (QueryException $e) {
        // Capturar excepciones de consulta (errores de la base de datos)
        $errorCode = $e->errorInfo[1];

        if ($errorCode == 1062) {
            // Error de duplicación de clave única (por ejemplo, ya existe un tipo de evento con el
            //mismo nombre)
            $respuesta = response()->json(['mensaje' => 'El tipo de evento ya existe', 'error' => true]);
        } else {
            if ($errorCode == 1406) {
                // Error de longitud de campo demasiado grande
                $respuesta = response()->json(['mensaje' => 'Campo demasiado grande', 'error' => true]);
            } else {
                // Otro tipo de error de base de datos
                $respuesta = response()->json(['mensaje' => $e->getMessage(), 'error' => true]);
            }
        }
        // Devolver la respuesta según el tipo de error capturado
        return $respuesta;
    }
}
```

Código JavaScript de crear tipo de evento

Se centra en la gestión de un formulario para la creación y administración de tipos de eventos en una aplicación web. El código utiliza la biblioteca Axios para realizar solicitudes HTTP a la API del servidor. El formulario incluye campos como nombre, descripción y color, y se integran validaciones en tiempo real para garantizar la integridad de los datos ingresados.

La función crearTipoEvento procesa la creación de un nuevo tipo de evento, mostrando mensajes de alerta según la respuesta del servidor. En caso de que ya exista un tipo de evento con el mismo nombre, se captura y almacena el nombre anterior para su validación. Además, se incluye la funcionalidad de restaurar tipos de evento eliminados y actualizar la interfaz mediante modales.

La validación en tiempo real aborda la existencia previa de nombres de tipo de evento, destacando visualmente los campos que requieren atención. Se implementa una función para limpiar las validaciones y restablecer el formulario cuando sea necesario.

Ruta del archivo: public/js/TipoDeEvento/crearTipoEvento.js

```
// Declaración de variables
const form = document.getElementById("formularioTipoEvento");
const botonCancelar = document.getElementById("cancelarBoton");
const inputNombre = document.getElementById("nombreTipoEvento");
const mensajeNombre = document.getElementById("mensajeNombre");
const inputDescripcion = document.getElementById("detalleTipoEvento");
const color = document.getElementById("colorTipoEvento");
let nombreAnterior = ""; // Almacena el nombre previo para validación
let idTipoEvento; // Almacena el ID del tipo de evento

// Función para crear un nuevo tipo de evento mediante una solicitud POST
const crearTipoEvento = (formData) => {
    axios.post("/api/tipo-evento", formData)
        .then(function (response) {
            // Manejo de la respuesta del servidor
            console.log(response.data);
            const mensaje = response.data.mensaje;
            const nombreIgual = "El tipo de evento ya existe";

            // Mostrar mensaje de alerta según la respuesta del servidor
            if (response.data.borrado) {
                idTipoEvento = response.data.id;
                $("#modalTipoEventoExistente").modal("show");
            } else {
                mostrarAlerta(
                    response.data.error ? "Peligro" : "Éxito",
                    response.data.mensaje,
                    response.data.error ? "danger" : "success"
                );
            }

            // Validar y manejar la existencia previa del tipo de evento
            if (mensaje === nombreIgual) {
                nombreAnterior = inputNombre.value;
                inputNombre.classList.remove("is-valid");
                inputNombre.classList.add("is-invalid");
                inputDescripcion.classList.add("is-valid");
                mensajeNombre.textContent = "El tipo de evento ya existe";
            } else {
```

```
                    form.querySelectorAll(".form-control, .form-select").forEach(
                        (Element) => {
                            Element.classList.remove("is-valid");
                        }
                    );
                    if (!response.data.borrado) {
                        form.reset();
                        setTimeout(() => {
                            window.location.href = "/admin/tipos-de-evento";
                        }, 1800);
                    }
                }
            })
            .catch(function () {
                mostrarAlerta(
                    "Error",
                    "Hubo un error al guardar el tipo de evento",
                    "danger"
                );
            });
};

// Función para restaurar un tipo de evento previamente eliminado
const restaurarTipoEvento = async () => {
    await axios.post("/api/tipo-evento/restaurar/" + idTipoEvento)
        .then(() => {
            $("#modalTipoEventoExistente").modal("hide");
            $("#modalActualizarTipoEvento").modal("show");
        })
        .catch((error) => {
            console.log(error);
        });
    modalActualizar();
};

// Función para actualizar un tipo de evento
const actualizarTipoEvento = async (bb) => {
    const formulario = document.getElementById("formularioTipoEvento");
    const formData = new FormData(formulario);
```

```
    let error = false;
    let mensaje = "Restaurado exitosamente.";

    if (bb) {
        const response = await axios.post(
            "/api/tipo-evento/actualizar/" + idTipoEvento,
            formData
        );
        error = response.data.error;
        mensaje = response.data.mensaje;

        $("#modalActualizarTipoEvento").modal("hide");
    }
    mostrarAlerta(
        error ? "Peligro" : "Éxito",
        mensaje,
        error ? "danger" : "success"
    );
    setTimeout(() => {
        window.location.href = "/admin/tipos-de-evento";
    }, 1800);
};

// Evento de envío del formulario
form.addEventListener("submit", (event) => {
    event.preventDefault();
    form.querySelectorAll(".form-control, .form-select").forEach((Element) => {
        Element.dispatchEvent(new Event("change"));
    });
    if (validar()) {
        const formData = new FormData(form);
        crearTipoEvento(formData);
    }
});
```

```
// Función para mostrar el modal de actualización de tipo de evento
const modalActualizar = async () => {
    let nuevo = document.getElementById("nuevoNombre");
    let nuevaDescripcion = document.getElementById("nuevoDescripcion");
    let nuevoColor = document.getElementById("nuevoColor");
    let nombreAnterior = document.getElementById("antiguoNombre");
    let colorAnterior = document.getElementById("antiguoColor");
    let descripcionAnterior = document.getElementById("antiguoDescripcion");
    const response = await axios.get("/api/tipo-evento/" + idTipoEvento);
    nombre.textContent = inputNombre.value;
    nuevaDescripcion.textContent = inputDescripcion.value;
    nuevoColor.style.backgroundColor = color.value;
    colorAnterior.style.backgroundColor = response.data.color;
    descripcionAnterior.textContent = response.data.descripcion;
};

// Función para validar el formulario
const validar = () => {
    return form.querySelector(".is-invalid") === null;
};

// Agregar validación a los inputs
form.querySelectorAll(".form-control, .form-select").forEach((Element) => {
    Element.addEventListener("change", () => {
        if (Element.hasAttribute("required") && Element.value === "") {
            Element.classList.remove("is-valid");
            Element.classList.add("is-invalid");
        } else {
            Element.classList.remove("is-invalid");
            Element.classList.add("is-valid");
        }
    });
});
```

```
// Función para validar nombre repetido
function validarNombreRepetido() {
    if (nombreAnterior === "") {
        if (inputNombre.value === "") {
            inputNombre.classList.remove("is-valid");
            inputNombre.classList.add("is-invalid");
            mensajeNombre.textContent = "El nombre no puede estar vacío.";
        } else {
            inputNombre.classList.remove("is-invalid");
            inputNombre.classList.add("is-valid");
        }
    } else if (
        inputNombre.value !== nombreAnterior &&
        inputNombre.value !== ""
    ) {
        inputNombre.classList.remove("is-invalid");
        inputNombre.classList.add("is-valid");
        mensajeNombre.textContent = "";
    } else if (inputNombre.value === "") {
        inputNombre.classList.remove("is-valid");
        inputNombre.classList.add("is-invalid");
        mensajeNombre.textContent = "El nombre no puede estar vacío.";
    } else {
        inputNombre.classList.remove("is-valid");
        inputNombre.classList.add("is-invalid");
        mensajeNombre.textContent = "El tipo de evento ya existe";
    }
}

// Eventos de input y cambio para validar nombre repetido
inputNombre.addEventListener("input", validarNombreRepetido);
inputNombre.addEventListener("change", validarNombreRepetido);

// Función para quitar validación
function quitarValidacion() {
    form.querySelectorAll(".form-control, .form-select").forEach((Element) => {
        Element.classList.remove("is-valid");
        Element.classList.remove("is-invalid");
    });
}

// Función para limpiar el formulario
const limpiarForm = () => {
    form.reset();
    quitarValidacion();
    $("#modalTipoEventoExistente").modal("hide");
};
```

Vista de crear tipo de evento

La vista presenta un formulario para la creación de un tipo de evento, con campos para el nombre, descripción y color de referencia. Se incluyen validaciones de formulario y se

utiliza el componente modal para manejar la situación en la que el tipo de evento ya existe. El formulario utiliza AJAX para enviar la solicitud de creación al controlador, y el archivo JavaScript crearTipoEvento.js maneja la lógica del formulario y las interacciones con el usuario. Se utiliza un modal adicional (modalActualizarTipoEvento) para confirmar la actualización de datos en caso de cambios.

Ruta del archivo: resources/views/tipos-de-evento/crearTiposDeEvento.blade.php

```
@extends('layouts.app')
@section('content')
    <div class="container">
        <div class="row justify-content-center">
            <div class="col-md-5" style="min-height: 500px">
                <h2 class="text-center">Crear tipo de evento</h2>
                <form id="formularioTipoEvento" class="needs-validation" novalidate>
                    @csrf
                    <div class="container">
                        <!-- Sección para el nombre del tipo de evento -->
                        <div class="row">
                            <div class="col-md-12">
                                <label for="nombreTipoEvento" class="form-label">Nombre del tipo de evento *</label>
                                <input name="nombre" type="text" class="form-control custom-input" id="nombreTipoEvento"
                                    value="" required placeholder="Ingrese el nombre del tipo de evento"
                                    maxlength="64" autocomplete="off">
                                <div id="mensajeNombre" class="invalid-feedback">
                                    El nombre no puede estar vacío.
                                </div>
                            </div>
                        </div>

                        <!-- Sección para la descripción del tipo de evento -->
                        <div class="row">
                            <div class="col-md-12 mt-3">
                                <label for="detalleTipoEvento" class="form-label">Descripción del tipo de evento</label>
                                <textarea name="descripcion" class="form-control" id="detalleTipoEvento" rows="6" style="resize: none;"
                                    placeholder="Ingrese una descripción..." maxlength="500"></textarea>
                            </div>
                        </div>

                        <!-- Sección para el color de referencia del tipo de evento -->
                        <div class="row">
                            <div class="col-md-12">
                                <div class="row contenedor-titulo-color">
                                    <div class="contenedor-subtitulo">
                                        <div class=" mt-3">
                                            <label for="colorTipoEvento" class="form-label text-container">Color de referencia</label>
                                        </div>
                                    </div>
                                    <div class=" mt-2 custom-col colum-col">
                                        <input name="color" type="color" class="form-control-color controlador"
                                            id="colorTipoEvento" value="#563d7c" title="Choose your color">
                                    </div>
                                </div>
                            </div>
                        </div>

                        <!-- Botones para cancelar y confirmar la creación del tipo de evento -->
                        <div class="text-center my-5">
                            <button id="cancelarBoton" type="reset" class="btn btn-secondary mx-5"
                                onClick="quitarValidacion()">Cancelar</button>
                            <button id="confirmarBoton" type="submit" class="btn btn-primary mx-5">Crear</button>
                        </div>
                    </form>

                    <!-- Componente Modal para Tipo de Evento Existente -->
                    @component('components.modal')
                        @slot('modalId', 'modalTipoEventoExistente')
                        @slot('modalTitle', 'Tipo de evento ya existe')
                        @slot('modalContent')
```

```
                        <p>
                            <b>El tipo de evento que intentó crear ya existe,</b>
                            ¿Desea habilitarlo nuevamente?
                        </p>
                    @endslot
                    @slot('modalButton')
                        <button type="button" class="btn btn-secondary" data-bs-dismiss="modal" onclick="limpiarForm()">
                            No</button>
                        <button type="button" class="btn btn-danger" onclick="restaurarTipoEvento()">Sí</button>
                    @endslot
                @endcomponent

                <!-- Modal para Actualizar Tipo de Evento -->
                <div class="modal fade" id="modalActualizarTipoEvento" tabindex="-1" aria-labelledby="modalAnularLabel"
                    aria-hidden="true">
                    <!-- Contenido del modal para actualizar el tipo de evento -->
                </div>
            </div>
        </div>
    </div>
    <!-- Archivo JavaScript asociado a la creación de tipo de evento -->
    <script src="{{ asset('js/TipoDeEvento/crearTipoEvento.js') }}" defer></script>
@endsection
```

7.1.2.3. Interfaz de usuario de crear tipo de evento

La interfaz de usuario para la creación de un tipo de evento proporciona un formulario intuitivo y fácil de usar. A continuación, se describe cada elemento de la interfaz:

Formulario de Creación de Tipo de Evento:

Nombre del Tipo de Evento: Campo de entrada de texto para ingresar el nombre del tipo de evento. Es un campo obligatorio y admite hasta 64 caracteres.

Descripción del Tipo de Evento: Área de texto que permite ingresar una descripción detallada del tipo de evento. Este campo es opcional y puede contener hasta 500 caracteres.

Color de Referencia: Selector de color que permite elegir un color de referencia para el tipo de evento. Facilita la identificación visual del tipo de evento. Se inicia con un color predeterminado y se puede ajustar según las preferencias del usuario.

Botones:

Cancelar: Un botón secundario que permite descartar los cambios y cerrar el formulario sin realizar ninguna acción.

Crear: Un botón primario que envía el formulario al servidor para crear un nuevo tipo de evento. Este botón desencadena la validación del formulario antes de enviar la solicitud al servidor.

Validación de Formulario:

Nombre del Tipo de Evento: Debe estar presente y no puede estar vacío. Se muestra un mensaje de error si no se proporciona.

Descripción del Tipo de Evento: Acepta hasta 500 caracteres, pero no es obligatorio. Se pueden ingresar caracteres alfabéticos, numéricos y especiales.

Mensajes de Confirmación y Modal:

Éxito en la Creación: Si la creación del tipo de evento es exitosa, se muestra un mensaje de confirmación indicando que se ha creado correctamente.

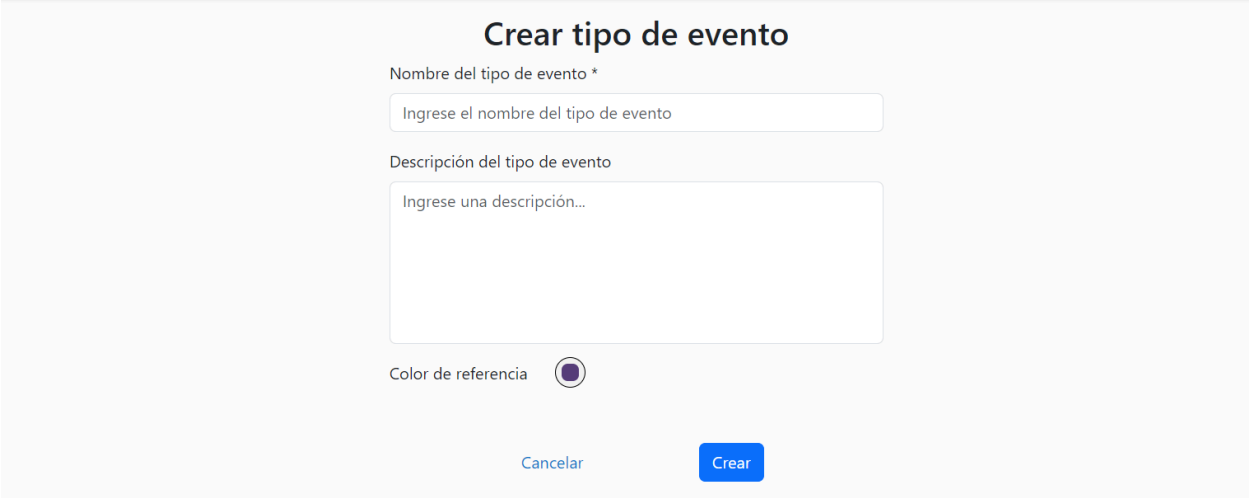
Tipo de Evento Existente: Si el nombre del tipo de evento ya existe, se activa un modal que ofrece opciones para habilitar nuevamente el tipo de evento.

Interacción del Usuario:

Selección de Color: Los usuarios pueden seleccionar un color utilizando el selector de color para asignar un color distintivo al tipo de evento.

Cancelar y Crear:

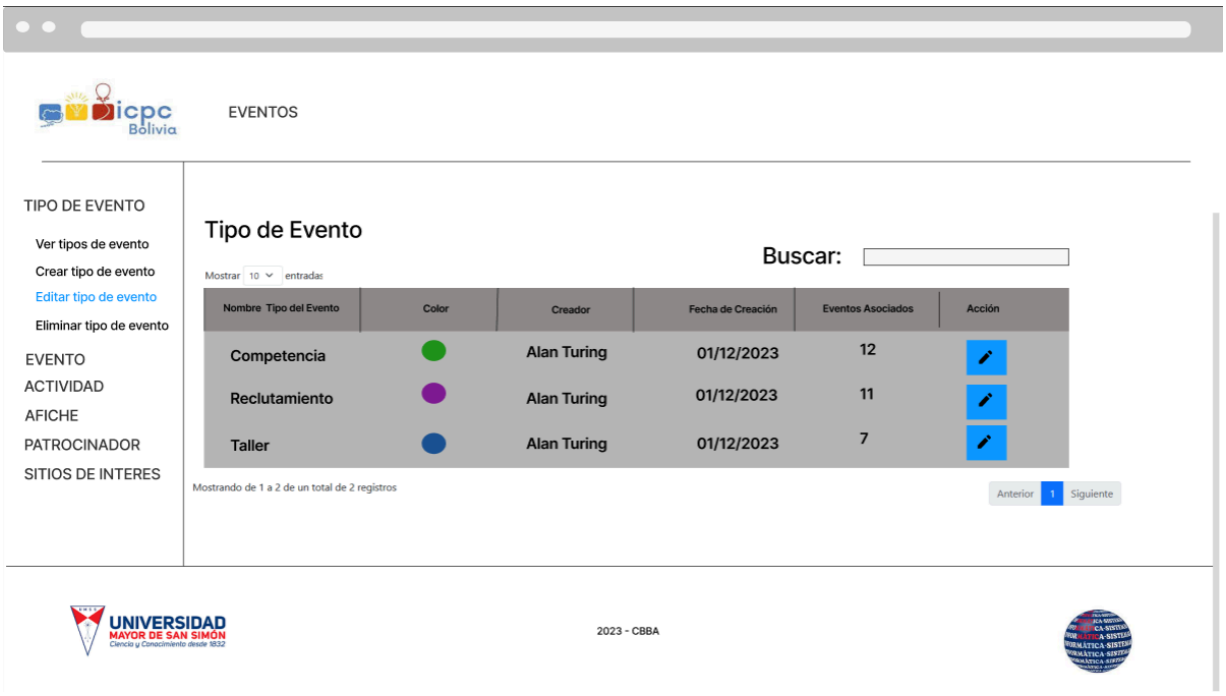
Los botones "Cancelar" y "Crear" ofrecen opciones para descartar o confirmar la creación del tipo de evento, respectivamente.

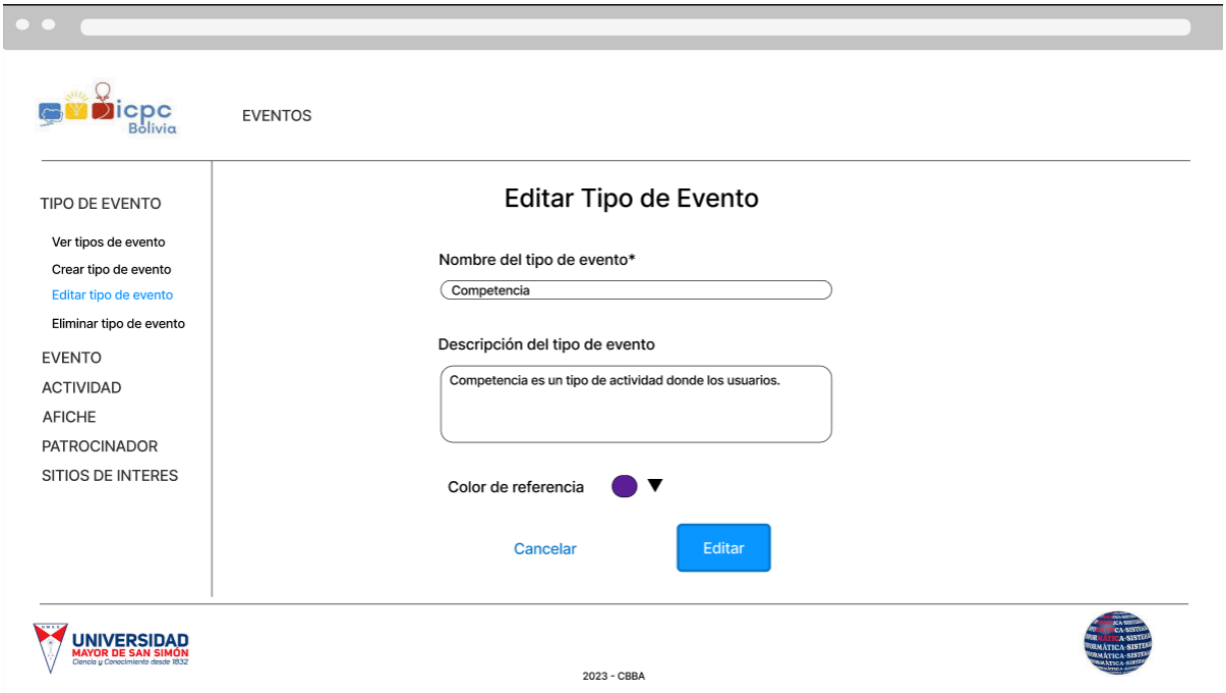


7.1.3. Editar tipo de evento

7.1.3.1. Historia de usuario de editar tipo de evento

Historia de Usuario	
Título: Editar un tipo de evento	**ID: 3**
Estimación: 8	**Importancia: Baja**
Descripción: Yo como usuario con privilegios de administrar tipo de eventos, quiero tener la capacidad de poder actualizar los tipos de eventos para mantener actualizada la lista de tipos de eventos disponibles.	
Mockups	





Criterios de aceptación.

1. Al hacer clic en la sección “TIPO DE EVENTO” en el menú lateral y seleccionar la opción “Editar tipo de evento” se carga una tabla que muestra la información de los tipos de eventos.

2. La tabla tiene las columnas “Nombre del tipo de evento”, “Color de referencia”, “Autor”, “Fecha de creación”, “Eventos asociados” y “Acción”.
3. La columna “Acción” contiene el botón “  ” para cada tipo de evento.
4. Al hacer clic en el botón “  ” de la fila de un tipo de evento, se redirige al formulario de “Editar tipo de evento”, con los campos “Nombre del tipo de evento *”, “Descripción del tipo de evento”, “Color de referencia”, los botones "Cancelar" y "Editar ".
5. El campo "Nombre del tipo de evento" contiene el nombre registrado al crear el tipo de evento.
6. Si el tipo de evento que se está editando tiene eventos asociados, el campo “Nombre del tipo de evento” no puede ser editado.
7. Si el tipo de evento que se está editando no tiene eventos asociados, el campo “Nombre del tipo de evento” es obligatorio.
8. El campo “Nombre del tipo de evento” es único.
9. El campo “Nombre del tipo de evento” tiene 64 caracteres como máximo.
10. El campo “Descripción del tipo de evento” contiene la descripción del tipo de evento registrado al crear el tipo de evento.
11. El campo “Descripción del tipo de evento” es opcional.
12. El campo “Descripción del tipo de evento” tiene un máximo de 500 caracteres.
13. El campo “Color” contiene el color del tipo de evento registrado al crear el tipo de evento..
14. El campo “Color” al hacer clic se despliega un selector de color específico del navegador.
15. Al seleccionar un color de la paleta de colores, el campo de color cambia su tono al color seleccionado.
16. Si el campo “Nombre del tipo de evento” está vacío, al hacer clic en el botón “Editar”, aparece un mensaje de validación “El nombre no puede estar vacío” debajo del campo “Nombre del tipo de evento *”.
17. Si el nombre del tipo de evento ya fue registrado, al presionar el botón “Editar”, aparece una alerta temporal con el mensaje “El tipo de evento ya existe”.
18. Si el nombre del tipo de evento ya fue registrado, al hacer clic en el botón “Editar”, aparece una mensaje de validación “El tipo de evento ya existe” debajo del campo “Nombre del tipo de evento”.
19. Al hacer clic en el botón “Cancelar” se redirige a la sección “Editar tipo de evento”.
20. Si los campos cumplen lo requerido, al presionar el botón “Editar” aparece una alerta temporal con el mensaje de éxito “Actualizado exitosamente”.
21. Si los campos cumplen lo requerido, al presionar el botón “Editar”, se redirige a la sección “Editar tipo de evento”.

7.1.3.2. Código fuente de editar tipo de evento

Controlador de editar tipo de evento

Esta función recibe una solicitud ($request) que contiene los datos actualizados del tipo de evento y el identificador ($id) del tipo de evento que se va a modificar. La función

busca el tipo de evento en la base de datos, actualiza sus atributos y guarda los cambios. En caso de éxito, devuelve una respuesta JSON indicando que la actualización fue exitosa. En caso de errores, maneja específicamente los errores de clave duplicada (nombre ya existente) y otros posibles errores de base de datos, proporcionando mensajes adecuados en la respuesta JSON.

Ruta del archivo: app/Http/Controllers/TipoEventoController.php

```php
/**
 * Actualiza un tipo de evento existente en la base de datos.
 *
 * @return \Illuminate\Http\Response
 */
public function update(Request $request, $id)
{
    try {
        // Buscar el tipo de evento por su identificador
        $tipo_evento = TipoEvento::find($id);
        // Actualizar los campos con los valores proporcionados en la solicitud
        $tipo_evento->nombre = $request->nombre;
        $tipo_evento->descripcion = $request->descripcion;
        $tipo_evento->color = $request->color;
        // Guardar los cambios en la base de datos
        $tipo_evento->save();

        // Respuesta exitosa en formato JSON
        return response()->json(['mensaje' => 'Actualizado exitosamente', 'error' => false,
        'borrado' => false]);

    } catch (QueryException $e) {
        // Capturar excepciones en caso de errores durante la actualización
        // Obtener el código de error específico
        $errorCode = $e->errorInfo[1];

        if ($errorCode == 1062) {
            // Error de clave duplicada (nombre de tipo de evento ya existente)
            $respuesta = response()->json(['mensaje' => 'El tipo de evento ya existe', 'error' => true,
            'borrado' => false]);
        } else {
            // Otros posibles errores, como campos demasiado grandes
            $respuesta = response()->json(['mensaje' => 'Error durante la actualización', 'error' => true,
            'borrado' => false]);
        }
        // Retornar la respuesta en formato JSON
        return $respuesta;
    }
}
```

Código JavaScript de editar tipo de evento

Gestiona la edición de tipos de eventos en un formulario. La funcionalidad se integra con la biblioteca DataTable para presentar y organizar datos tabulares de manera efectiva. Cuando el usuario envía el formulario de edición, se activan eventos que realizan validaciones en tiempo real para asegurar la integridad de los datos ingresados.

La función editarTipoEvento se encarga de enviar una solicitud HTTP asincrónica utilizando Axios para actualizar la información del tipo de evento en el backend. Dependiendo de la respuesta del servidor, se muestra una alerta de éxito o error, y se realizan acciones específicas, como mantener los datos del formulario en caso de un nombre duplicado.

La validación del formulario se lleva a cabo mediante la función validar, que verifica si hay elementos en el formulario que tienen la clase "is-invalid", indicando posibles errores. Se aplican estilos visuales en tiempo real para proporcionar retroalimentación al usuario. Además, se implementa la función validarNombreRepetido para verificar si el nombre del tipo de evento ya existe y realizar acciones correspondientes, como deshabilitar el botón de envío y mostrar mensajes de error.

Ruta del archivo: public/js/TipoDeEvento/editarTipoDeEvento.js

```
// Declaración de variables globales
let tablaDeTipos;
let tablaInicializada = false;
let form = document.getElementById("formularioTipoEvento");
const idTipoEvento = document.getElementById("id");
const botonCancelar = document.getElementById("cancelarBoton");
const inputNombre = document.getElementById("nombreTipoEvento");
const mensajeNombre = document.getElementById("mensajeNombre");
const inputDescripcion = document.getElementById("detalleTipoEvento");
let nombreAnterior = "";

// Opciones para la configuración de la tabla DataTable
const dataTableOptions = {
    pageLength: 10,
    lengthMenu: [5, 10, 15, 20],
    destroy: true,
    order: [[3, "desc"]],
    language: {
        // Configuración de mensajes en la tabla
        lengthMenu: "Mostrar _MENU_ entradas",
        zeroRecords: "Ningún tipo de evento encontrado",
        info: "Mostrando de _START_ a _END_ de un total de _TOTAL_ registros",
        infoEmpty: "Ningún tipo de evento encontrado",
        infoFiltered: "(filtrados desde _MAX_ registros totales)",
        search: "Buscar:",
        loadingRecords: "Cargando...",
        paginate: {
            first: "Primero",
            last: "Último",
            next: "Siguiente",
            previous: "Anterior",
        },
    },
};

// Evento de carga de la ventana
window.addEventListener("load", async () => {
    await initDataTable();
});
```

```
// Inicialización de la tabla DataTable
const initDataTable = async () => {
    if (tablaInicializada) {
        tablaDeTipos.destroy();
    }
    DataTable.datetime("DD-MM-YYYY");
    tablaDeTipos = $("#tablaTipoDeEvento").DataTable(dataTableOptions);
    tablaInicializada = true;
}

// Función para editar un tipo de evento
const editarTipoEvento = (formData) => {
    nombreAnterior = inputNombre.value; // Almacena el valor anterior del nombre
    axios.post(`/api/tipo-evento/actualizar/${idTipoEvento.value}`, formData)
        .then(function (response) {
            const mensaje = response.data.mensaje;
            const nombreIgual = 'El tipo de evento ya existe';

            // Muestra una alerta con el resultado de la operación
            mostrarAlerta(
                "Éxito",
                mensaje,
                response.data.error ? "danger" : "success"
            );

            // Si existe un tipo de evento con el mismo nombre, mantiene los datos del formulario
            if (mensaje === nombreIgual) {
                inputNombre.classList.remove("is-valid");
                inputNombre.classList.add("is-invalid");
                inputDescripcion.classList.add("is-valid");
                mensajeNombre.textContent = 'El tipo de evento ya existe';
            } else {
                // Restablece el formulario y redirige después de un tiempo
                setTimeout(() => {
                    window.location.href = "/admin/tipos-de-evento/editar-tipo";
                }, 1500);
            }
        })
        .catch(function (error) {
```

```
                console.log(mensaje);
                // Muestra una alerta en caso de error
                mostrarAlerta("Error", "Hubo un error al guardar el tipo de evento", "danger");
            })
};

// Evento de envío del formulario
form.addEventListener("submit", (event) => {
    event.preventDefault();
    event.stopPropagation();

    // Dispara el evento "change" en cada elemento del formulario
    form.querySelectorAll(".form-control, .form-select").forEach((element) => {
        element.dispatchEvent(new Event("change"));
    });

    // Si la validación es exitosa, habilita los elementos del formulario y realiza la edición
    if (validar()) {
        form.querySelectorAll(".form-control, .form-select").forEach((element) => {
            if (element.disabled)
                element.disabled = false;
        });
        const formData = new FormData(form);
        editarTipoEvento(formData);
    }
});

// Función de validación
const validar = () => {
    return form.querySelector(".is-invalid") === null;
};

// Agregar validación a los inputs
form.querySelectorAll(".form-control, .form-select").forEach((element) => {
    element.addEventListener("change", () => {
        if (element.hasAttribute("required") && element.value === "") {
            element.classList.remove("is-valid");
            element.classList.add("is-invalid");
        } else {
```

```
                element.classList.remove("is-invalid");
                element.classList.add("is-valid");
            }
        });
});

// Función para validar nombre repetido
function validarNombreRepetido() {
    if (nombreAnterior === '') {
        if (inputNombre.value === '') {
            inputNombre.classList.remove("is-valid");
            inputNombre.classList.add("is-invalid");
            mensajeNombre.textContent = "El nombre no puede estar vacío.";
        } else {
            inputNombre.classList.remove("is-invalid");
            inputNombre.classList.add("is-valid");
        }
    } else {
        if (inputNombre.value !== nombreAnterior && inputNombre.value !== '') {
            inputNombre.classList.remove("is-invalid");
            inputNombre.classList.add("is-valid");
            mensajeNombre.textContent = "";
        } else {
            if (inputNombre.value === "") {
                inputNombre.classList.remove("is-valid");
                inputNombre.classList.add("is-invalid");
                mensajeNombre.textContent = "El nombre no puede estar vacío.";
            } else {
                inputNombre.classList.remove("is-valid");
                inputNombre.classList.add("is-invalid");
                mensajeNombre.textContent = "El tipo de evento ya existe";
            }
        }
    }
}

// Eventos de input y cambio para validar nombre repetido
inputNombre.addEventListener("input", validarNombreRepetido);
inputNombre.addEventListener("change", validarNombreRepetido);

// Función para quitar validación
function quitarValidacion() {
    form.querySelectorAll(".form-control, .form-select").forEach((element) => {
        element.classList.remove("is-valid");
        element.classList.remove("is-invalid");
    });
}
```

Vista de editar tipo de evento

La vista incluye un formulario que permite modificar el nombre, descripción y color de un tipo de evento específico. El formulario envía la información actualizada mediante un método POST al controlador correspondiente. Se implementan validaciones para asegurar que el nombre no esté vacío y respetar la longitud máxima permitida.

Además, la vista maneja casos específicos, como desactivar el campo de nombre si el tipo de evento tiene eventos asociados, lo cual se refleja en el atributo disabled.

Ruta del archivo: resources/views/tipos-de-evento/editarTiposDeEvento.blade.php

```
@extends('layouts.app')

@section('content')
    <!-- Contenedor Principal -->
    <div class="container">
        <!-- Fila Centrada -->
        <div class="row justify-content-center">
            <!-- Columna de Ancho Medio con Altura Mínima -->
            <div class="col-md-5" style="min-height: 500px">
                <!-- Encabezado Centralizado -->
                <h2 class="text-center">Editar tipo de evento</h2>

                <!-- Muestra alertas de éxito y error si existen -->
                @if (session('success'))
                    <div class="alert alert-success">
                        {{ session('success') }}
                    </div>
                @endif
                @if (session('error'))
                    <div class="alert alert-danger">
                        {{ session('error') }}
                    </div>
                @endif

                <!-- Formulario de Edición -->
                <form id="formularioTipoEvento" class="needs-validation" novalidate method="POST"
                    action="{{ route('tipo-eventos.update', ['id' => $tipoEvento->id]) }}">
                    @csrf <!-- Token CSRF para protección -->

                    <!-- Contenedor de Contenido -->
                    <div class="container">
                        <!-- Fila para el Nombre del Tipo de Evento -->
                        <div class="row">
                            <!-- Columna de Ancho Completo -->
                            <div class="col-md-12">
                                <!-- Campo Oculto para el ID del Tipo de Evento -->
                                <input type="hidden" id="id" name="tipoEventoId" value="{{ $tipoEvento->id }}">

                                <!-- Etiqueta y Entrada para el Nombre del Tipo de Evento -->
                                <label for="nombreTipoEvento" class="form-label">Nombre del tipo de evento *</label>
                                <input name="nombre" type="text" class="form-control custom-input" id="nombreTipoEvento"
                                    value="{{ $tipoEvento->nombre }}" required
                                    title="{{ $tipoEvento->eventos->count() > 0 ? 'El nombre del tipo de evento no se puede editar porque tiene eventos asociados' : '' }}"
                                    placeholder="Ingrese el nombre del tipo de evento" maxlength="64"
                                    {{ $tipoEvento->eventos->count() > 0 ? 'disabled' : '' }}>

                                <!-- Mensaje de Retroalimentación para Nombre Vacío -->
                                <div id="mensajeNombre" class="invalid-feedback">
                                    El nombre no puede estar vacío.
                                </div>
                            </div>
                        </div>

                        <!-- Fila para la Descripción del Tipo de Evento -->
                        <div class="row">
                            <!-- Columna de Ancho Completo con Margen Superior -->
                            <div class="col-md-12 mt-3">

                                <!-- Etiqueta y Área de Texto para la Descripción -->
                                <label for="detalleTipoEvento" class="form-label">Descripción del tipo de evento</label>
                                <textarea name="descripcion" class="form-control" id="detalleTipoEvento" rows="6" style="resize: none;"
                                    placeholder="Ingrese una descripción..." maxlength="500">{{ $tipoEvento->descripcion }}</textarea>
                            </div>
                        </div>

                        <!-- Fila para el Color de Referencia -->
                        <div class="row">
                            <!-- Columna de Ancho Completo -->
                            <div class="col-md-12">
                                <!-- Contenedor para Título y Selector de Color -->
                                <div class="row contenedor-titulo-color">
                                    <!-- Contenedor para el Título -->
                                    <div class="contenedor-subtitulo">
                                        <!-- Margen Superior y Etiqueta para el Color de Referencia -->
                                        <div class="mt-3">
                                            <label for="colorTipoEvento" class="form-label text-container">Color de referencia</label>
                                        </div>
                                    </div>
                                    <!-- Contenedor para el Selector de Color -->
                                    <div class="mt-2 custom-col colum-col">
                                        <input name="color" type="color"
                                            class="form-control form-control-color controlador" id="colorTipoEvento"
                                            value="{{ $tipoEvento->color }}" title="Choose your color">
                                    </div>
                                </div>
                            </div>
                        </div>
                    </div>
```

```
                    <!-- Fila para Botones de Cancelar y Editar -->
                    <div class="text-center my-5">
                        <!-- Enlace para Cancelar con Estilo de Botón -->
                        <a href="/admin/tipos-de-evento/editar-tipo" class="btn btn-secondary mx-5">Cancelar</a>
                        <!-- Botón de Confirmación para Editar -->
                        <button id="confirmarBoton" type="submit" class="btn btn-primary mx-5">Editar</button>
                    </div>
                </div>
            </form>
        </div>
    </div>
</div>
<!-- Inclusión del Script para la Lógica de Edición -->
<script src="{{ asset('js/TipoDeEvento/editarTipoEvento.js') }}" defer></script>
@endsection
```

7.1.3.3. Interfaz de usuario de editar tipo de evento

La interfaz de usuario para la edición de un tipo de evento brinda un entorno eficiente y claro para realizar modificaciones. A continuación, se describen los elementos esenciales de la interfaz:

Tabla de Tipos de Evento:

Muestra una tabla que resume los tipos de eventos existentes con las siguientes columnas:

Nombre del Tipo de Evento: Identifica el tipo de evento.

Color de Referencia: Muestra el color asociado al tipo de evento para facilitar la identificación visual.

Autor: Indica el creador del tipo de evento.

Fecha de Creación: Muestra la fecha en que se creó el tipo de evento.

Eventos Asociados: Muestra la cantidad de eventos vinculados al tipo.

Acción (Editar): Presenta un botón con un ícono de lápiz que redirige a la interfaz de edición.

Formulario de Edición de Tipo de Evento:

Nombre del Tipo de Evento: Campo de texto prellenado con el nombre actual del tipo de evento. Permite la edición del nombre, aunque con restricciones si hay eventos asociados.

Descripción del Tipo de Evento: Área de texto que muestra y permite editar la descripción actual del tipo de evento.

Color de Referencia: Selector de color para ajustar o mantener el color asociado al tipo de evento.

Botones:

Cancelar: Botón secundario que descarta los cambios y cierra el formulario sin aplicar ninguna modificación.

Editar: Botón primario que envía los cambios al servidor para actualizar el tipo de evento. Desencadena la validación del formulario antes de procesar la solicitud.

Validación del Formulario:

Nombre del Tipo de Evento: Debe estar presente y no puede estar vacío. Se muestra un mensaje de error si no se proporciona.

Descripción del Tipo de Evento: Acepta hasta 500 caracteres y permite edición.

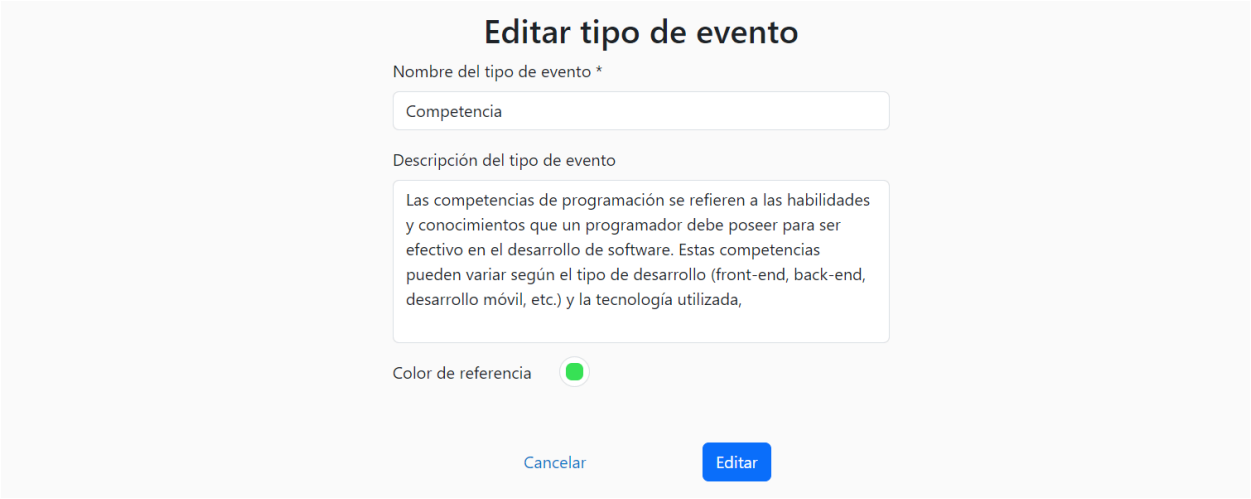
Mensajes de Confirmación y Modal:

Éxito en la Edición: Después de una edición exitosa, se muestra un mensaje de confirmación indicando que el tipo de evento se ha actualizado correctamente.

Interacción del Usuario:

Selección de Color: Los usuarios pueden elegir un nuevo color utilizando el selector de color para actualizar la identificación visual del tipo de evento.

Cancelar y Editar: Los botones "Cancelar" y "Editar" ofrecen opciones para descartar o confirmar las modificaciones en el tipo de evento, respectivamente.



7.1.4. Eliminar tipo de evento

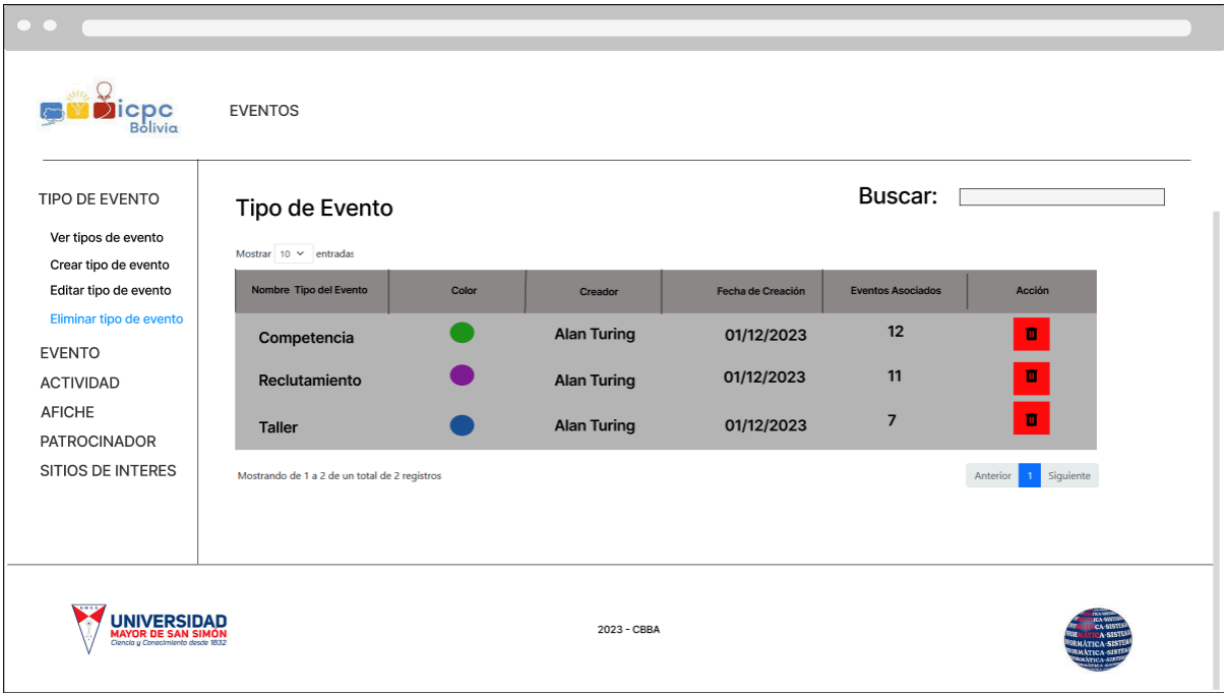
7.1.4.1. Historia de usuario de eliminar tipo de evento

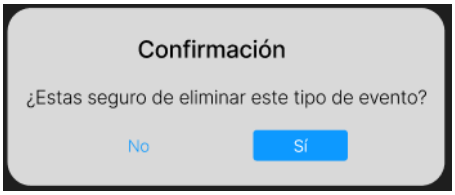
Historia de Usuario	
Título: Eliminar tipo de evento	**ID: 4**
Estimación: 8	**Importancia: Baja**

Descripción:

Yo como usuario con privilegios de administrar tipo de eventos, quiero tener la capacidad de eliminar tipos de eventos que ya no sean relevantes o usados.

Mockups





Criterios de aceptación.

1. Al hacer clic en la sección “TIPO DE EVENTO” en el menú lateral y seleccionar la opción “Eliminar tipo de evento” se muestra una tabla con los tipos de evento creados previamente.
2. La tabla tiene las columnas “Nombre del tipo de evento”, “Color de referencia”, “Autor”, “Fecha de creación”, “Eventos asociados” y “Acción”.

3. Al hacer clic en el botón “  ” sale un modal con el mensaje de “¿Está seguro de eliminar este tipo de evento?” con los botones de “No” y “Sí”.
4. Al hacer clic en el botón “Sí” se elimina el tipo de evento.
5. Al hacer clic en el botón“Sí” se cierra el modal.
6. Al eliminarse un tipo de evento, se muestra una alerta temporal con el mensaje de “Eliminado exitosamente”.
7. Al hacer clic en el botón “No” no hay cambios.
8. Al hacer clic en el botón “No” se cierra el modal.
9. Cuando se elimina un tipo de evento, después de un breve periodo de tiempo este deja de ser visible en la lista de tipos de eventos.

7.1.4.2. Código fuente de eliminar tipo de evento

Controlador de eliminar tipo de evento

Se implementa la eliminación de un tipo de evento en una aplicación Laravel. Recibe el identificador ($id) y busca el registro correspondiente. Si existe, se elimina y retorna un mensaje JSON indicando éxito. En caso de eventos asociados, informa sobre la restricción de eliminación. El bloque try-catch maneja excepciones, retornando mensajes claros en caso de error. La función asegura una gestión eficiente y detallada de la eliminación de tipos de eventos.

Ruta del archivo: app/Http/Controllers/TipoEventoController.php

```php
/**
 * Elimina un tipo de evento específico.
 *
 * @return \Illuminate\Http\Response
 */
public function destroy($id)
{
    try {
        // Busca el tipo de evento por su identificador único.
        $tipo_evento = TipoEvento::find($id);

        // Elimina el tipo de evento.
        $tipo_evento->delete();

        // Responde con un mensaje de éxito.
        return response()->json(['mensaje' => 'Eliminado exitosamente', 'error' => false]);
    } catch (QueryException $e) {
        // Maneja excepciones generadas por consultas a la base de datos.
        if ($e->errorInfo[1] == 1451) {
            // Si hay eventos asociados al tipo de evento, devuelve un mensaje de error específico.
            return response()->json(['mensaje' => 'El tipo de evento tiene eventos asociados', 'error' =>
            true]);
        }

        // Si se produce otro tipo de error, devuelve el mensaje de la excepción.
        return response()->json(['mensaje' => $e->getMessage(), 'error' => true]);
    }
}
```

Código JavaScript de eliminar tipo de evento

Gestiona la visualización y eliminación de tipos de eventos en una interfaz web. Utiliza DataTables para presentar los datos en una tabla paginada y personalizada. La función

initDataTable inicializa la tabla con opciones específicas, asegurando su correcto funcionamiento y destruyendo la instancia anterior si existe.

La eliminación de tipos de evento se realiza mediante la función eliminarTipoEvento, que utiliza Axios para enviar una solicitud de eliminación al servidor. En caso de éxito, se cierra el modal correspondiente y se muestra una alerta de éxito. En caso de error, se muestra un mensaje de error en otro modal.

La recarga de eventos (recargarEventos) se utiliza para actualizar la página después de una eliminación exitosa, proporcionando una experiencia de usuario fluida. En general, este código facilita la interacción del usuario con los tipos de eventos, permitiendo la eliminación con confirmación y actualización automática de la interfaz.

Ruta del archivo: public/js/TipoDeEvento/eliminarTipovento.js

```
let tablaDeTipos;
let tablaInicializada = false;
let formulario = document.getElementById("formularioTipoEvento");

const dataTableOptions = {
    pageLength: 10,
    lengthMenu: [5, 10, 15, 20],
    destroy: true,
    order: [[3, "desc"]],
    language: {
        lengthMenu: "Mostrar _MENU_ entradas",
        zeroRecords: "Ningún tipo de evento encontrado",
        info: "Mostrando de _START_ a _END_ de un total de _TOTAL_ registros",
        infoEmpty: "Ningún tipo de evento encontrado",
        infoFiltered: "(filtrados desde _MAX_ registros totales)",
        search: "Buscar:",
        loadingRecords: "Cargando...",
        paginate: {
            first: "Primero",
            last: "Último",
            next: "Siguiente",
            previous: "Anterior",
        },
    },
};

const initDataTable = async () => {
    if (tablaInicializada) {
        tablaDeTipos.destroy();
    }
    DataTable.datetime("DD-MM-YYYY");
    tablaDeTipos = $("#tablaTipoDeEvento").DataTable(dataTableOptions);
    tablaInicializada = true;
};
```

```
//Borrar tipo de evento
async function eliminarTipoEvento(id) {
    // Aquí utilizamos Axios para enviar una solicitud de eliminación al servidor
    axios.delete(`/api/tipo-evento/${id}`)
        .then(response => {
            if (!response.data.error) {
                // Si la eliminación es exitosa, mostramos un mensaje de éxito en el modal
                $('#modalEliminarTipoEvento' + id).modal('hide'); // Cerrar el modal
                mostrarAlerta(
                    "Éxito",
                    response.data.mensaje,
                    response.data.error ? "danger" : "success"
                );
                recargarEventos();

            } else {
                // Si hay un error, mostramos un mensaje de error en el modal
                $('#modalEliminarTipoEvento' + id).modal('hide'); // Cerrar el modal
                $('#modalMensaje').text('Error: ' + response.data.mensaje);
                $('#modalError').modal('show');
            }
        })
        .catch(error => {
            console.error(error);
        });
}

window.addEventListener("load", async () => {
    initDataTable();
});

/**Para recargar eventos, si o si debemos llamar a la pagina**/
const recargarEventos = () => {
    setTimeout(() => {
        window.location.href = "/admin/tipos-de-evento/eliminar-tipo";
    }, 1700);
}
```

Vista de eliminar tipo de evento

La interfaz muestra una tabla con información detallada sobre los tipos de evento, incluyendo nombre, color, autor, fecha de creación, cantidad de eventos asociados y opciones de acción. Cada fila de la tabla contiene un botón de eliminación, que activa un modal para confirmar la eliminación. Además, se emplean componentes modales para gestionar mensajes de éxito y error. Se utiliza DataTables para mejorar la presentación y funcionalidad de la tabla, y se carga con enlaces a bibliotecas externas.

Ruta del archivo: resources/views/tipos-de-evento/eliminarTiposDeEvento.blade.php

```
@extends('layouts.app')

@section('content')
    <div class="container">
        <div class="row align-items-end">
            <div class="col-md-6">
                <h2>Eliminar tipo de evento</h2>
            </div>
            <div class="col-md-1">
                <x-ModalCrearTipoEvento />
            </div>
            <div class="col-md-9">

            </div>
        </div>
        <div class="row mt-3">
            <!-- Tabla para mostrar tipos de evento -->
            <table class="table table-striped text-secondary" id="tablaTipoDeEvento">
                <caption>Tipos de evento</caption>
                <thead>
                    <tr>
                        <th scope="col" class="col-md-2">Nombre del tipo de evento</th>
                        <th scope="col" class="col-md-2 text-center">Color de referencia</th>
                        <th scope="col" class="col-md-2 text-center">Autor</th>
                        <th scope="col" class="col-md-2 text-center">Fecha de creación</th>
                        <th scope="col" class="col-md-3 text-center">Eventos asociados</th>
                        <th scope="col" class="col-md-3 text-center">Acción</th>
                    </tr>
                </thead>
                <tbody id="datosTabla">
                    @foreach ($tiposDeEventos as $tipoDeEvento)
                        <tr>
                            <td>{{ $tipoDeEvento->nombre }}</td>
                            <td class="container-color">
                                <div class="color-cell" style="background-color:{{ $tipoDeEvento->color }};"></div>
                            </td>
                            <td class="text-center">Yo</td>
                            <td class="text-center">{{ date('d-m-Y', strtotime($tipoDeEvento->created_at)) }}</td>
                            <td class="text-center">{{ $tipoDeEvento->eventos->count() }}</td>
                            <td class="text-center">
                                <!-- Botón para eliminar tipo de evento -->
                                <button type="button" class="btn btn-danger btn-sm" data-bs-toggle="modal"
                                    data-bs-target="#modalEliminarTipoEvento{{ $tipoDeEvento->id }}">
                                    <i class="bi bi-trash"></i>
                                </button>
                                <!-- Modal para confirmar eliminación -->
                                @component('components.modal')
                                    @slot('modalId', 'modalEliminarTipoEvento' . $tipoDeEvento->id)
                                    @slot('modalTitle', 'Eliminar tipo de evento')
                                    @slot('modalContent')
                                        ¿Está seguro de eliminar este tipo de evento?
                                    @endslot
                                    @slot('modalButton')
                                        <!-- Botones para confirmar o cancelar eliminación -->
                                        <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">No</button>
                                        <button type="button" class="btn btn-danger"
                                            onclick="eliminarTipoEvento({{ $tipoDeEvento->id }})">Sí</button>
                                    @endslot
                                @endcomponent
```

```
                    <!-- Modal para mensaje de éxito -->
                    @component('components.modal')
                        @slot('modalId', 'modalExito')
                        @slot('modalTitle', 'Éxito')
                        @slot('modalContent')
                            <div id="modalMensajeExito">
                                Eliminado satisfactoriamente!
                            </div>
                        @endslot
                        @slot('modalButton')
                            <button type="button" class="btn btn-primary" data-bs-dismiss="modal">Aceptar</button>
                        @endslot
                    @endcomponent
                    <!-- Modal para mensaje de error -->
                    @component('components.modal')
                        @slot('modalId', 'modalError')
                        @slot('modalTitle', 'Error')
                        @slot('modalContent')
                            <div id="modalMensajeError">El tipo de evento que quieres eliminar tiene
                                eventos asociados a él!
                            </div>
                        @endslot
                        @slot('modalButton')
                            <button type="button" class="btn btn-primary" data-bs-dismiss="modal">Aceptar</button>
                        @endslot
                    @endcomponent
                </td>
            </tr>
        @endforeach
    </tbody>
</table>
</div>
</div>

<!-- Enlaces y scripts externos -->
<link href="https://cdn.datatables.net/1.13.6/css/dataTables.bootstrap5.min.css" rel="stylesheet">
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.11.1/font/bootstrap-icons.css">
<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.7.1/jquery.min.js"></script>
<script src="https://cdn.datatables.net/1.13.6/js/jquery.dataTables.min.js"></script>
<script src="https://cdn.datatables.net/1.13.6/js/dataTables.bootstrap5.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/moment.js/2.29.2/moment.min.js"></script>
<script src="{{ asset('js/TipoDeEvento/eliminarTipoEvento.js') }}" defer></script>
@endsection
```

7.1.4.3. Interfaz de usuario de eliminar tipo de evento

La interfaz de usuario para la eliminación de un tipo de evento presenta un diseño claro y eficiente para realizar esta acción de manera consciente. Aquí se detallan los elementos clave de la interfaz:

Tabla de Tipos de Evento:

La tabla proporciona un resumen de los tipos de eventos existentes con las siguientes columnas:

Nombre del Tipo de Evento: Identifica el tipo de evento.

Color de Referencia: Muestra el color asociado al tipo de evento para facilitar la identificación visual.

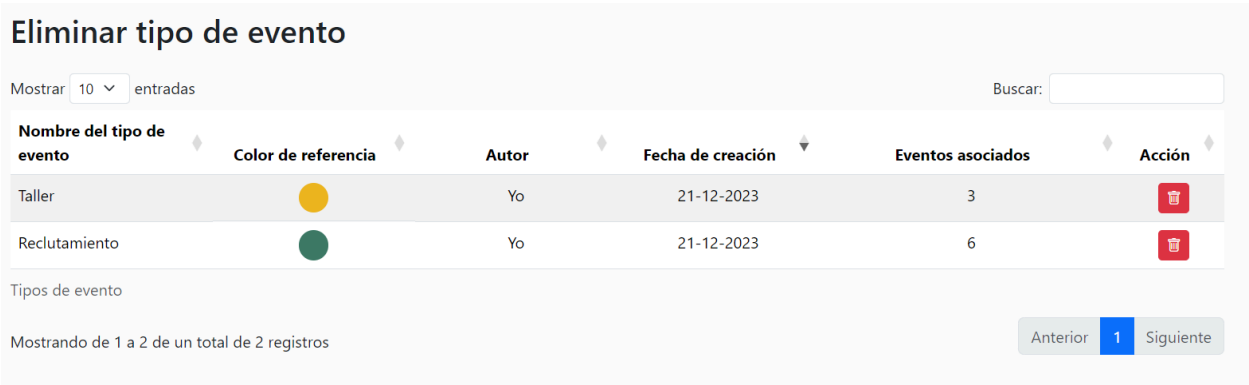
Autor: Indica el creador del tipo de evento.

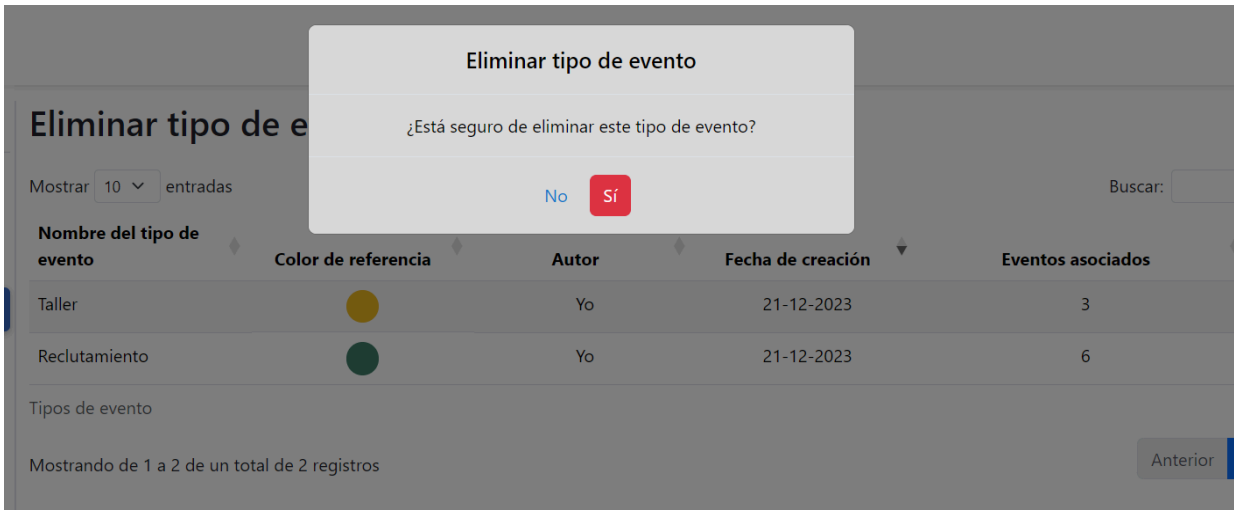
Fecha de Creación: Muestra la fecha en que se creó el tipo de evento.

Eventos Asociados: Indica la cantidad de eventos vinculados al tipo.

Acción (Eliminar): Contiene un botón con un ícono de papelera que activa un modal de confirmación.

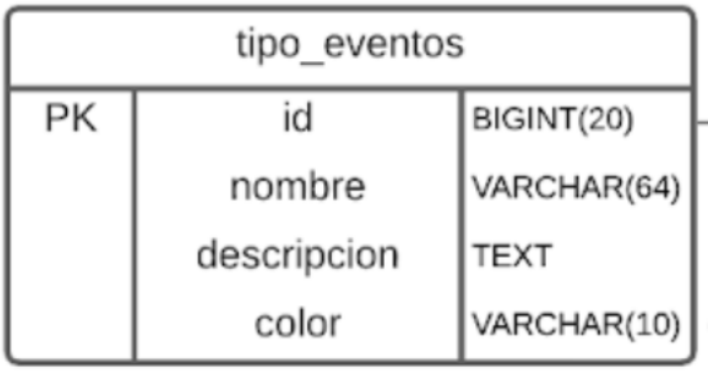
Modal de Confirmación:
El botón de acción en la columna "Acción" desencadena un modal con el mensaje "¿Está seguro de eliminar este tipo de evento?".
El modal ofrece dos opciones: "No" para cancelar la acción y "Sí" para confirmar la eliminación del tipo de evento.
Interacción del Usuario:
El ícono de papelera en la columna "Acción" permite al usuario iniciar el proceso de eliminación con un solo clic.
El modal de confirmación asegura que el usuario confirme su intención de eliminar el tipo de evento, evitando eliminaciones accidentales.





7.1.5. Tabla de base de datos de tipo de evento

La tabla 'tipo_eventos' contiene información esencial sobre los eventos, con campos como 'id' para identificación única, 'nombre' para descripción concisa, 'descripcion' para detalles adicionales, y 'color' para una identificación visual rápida.

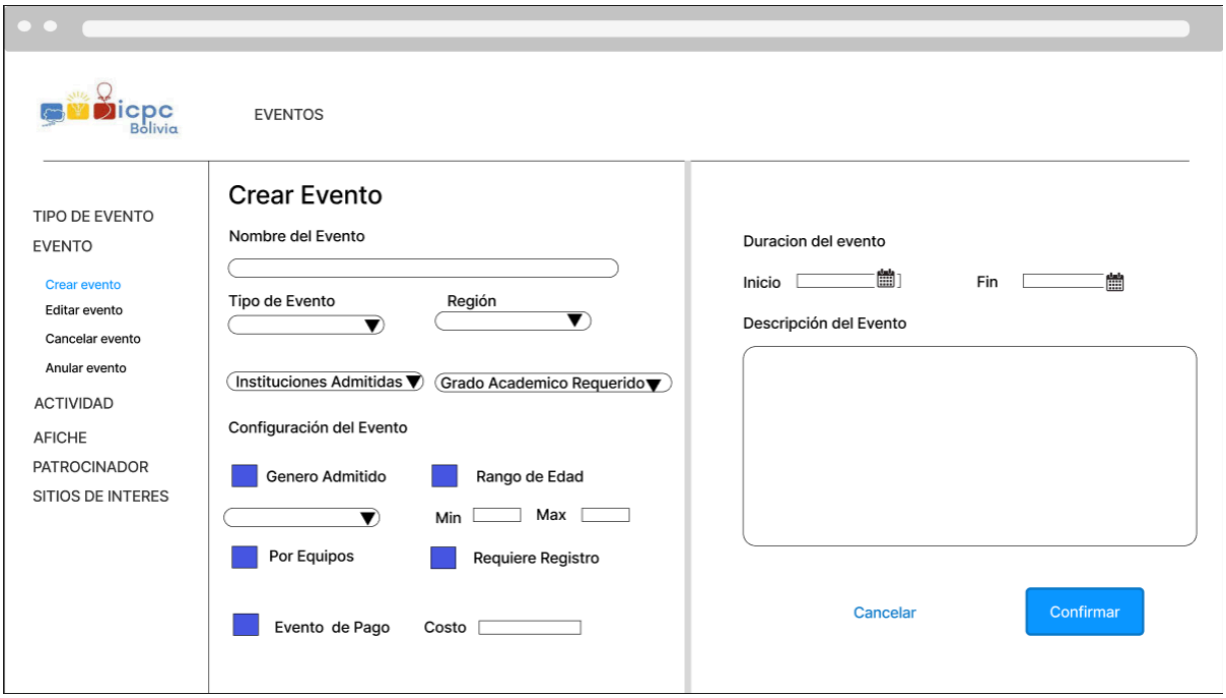


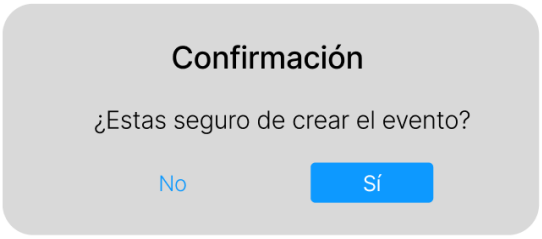
7.2. EVENTO

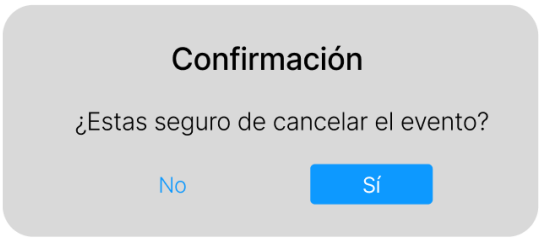
7.2.1. Crear evento

7.2.1.1. Historia de usuario de crear evento

Historia de Usuario	
Título: Crear un evento con información básica	**ID: 5**
Estimación: 5	**Importancia: Alta**
Descripción Yo como usuario con privilegios para crear evento, quiero crear un evento con la siguiente información: nombre del evento, tipo de evento, región, instituciones admitidas ,grado académico requerido, opciones de configuración (género admitido, rango de edad, evento por equipos, requerimiento de registro, evento de pago), duración del evento (fecha inicio, fecha fin) y descripción del evento para que el usuario pueda organizar de forma clara el evento.	
MockUps	







Criterios de aceptación

1. Al hacer clic en la sección “EVENTO” en el menú lateral y seleccionar la opción “Crear evento” se muestra un formulario con los siguientes campos: “Nombre del evento *”, “Tipo de evento”, “Región”, “Instituciones admitidas”, “Grado académico requerido”, Configuración del evento (“Género admitido”, “Rango de edad”, “Por equipos”, “Requiere talla de polera”, “Evento de pago”); Duración del evento * (“Inicio”, “Fin”), “Descripción del evento”; los botones “Cancelar” y “Confirmar”.
2. El campo “Nombre del evento *” es obligatorio.
3. El campo “Nombre del evento *” es único.
4. El campo “Nombre del evento *” tiene como máximo 64 caracteres.
5. El campo “Tipo de evento” despliega los tipos de eventos ya registrados.
6. El campo “Tipo de evento” tiene por defecto el primer tipo de evento registrado.
7. El campo “Descripción del evento” tiene como máximo de 2048 caracteres.
8. El apartado de “Duración del evento *” tiene 2 campos para fechas ("Inicio " y "Fin ").
9. El apartado de “Duración de evento *” despliega calendarios para registrar las fechas.
10. Los campos que son fechas tienen el formato DD-MM-AAAA y hh:mm.

11. Si la fecha del campo “Inicio” es menor a la fecha actual, se muestra el mensaje de “Seleccione una fecha correcta.” debajo del campo “Inicio”.
12. El campo “Fin” comienza inhabilitado.
13. Si la fecha del campo “Inicio” es mayor a la fecha actual, el campo de la fecha “Fin” se habilita.
14. Si la fecha del campo “Inicio” tiene un valor menor a la fecha actual se restablece el campo de la fecha “Fin” y se deshabilita.
15. Si la fecha del campo “Fin” es menor al campo de la fecha “Inicio”, se muestra el mensaje de “Seleccione una fecha correcta.” debajo del campo “Fin”.
16. Los campos de “Duración de evento” son obligatorios.
17. El campo “Rango de edad” es una casilla de verificación.
18. Al marcar la casilla “Rango de edad“ se muestran dos campos con la edad mínima (Min) y la edad máxima (Max).
19. Al menos uno de los campos “Min” o “Max” requiere un valor.
20. El valor mínimo para el campo “Min” es 10.
21. El valor mínimo para el campo “Max” es 10.
22. El valor máximo para el campo “Min” es 99.
23. El valor máximo para el campo “Max” es 99.
24. Si el valor del campo “Min” o “Max” es menor al valor mínimo establecido, aparece un mensaje de validación “Rango de edades no válido”.
25. Si el valor del campo “Min” o “Max” es mayor al valor máximo establecido, aparece un mensaje de validación “Rango de edades no válido”.
26. Si el valor del campo “Min” es mayor que el valor del campo “Max” aparece un mensaje de validación “Rango de edades no válido”.
27. Si solo el campo “Min” tiene un valor, aparece un mensaje de validación “Edad válida desde los (valor del campo “Min”) años”.
28. Si solo el campo “Max” tiene un valor, aparece un mensaje de validación “Edad válida hasta los (valor del campo “Max”) años”.
29. El campo “Género admitido” es una casilla de verificación.
30. Al marcar la casilla “Género admitido” se muestra un selector.
31. El selector de “Género admitido” muestra dos opciones “Femenino” y “Masculino”.
32. El selector de “Género admitido” está por defecto con la opción “Femenino”.
33. El campo “Evento de pago” es una casilla de verificación.
34. Al marcar la casilla “Evento de pago” se muestra un campo “Costo”.
35. El valor mínimo del campo “Costo” es 1.
36. El valor máximo del campo “Costo” es 5000.
37. Si se ingresa un valor superior al valor máximo en el campo “Costo” aparece un mensaje de validación “Monto máximo 5000 Bs.”.
38. Si se ingresa un valor inferior al valor mínimo en el campo “Costo” aparece un mensaje de validación “Monto mínimo 1 Bs.”.
39. El campo “Costo” solo admite un decimal.
40. El campo “Por equipos” es una casilla de verificación.

41. Al marcar la casilla “Por equipos“ se muestran dos campos, “Min” y “Max”, para la cantidad de participantes admitidos por equipo.
42. El campo “Max” requiere un valor.
43. El campo “Min” puede estar vacío.
44. El valor mínimo para el campo “Min” es 2.
45. El valor del campo “Max” no puede ser menor al valor del campo “Min” .
46. El valor máximo para el campo “Min” es 50.
47. El valor máximo para el campo “Max” es 50
48. El campo “Requiere talla de polera” es una casilla de verificación.
49. El campo “Grado académico requerido” es un menú desplegable.
50. El menú desplegable de “Grado académico requerido” tiene las opciones de “Todas”, “Primaria”, “Secundaria”, “Universidad”,”Licenciatura”,”Maestría” y ”Doctorado” los cuales son casillas de selección.
51. La casilla de selección “Todas” marca y desmarca a todas las casillas de selección del menú desplegable de “Grado académico requerido”.
52. El campo “Instituciones admitidas” es un menú desplegable.
53. El menú desplegable de “Instituciones admitidas” tiene los valores de “TODAS”, “UMSS”, “UMSA”, ”UPSA”, ”UCB”, “UPB” y ”UNIFRANZ” los cuales son casillas de selección.
54. La casilla de selección “TODAS” marca y desmarca a todas las casillas de selección del menú desplegable de “Instituciones admitidas”.
55. Al hacer clic en el botón “Cancelar” aparece un modal de confirmación con el mensaje de ¿Está seguro de cancelar la creación del evento?, con los botones de “No” y “Sí” .
56. Al hacer clic en el botón “Sí” se limpian los campos.
57. Al hacer clic en el botón “Sí” se cierra el modal.
58. Al hacer clic en el botón “No” no se realiza ninguna acción.
59. Al hacer clic en el botón “No” se cierra el modal.
60. Al hacer clic en el botón “Confirmar” aparece un modal de confirmación con el mensaje de “¿Está seguro de crear el evento?”, con los botones de “No” y “Sí”.
61. Si el campo “Nombre del evento *” ya fue registrado, al hacer clic en el botón “Sí” se muestra una alerta temporal con el mensaje “El evento ya existe”.
62. Si el campo “Nombre del evento *” ya fue registrado, al hacer clic en el botón “Sí” se muestra un mensaje de validación “El evento ya existe.” debajo del campo “Nombre del evento *”.
63. Si el campo “Nombre del evento *” está vacío, al hacer clic en el botón “Sí” se muestra un mensaje de validación “El nombre no puede estar vacío” debajo del campo “Nombre del evento *”.
64. Si alguna fecha del evento “Inicio” o “Fin” está vacía, al hacer clic en el botón “Si” se muestra un mensaje de validación “Seleccione una fecha y hora.” debajo de la fecha correspondiente.
65. Si los campos están correctamente llenados, al hacer clic en el botón “Sí” se muestra una alerta temporal con el mensaje “Creado exitosamente”.
66. Al finalizar el tiempo de la alerta temporal, se redirige a la vista de Eventos.

7.2.1.2. Código fuente de crear evento

Controlador de crear evento

La función store en el controlador EventoController se encarga de almacenar un nuevo evento en la base de datos. Recibe los datos del evento a través de una solicitud HTTP (Request $request). Dentro de la función, se instancia un nuevo objeto de la clase Evento, y se asignan los valores de los campos del evento utilizando los datos proporcionados en la solicitud. Luego, se intenta guardar el evento en la base de datos. Si la operación es exitosa, la función devuelve una respuesta JSON indicando que el evento se creó correctamente. En caso de que ocurra una excepción durante el proceso de almacenamiento, se maneja con un bloque try-catch, y se devuelve una respuesta JSON indicando un mensaje de error.

Ruta del archivo: app/Http/Controllers/EventoController.php

```php
/**
 * Almacena un nuevo evento en la base de datos.
 */
public function store(Request $request)
{
    try {
        $evento = new Evento();
        $evento->nombre             = $request->nombre;
        $evento->descripcion        = $request->descripcion;
        $evento->inicio_evento      = $request->inicio_evento;
        $evento->fin_evento         = $request->fin_evento;
        $evento->institucion        = $request->institucion;
        $evento->region             = $request->region;
        $evento->grado_academico    = $request->grado_academico;
        $evento->equipo_minimo      = $request->equipo_minimo;
        $evento->equipo_maximo      = $request->equipo_maximo;
        $evento->talla              = $request->talla;
        $evento->edad_minima        = $request->edad_minima;
        $evento->edad_maxima        = $request->edad_maxima;
        $evento->genero             = $request->genero;
        $evento->precio_inscripcion = $request->precio_inscripcion;
        $evento->id_tipo_evento     = $request->id_tipo_evento;
        $evento->save();
        return response()->json(['mensaje' => 'Creado exitosamente', 'error' => false]);
    } catch (QueryException $e) {
        // Manejo de errores durante el proceso de creación del evento.
        // Puedes personalizar la gestión de errores según tus necesidades.
        return response()->json(['mensaje' => 'Error al crear el evento', 'error' => true]);
    }
}
```

Código JavaScript de crear evento

Comienza definiendo variables que representan elementos del Document Object Model (DOM) para acceder y manipular elementos HTML. Estos elementos incluyen casillas de verificación, campos de entrada de texto, fechas y otros controles de formulario. El código presenta funciones que realizan acciones específicas, como mostrar u ocultar elementos del formulario según las selecciones del usuario, preparar los datos del formulario para su envío, y realizar operaciones de creación y edición de eventos mediante solicitudes a una API utilizando la biblioteca Axios. Se emplea un enfoque

modular al dividir la lógica en funciones específicas, como la validación del formulario, la manipulación de fechas y la gestión de eventos. El código también incluye manejo de eventos, como cambios en las casillas de verificación y campos de entrada, así como el uso de ventanas modales para confirmar acciones antes de realizarlas.

Ruta del archivo: public/js/Evento/crearEvento.js

```
// Declaración de variables que representan elementos del DOM
const inputGenero = document.getElementById("generoCheck");
const inputEdad = document.getElementById("edadCheck");
const inputCosto = document.getElementById("eventoPagoCheck");
const form = document.getElementById("formularioCrearEvento");
const fechaInicio = document.getElementById("fechaInicio");
const fechaFin = document.getElementById("fechaFin");
const edadMinima = document.getElementById("edadMinima");
const edadMaxima = document.getElementById("edadMaxima");
const chekcTodas = document.getElementById("check-institucion-TODAS");
const costo = document.getElementById("costoEvento");
const checkTodasRango = document.getElementById("input-grado-Todas");
const nombreEvento = document.getElementById("nombreDelEvento");
const mensajeNombre = document.getElementById("mensajeNombre");
const checkEquipo = document.getElementById("equipoCheck");
const checkNotificacion = document.getElementById("notificacion");
const equipoMaximo = document.getElementById("equipoMaximo");
const equipoMinimo = document.getElementById("equipoMinimo");

let boolCheckEquipo = true;
let boolMinEquipo = true;
let boolMaxEquipo = true;
let boolFecha = true;
let boolCosto = true;
let boolMinEdad = true;
let boolcheckEdad = true;
let boolMaxEdad = true;
let crear = true;
let datosActualizados = false;
let nombreAnterior = '';
let mensajeRepetido = '';

let fechaLocal = new Date();
fechaLocal.setHours(fechaLocal.getHours() - 4);
let laFecha = fechaLocal.toISOString().substring(0, 16);
```

```
// Función que muestra u oculta un elemento del formulario según el estado de una casilla de verificación
const mostrarInput = (indInput, check) => {
    let input = document.getElementById(indInput);
    if (!check) {
        input.style.display = "none";
    } else {
        input.style.display = "flex";
    }
};

// Función que prepara los datos del formulario para ser enviados mediante una solicitud
const prepararFormData = () => {
    if (!crear) {
        document.querySelectorAll(".fecha-editar").forEach(Element => {
            Element.disabled = false;
        });
    }
    const formData = new FormData(form);
    if (!inputEdad.checked) {
        formData.set("edad_minima", "");
        formData.set("edad_maxima", "");
    }
    if (!inputGenero.checked) {
        formData.set("genero", "");
    }
    if (!inputCosto.checked) {
        formData.set("precio_inscripcion", "");
    }
    if (!checkEquipo.checked) {
        formData.set("equipo_minimo", "");
        formData.set("equipo_maximo", "");
    }
    if (!document.getElementById("tallaCheck").checked) {
        formData.set("talla", "");
    }
    if (!crear) {
        if (!checkNotificacion.checked) {
            formData.set("notificacion", "");
        } else {
            formData.set("notificacion", "on");
```

```
        }
    }

    return formData;
};

// Función que realiza la edición de un evento mediante una solicitud a la API
const editarEvento = (formData) => {
    if (!datosActualizados) {
        window.location.href =
            "/eventos/" + document.getElementById("nombreDelEvento").value;
    }
    nombreAnterior = nombreEvento.value;
    axios
        .post("/api/evento/actualizar/" + formData.get("evento_id"), formData)
        .then(function (response) {
            mostrarAlerta(
                "Éxito",
                response.data.mensaje,
                response.data.error ? "danger" : "success"
            );
            mensajeRepetido = response.data.mensaje;
            if (mensajeRepetido !== 'El evento ya existe') {
                setTimeout(() => {
                    window.location.href = "/editarEvento";
                }, 1800);
                form.reset();
            } else {
                nombreEvento.classList.remove("is-valid");
                nombreEvento.classList.add("is-invalid");
                mensajeNombre.textContent = 'El evento ya existe.';
            }
        })
        .catch(function (error) {
            mostrarAlerta(
                "Error",
                "Hubo un error al guardar el tipo de evento",
                "danger"
            );
```

```
        });
};

// Función que realiza la creación de un evento mediante una solicitud a la API
const crearEvento = (formData) => {
    nombreAnterior = nombreEvento.value;
    axios
        .post("/api/evento", formData)
        .then(function (response) {
            mostrarAlerta(
                "Éxito",
                response.data.mensaje,
                response.data.error ? "danger" : "success"
            );
            mensajeRepetido = response.data.mensaje;
            if (mensajeRepetido === 'El evento ya existe') {
                nombreEvento.classList.remove("is-valid");
                nombreEvento.classList.add("is-invalid");
                mensajeNombre.textContent = 'El evento ya existe.';
            } else {
                setTimeout(() => {
                    window.location.href = "/eventos/";
                }, 1800);
            }
        })
        .catch(function (error) {
            mostrarAlerta(
                "Error",
                "Hubo un error al guardar el tipo de evento",
                "danger"
            );
        });
};

// Función que se ejecuta cuando se detecta un cambio en los datos del formulario
const datoCambiado = () => {
    if (!crear) {
        datosActualizados = true;
    }
```

```
};

// Inicialización de ciertas acciones cuando la ventana ha cargado completamente
window.addEventListener("load", () => {
    checkTodasRango.classList.remove("grado-requerido");
    chekcTodas.classList.remove("institucion");
    if (document.getElementById("nombreDelEvento").value != "") {
        crear = false;
        iniciarEditar();
    } else {
        fechasMin();
        fechaFin.disabled = true;
    }
    axios
        .get("/api/tipo-evento")
        .then(function (response) {
            const idTipoEvento = document.getElementById("tipoDelEvento").getAttribute("data-id");
            const select = document.getElementById("tipoDelEvento");
            const tiposDeEvento = response.data;
            tiposDeEvento.forEach(function (tipo) {
                if (idTipoEvento != tipo.id) {
                    const option = document.createElement("option");
                    option.value = tipo.id;
                    option.text = tipo.nombre;
                    select.appendChild(option);
                }
            });
            if (idTipoEvento != "") {
                select.value = idTipoEvento;
            }
        })
        .catch(function (error) {
            console.error(error);
        });
});

// Función que se ejecuta al cerrar el formulario de edición o creación de eventos
const cerrar = (edicion) => {
    if (crear) {
```

```
            location.reload();
            form.reset();
        } else {
            window.location.href = "/editarEvento/";
        }
};

// Event listener para el envío del formulario
form.addEventListener("submit", (event) => {
    event.preventDefault();
    form.querySelectorAll(".form-control, .form-select").forEach((Element) => {
        Element.dispatchEvent(new Event("change"));
    });
    if (!validar()) {
        $("#modalConfirmacion").modal("hide");
    } else {
        let aux = "";
        form.querySelectorAll(".institucion").forEach((Element) => {
            if (Element.checked) {
                aux = Element.value + "-" + aux;
            }
        });
        let insti = "";
        form.querySelectorAll(".grado-requerido").forEach((Element) => {
            if (Element.checked) {
                insti = Element.value + "-" + insti;
            }
        });
        const formData = prepararFormData();
        formData.set("institucion", aux.slice(0, -1));
        formData.set("grado_academico", insti.slice(0, -1));
        if (!crear) {
            editarEvento(formData);
        } else {
            crearEvento(formData);
        }
        $("#modalConfirmacion").modal("hide");
    }
});
```

```
// Función de validación del formulario
const validar = () => {
    if (form.querySelector(".is-invalid") === null) {
        return true;
    } else {
        return false;
    }
};

// Event listeners para cambios en las casillas de verificación y campos del formulario
inputGenero.addEventListener("change", () => {
    mostrarInput("genero", inputGenero.checked);
});
inputEdad.addEventListener("change", () => {
    mostrarInput("rangosDeEdad", inputEdad.checked);
});
inputCosto.addEventListener("change", () => {
    mostrarInput("eventoPago", inputCosto.checked);
});
checkEquipo.addEventListener("change", () => {
    mostrarInput("numero-integrantes", checkEquipo.checked);
});

nombreEvento.addEventListener("input", validarNombreRepetido);
nombreEvento.addEventListener("change", validarNombreRepetido);

// Función que se ejecuta al iniciar la edición de un evento existente
const iniciarEditar = () => {
    mostrarInput("genero", inputGenero.checked);
    mostrarInput("rangosDeEdad", inputEdad.checked);
    mostrarInput("eventoPago", inputCosto.checked);
    mostrarInput("numero-integrantes", checkEquipo.checked);
    let f1 = new Date(fechaInicio.value);
    let fl1 = new Date(laFecha);
    let boolGrado = true;
    let boolInstitucion = true;
    const instituciones = document.getElementById("ul-institucion").getAttribute("data-institucion");
    document.querySelectorAll(".institucion").forEach(Element => {
        if (instituciones.includes(Element.value)) {
```

```
                Element.checked = true;
                Element.classList.add("fecha-editar");
            } else {
                boolInstitucion = false;
            }
        });
        const grados = document.getElementById("ul-grado").getAttribute("data-grado");
        document.querySelectorAll(".grado-requerido").forEach(Element => {
            if (grados.includes(Element.value)) {
                Element.checked = true;
                Element.classList.add("fecha-editar");
            } else {
                boolGrado = false;
            }

        });
        checkTodasRango.checked = boolGrado;
        chekcTodas.checked = boolInstitucion;

        if (f1 <= fl1) {
            checkTodasRango.disabled = boolGrado;
            chekcTodas.disabled = boolInstitucion;

            if (!inputGenero.checked) {
                inputGenero.classList.add("fecha-editar");
            }
            document.getElementById("ul-institucion").classList.add("fecha-editar");
            document.getElementById("todas-grado").classList.add("fecha-editar");

            document.querySelectorAll(".fecha-editar").forEach(Element => {
                Element.disabled = true;
            });
            fechaFin.min = laFecha;
        } else {
            fechasMin();
        }
    }
```

```
// Función que establece fechas mínimas para los campos de fecha
const fechasMin = () => {
    let fechaLocal = new Date();
    fechaLocal.setHours(fechaLocal.getHours() - 4);
    let laFecha = fechaLocal.toISOString().substring(0, 16);
    fechaInicio.min = laFecha;
    fechaFin.min = laFecha;
};

// Función que valida si el nombre del evento es único
function validarNombreRepetido() {
    if (nombreAnterior === '') {
        if (nombreEvento.value === '') {
            nombreEvento.classList.remove("is-valid");
            nombreEvento.classList.add("is-invalid");
            mensajeNombre.textContent = "El nombre no puede estar vacío.";
        } else {
            nombreEvento.classList.remove("is-invalid");
            nombreEvento.classList.add("is-valid");
        }
    } else {
        if (nombreEvento.value !== nombreAnterior && nombreEvento.value !== '') {
            nombreEvento.classList.remove("is-invalid");
            nombreEvento.classList.add("is-valid");
        } else if (nombreEvento.value == '') {
            nombreEvento.classList.remove("is-valid");
            nombreEvento.classList.add("is-invalid");
            mensajeNombre.textContent = 'El nombre no puede estar vacío.';
        } else {
            nombreEvento.classList.remove("is-valid");
            nombreEvento.classList.add("is-invalid");
            mensajeNombre.textContent = 'La actividad ya existe';
        }
    }
}
```

Vista de crear evento

La vista comienza extendiendo la plantilla 'layouts.app' y define la sección 'content'. Dentro de esta sección, hay un formulario que utiliza Bootstrap para el diseño y validación. Se incluye un campo de entrada oculto para el identificador del evento, que se establece en caso de edición. La sección de encabezado del formulario muestra un título dinámico según si la ruta actual es para editar o crear un evento. Luego, hay dos columnas, la primera de las cuales contiene información básica del evento, como el nombre, tipo, región, instituciones admitidas y grado académico requerido.

La segunda columna aborda la configuración específica del evento, como género admitido, rango de edad, participación por equipos, evento de pago y requisitos adicionales como la talla de la polera. También incluye campos relacionados con la duración del evento, fecha de inicio y fin, y una sección para la descripción del evento.

Además, se añade un componente de modal para la confirmación antes de cancelar o confirmar la acción (crear). También hay scripts al final que incluyen archivos JS para manejar la lógica relacionada con la creación y validación de eventos.

Ruta del archivo: resources/views/crear-evento/crearEvento.blade.php

```
@extends('layouts.app')
@section('content')
    {{-- @php
        dd($datos)
    @endphp --}}
    <div class="container">
        <!-- Formulario para crear/editar eventos -->
        <form class="row g-4 needs-validation mt-1" method="POST" novalidate id="formularioCrearEvento">
            @csrf
            <!-- Campo oculto para almacenar el ID del evento si existe -->
            <input type="hidden" name="evento_id" value="{{ isset($datos['evento_id']) ? $datos['evento_id'] : '' }}">

            <!-- Sección para el título del evento -->
            <div class="col-md-12">
                @if (Route::currentRouteName() == 'evento.editar')
                    <h2 id="titulo">Editar evento</h2>
                @else
                    <h2 id="titulo">Crear evento</h2>
                @endif
            </div>

            <!-- Sección para detalles básicos del evento -->
            <div class="col-md-6 border-end">
                <div class="col-md-12">
                    <!-- Campo para ingresar el nombre del evento -->
                    <label for="nombreDelEvento" class="form-label">Nombre del evento *</label>
                    <input name="nombre" type="text" class="form-control fecha-editar" id="nombreDelEvento"
                        onchange="datoCambiado()" placeholder="Ingrese el nombre del evento" maxlength="64"
                        value="{{ isset($datos['nombreDelEvento']) ? $datos['nombreDelEvento'] : '' }}" required>
                    <div id="mensajeNombre" class="invalid-feedback">
                        El nombre no puede estar vacío.
                    </div>
                </div>

                <!-- Sección para seleccionar el tipo de evento -->
                <div class="row mt-4">
                    <div class="col-md-6">
                        <label for="tipoDelEvento" class="form-label">Tipo de evento</label>
                        <!-- Menú desplegable para seleccionar el tipo de evento -->
                        <select name="id_tipo_evento" class="form-select fecha-editar" id="tipoDelEvento"
                            onchange="datoCambiado()" aria-placeholder="Elija un tipo de evento..."
                            data-id="{{ $datos['id_tipo_evento'] }}" required>
                            @if( $datos['nombre_tipo_evento'] != '' )
                                <option value="{{ $datos['id_tipo_evento'] }}" {{$datos['id_tipo_evento'] ? 'selected' : '' }}>
                                    {{ $datos['nombre_tipo_evento'] }}
                                </option>
                            @endif
                        </select>
                        <div class="invalid-feedback">
                            Seleccione un tipo de evento.
                        </div>
                    </div>
                    <div class="col-md-6">
                        <label for="select-region" class="form-label">Región</label>
                        <!-- Menú desplegable para seleccionar la región del evento -->
                        <select class="form-select" name="region" id="select-region" required>
                            @foreach (['Internacional', 'Nacional', 'Departamental'] as $regionDato)
                                <option value="{{ $regionDato }}" @if ($datos['region'] == $regionDato || 'Departamental' == $regionDato) selected @endif>
                                    {{ $regionDato }}
                                </option>
```

```
                @endforeach
            </select>
        </div>
    </div>

    <!-- Sección para seleccionar instituciones admitidas y grado académico requerido -->
    <div class="row mt-4">
        <div class="col-lg-6 col-md-12 mb-3 ">
            <!-- Botón para seleccionar instituciones admitidas -->
            <button id="btnGroupDrop1" type="button"
                class="btn dropdown-toggle w-100 text-start ps-3 pt-2 pb-2 "
                style ="border-color: rgb(207, 207, 207)" data-bs-toggle="dropdown" aria-expanded="false">
                Instituciones admitidas
            </button>
            <!-- Menú desplegable para seleccionar instituciones admitidas -->
            <ul class="dropdown-menu p-3 col-lg-2 col-md-4" id="ul-institucion" aria-labelledby="btnGroupDrop1"
                data-institucion="{{ $datos['institucion'] }}">
                @foreach (['TODAS', 'UMSS', 'UMSA', 'UPSA', 'UCB', 'UPB', 'UNIFRANZ'] as $institucion)
                    <li @if ($institucion == 'Todas') id="todas-institucion" @endif>
                        <input class="form-check-input institucion border-dark" type="checkbox"
                            value="{{ $institucion }}" id="check-institucion-{{ $institucion }}">
                        <label class="form-check-label" for="check-institucion-{{ $institucion }}">
                            {{ $institucion }}
                        </label>
                    </li>
                @endforeach
            </ul>
        </div>
        <div class="col-lg-6 col-md-12">
            <!-- Botón para seleccionar grado académico requerido -->
            <button id="boton-grado" type="button" class="btn dropdown-toggle w-100 text-start ps-3 pt-2 pb-2"
                style ="border-color: rgb(207, 207, 207)" data-bs-toggle="dropdown" aria-expanded="false">
                Grado académico requerido
            </button>
            <!-- Menú desplegable para seleccionar grado académico requerido -->
            <ul class="dropdown-menu p-3 col-lg-2 col-md-4" id="ul-grado" aria-labelledby="boton-grado"
                data-grado="{{ $datos['grado_academico'] }}">
                @foreach (['Todas', 'Primaria', 'Secundaria', 'Universidad', 'Licenciatura', 'Maestria', 'Doctorado'] as $grado)
                    <li @if ($grado == 'Todas') id="todas-grado" @endif>
                        <input class="form-check-input grado-requerido border-dark" type="checkbox"
                            value="{{ $grado }}" id="input-grado-{{ $grado }}">
                        <label class="form-check-label" for="input-grado-{{ $grado }}">
                            {{ $grado }}
                        </label>
                    </li>
                @endforeach
            </ul>
        </div>
    </div>

    <!-- Sección para la configuración del evento -->
    <div class="col-md-12 mt-3">
        Configuración del evento
    </div>

    <div class="row">
        <!-- Sección para opciones de género admitido y rango de edad -->
        <div class="row mt-4">
            <div class="col-md-6">
```

```
                <!-- Checkbox para género admitido -->
                <input name="evento_genero" type="checkbox" class="form-check-input border-dark"
                    id="generoCheck" onchange="datoCambiado()" data-id="{{ $datos['genero'] }}"
                    @if ($datos['genero']) checked @endif>
                <label for="genero" class="form-check-label">Género admitido</label>
                <!-- Menú desplegable para seleccionar género admitido -->
                <select class="form-select fecha-editar" name="genero" id="genero" style="display:none;">
                    @foreach (['Femenino', 'Masculino'] as $sexo)
                        <option value="{{ $sexo }}" @if ($datos['genero'] == $sexo) selected @endif>
                            {{ $sexo }}
                        </option>
                    @endforeach
                </select>
            </div>

            <div class="col-md-6">
                <!-- Checkbox para rango de edad -->
                <input name="rango_edad" type="checkbox" class="form-check-input border-dark fecha-editar"
                    id="edadCheck" onchange="datoCambiado()" data-id="{{ $datos['edad_minima'] }}"
                    @if ($datos['edad_minima']) checked @endif>
                <label for="limiteDeEdad" class="form-check-label">Rango de edad</label>
                <div class="valid-feedback" id="ValidoRangoEdad">
                </div>
                <div id="validationServerUsernameFeedback" class="invalid-feedback">
                    Rango de edades no válido.
                </div>
                <div class="row" id="rangosDeEdad" style="display: none;">
                    <!-- Campo para edad mínima -->
                    <div class="col-md-6">
                        <div class="row " id="rangoMin">
```

```
                            <div class="col-md-3">
                                <label for="edadMinima" class="form-label">Min</label>
                            </div>
                            <div class="col-md-9">
                                <div class="input-group">
                                    <input name="edad_minima" type="number"
                                        class="form-control input-edad fecha-editar entero" min="10"
                                        id="edadMinima" step="1" max="99"
                                        value="{{ isset($datos['edad_minima']) ? $datos['edad_minima'] : '' }}">
                                </div>
                            </div>
                        </div>
                    </div>
                    <!-- Campo para edad máxima -->
                    <div class="col-md-6">
                        <div class="row " id="rangoMax">
                            <div class="col-md-3">
                                <label for="edadMaxima" class="form-label">Max</label>
                            </div>
                            <div class="col-md-9">
                                <div class="input-group">
                                    <input name="edad_maxima" type="number"
                                        class="form-control input-edad fecha-editar entero" id="edadMaxima"
                                        step="1" min="10" max="99"
                                        value="{{ isset($datos['edad_maxima']) ? $datos['edad_maxima'] : '' }}">
                                </div>
                            </div>
                        </div>
                    </div>
                </div>
```

```
        </div>
    </div>

    <!-- Sección para opciones de por equipos y evento de pago -->
    <div class="row mt-4">
        <div class="col-md-6 mt-2">
            <!-- Checkbox para opción "Por equipos" -->
            <input type="checkbox" class="form-check-input border-dark fecha-editar"
                onchange="datoCambiado()" id="equipoCheck"  @if ($datos['equipo_minimo']) checked @endif>
            <label class="form-check-label " for="equipoCheck">Por equipos</label>
            <div class="invalid-feedback" id="ValidarRangoEquipo">
            </div>
        </div>

        <div class="col-md-6 mt-2">
            <!-- Checkbox para opción "Evento de pago" -->
            <input name="evento_pago" type="checkbox" class="form-check-input border-dark fecha-editar"
                onchange="datoCambiado()" id="eventoPagoCheck"
                data-id="{{ $datos['precio_inscripcion'] }}"
                @if ($datos['precio_inscripcion']) checked @endif>
            <label class="form-check-label" for="eventoPagoCheck">Evento de pago</label>
        </div>
    </div>

    <!-- Sección para configuración de número de integrantes y costo del evento -->
    <div class="row mt-4">
        <div class="col-md-6">
            <div class="row" id="numero-integrantes" style="display:none;">
                <!-- Campo para número mínimo y máximo de integrantes por equipo -->
                <div class="col-md-6">
```

```
                    <div class="row " id="rangoEquipoMin">
                        <div class="col-md-3">
                            <label for="equipoMinimo" class="form-label">Min</label>
                        </div>
                        <div class="col-md-9">
                            <div class="input-group">
                                <input name="equipo_minimo" type="number"
                                    class="form-control input-edad fecha-editar entero" min="2"
                                    id="equipoMinimo" step="1" max="50"
                                    value="{{ isset($datos['equipo_minimo']) ? $datos['equipo_minimo'] : '' }}">
                            </div>
                        </div>
                    </div>
                </div>
                <div class="col-md-6">
                    <div class="row " id="rangoEquipoMax">
                        <div class="col-md-3">
                            <label for="equipoMaximo" class="form-label">Max</label>
                        </div>
                        <div class="col-md-9">
                            <div class="input-group">
                                <input name="equipo_maximo" type="number"
                                    class="form-control input-edad fecha-editar entero" id="equipoMaximo"
                                    step="1" min="2" max="50"
                                    value="{{ isset($datos['equipo_maximo']) ? $datos['equipo_maximo'] : '' }}">
                            </div>
                        </div>
                    </div>
                </div>
            </div>
```

```
                    </div>

                    <div class="col-md-6">
                        <div class="row " id="eventoPago" style="display: none;">
                            <!-- Campo para el costo del evento -->
                            <div class="col-md-4">
                                <label class="col-form-label" for="costoEvento">Costo</label>
                            </div>
                            <div class="col-md-8">
                                <div class="input-group mb-3">
                                    <span class="input-group-text">Bs.</span>
                                    <input name="precio_inscripcion" type="number" class="form-control fecha-editar"
                                        min="1" max="5000" id="costoEvento" step="0.5"
                                        onchange="setCostoInvalidoFeedback()"
                                        value="{{ isset($datos['precio_inscripcion']) ? $datos['precio_inscripcion'] : '0.0' }}">
                                    <div id="costoInvalido" class="invalid-feedback"></div>
                                </div>
                            </div>
                        </div>
                    </div>
                </div>
                <div class="row mt-4">
    <!-- Checkbox y etiqueta para indicar si el evento requiere talla de polera -->
    <div class="col-md-6">
        <input name="talla" type="checkbox"
            class="form-check-input border-dark fecha-editar" onchange="datoCambiado()"
            id="tallaCheck" @if ($datos['talla']) checked @endif>
        <label class="form-check-label" for="tallaCheck">Requiere talla de polera</label>
    </div>
</div>
```

```
<!-- Contenedor de la segunda mitad del formulario -->
</div>

<!-- Sección de duración del evento -->
<div class="col-md-6">
    <div class="col-md-12 mt-4 ms-3">
        <h6>Duración del evento *</h6>
    </div>
    <div class="row mt-3 ms-3">
        <!-- Campo de inicio del evento -->
        <div class="col-md-1 col-sm-3 align-self-center">Inicio:</div>
        <div class="col-lg-12 col-md-12 col-sm-9">
            <input name="inicio_evento" id="fechaInicio" class="form-control fecha-editar"
                type="datetime-local" onchange="datoCambiado()" min=""
                value="{{ isset($datos['inicio_evento']) ? $datos['inicio_evento'] : '' }}" required />
            <div class="invalid-feedback" id="mensajeErrorFechaInicio"></div>
        </div>
        <!-- Campo de fin del evento -->
        <div class="col-md-1 col-sm-3 align-self-center">Fin:</div>
        <div class="col-lg-12 col-md-12 col-sm-9">
            <input name="fin_evento" id="fechaFin" class="form-control" type="datetime-local"
                onchange="datoCambiado()" min=""
                value="{{ isset($datos['fin_evento']) ? $datos['fin_evento'] : '' }}"required
                {{ strtotime($datos['inicio_evento']) > time() ? 'disabled' : '' }} />
            <div class="invalid-feedback" id="mensajeErrorFechaFin"></div>
        </div>
    </div>
    <!-- Campo de descripción del evento -->
    <div class="col-md-12 mt-5 ms-3">
        <label for="descripcionDelEvento" class="form-label">Descripción del evento</label>
```

```
        <textarea name="descripcion" class="form-control" id="descripcionDelEvento" rows="8" style="resize: none;"
            onchange="datoCambiado()" placeholder="Ingrese una descripción..." maxlength="2048">{{ $datos['descripcionDelEvento'] }}</textarea>
    </div>
    <!-- Checkbox para notificación de edición (aparece solo en caso de edición) -->
    @if(Route::currentRouteName() == 'evento.editar')
        <div class="col-md-12 mt-5 ms-3 form-check form-switch">
            <input name="notificacion" class="form-check-input" type="checkbox"
                id="notificacion" role="switch" checked>
            <label for="notificacion" class="form-check-label">Enviar notificación de la edición</label>
        </div>
    @endif
    <!-- Botones de cancelar y confirmar -->
    <div class="row mt-4 text-center">
        <div class="col-md-6">
            <!-- Botón para abrir el modal de confirmación de cancelación -->
            <button type="button" class="btn btn-light text-primary" data-bs-toggle="modal"
                data-bs-target="#modalCancelar">
                Cancelar
            </button>
            <!-- Modal de confirmación de cancelación -->
            @component('components.modal')
                @slot('modalId', 'modalCancelar')
                @slot('modalTitle', 'Confirmación')
                @slot('modalContent')
                    @if (Route::currentRouteName() == 'evento.editar')
                        ¿Está seguro de cancelar la edición del evento?
                    @else
                        ¿Está seguro de cancelar la creación del evento?
                    @endif
                @endslot
                @slot('modalButton')
                    <!-- Botones dentro del modal de confirmación -->
                    <button type="button" class="btn btn-secondary w-25 mx-8"
                        data-bs-dismiss="modal">No</button>
                    <button type="reset" class="btn btn-primary w-25 mx-8" data-bs-dismiss="modal"
                        onclick="cerrar()">Sí</button>
                @endslot
            @endcomponent
        </div>
        <div class="col-md-6">
            <!-- Botón para abrir el modal de confirmación de confirmación -->
            <button type="button" class="btn btn-primary" data-bs-toggle="modal"
                data-bs-target="#modalConfirmacion">
                Confirmar
            </button>
            <!-- Modal de confirmación de confirmación -->
            @component('components.modal')
                @slot('modalId', 'modalConfirmacion')
                @slot('modalTitle', 'Confirmación')
                @slot('modalContent')
                    @if (Route::currentRouteName() == 'evento.editar')
                        ¿Está seguro de editar el evento?
                    @else
                        ¿Está seguro de crear el evento?
                    @endif
                @endslot
                @slot('modalButton')
                    <!-- Botones dentro del modal de confirmación -->
                    <button type="button" class="btn btn-secondary w-25 mx-8" data-bs-dismiss="modal">No</button>
                    <button type="submit" class="btn btn-primary w-25 mx-8">Sí</button>
                @endslot
            @endcomponent
        </div>
    </div>
</div>
```

7.2.1.3. Interfaz de usuario de crear evento

La interfaz de usuario para la creación de eventos se ha diseñado de manera intuitiva y funcional, brindando a los usuarios una experiencia clara y eficiente. A continuación, se describen los elementos clave de la interfaz:

Formulario de Creación de Evento:

Nombre del Evento: Campo de texto para ingresar el nombre del evento.

Tipo de Evento: Menú desplegable que permite seleccionar el tipo de evento.

Región: Menú desplegable para elegir la región del evento (Internacional, Nacional, Departamental).

Instituciones Admitidas: Lista de instituciones con casillas de verificación para seleccionar las instituciones permitidas.

Grado Académico Requerido: Lista de grados académicos con casillas de verificación para seleccionar los requeridos.

Configuración del Evento:

Género Admitido: Casilla de verificación con menú desplegable para seleccionar el género (Femenino, Masculino).

Rango de Edad: Casilla de verificación con campos de entrada para especificar el rango mínimo y máximo de edad.

Por Equipos: Casilla de verificación para indicar si el evento es por equipos.

Evento de Pago: Casilla de verificación con campo de entrada para el costo del evento.

Requiere Talla de Polera: Casilla de verificación para indicar si se requiere talla de polera.

Duración del Evento:

Fecha y Hora de Inicio y Fin: Campos de entrada de fecha y hora para establecer el inicio y fin del evento.

Descripción del Evento:

Campo de Texto: Área extensa para ingresar una descripción del evento.

Botones de Acción:

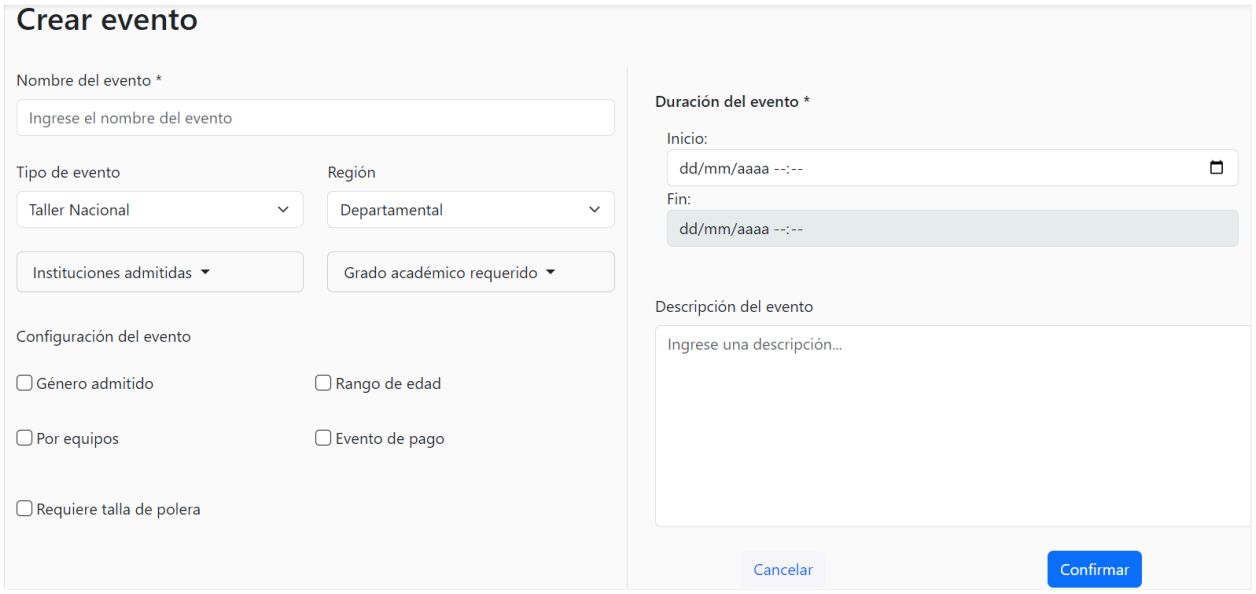
Cancelar: Botón para cancelar la creación o edición del evento con confirmación modal.

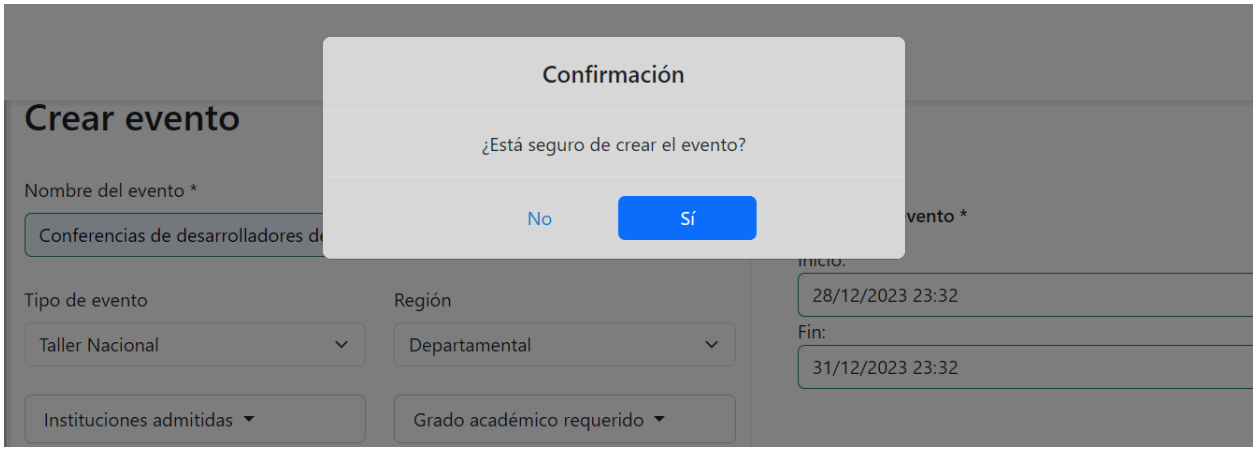
Confirmar: Botón para confirmar y enviar el formulario, con modal de confirmación.

Validación en Tiempo Real:

Mensajes de Validación: Mensajes visuales que indican campos obligatorios y ofrecen retroalimentación en tiempo real.

Desactivación de Campos: Algunos campos se desactivan dinámicamente según las selecciones del usuario.

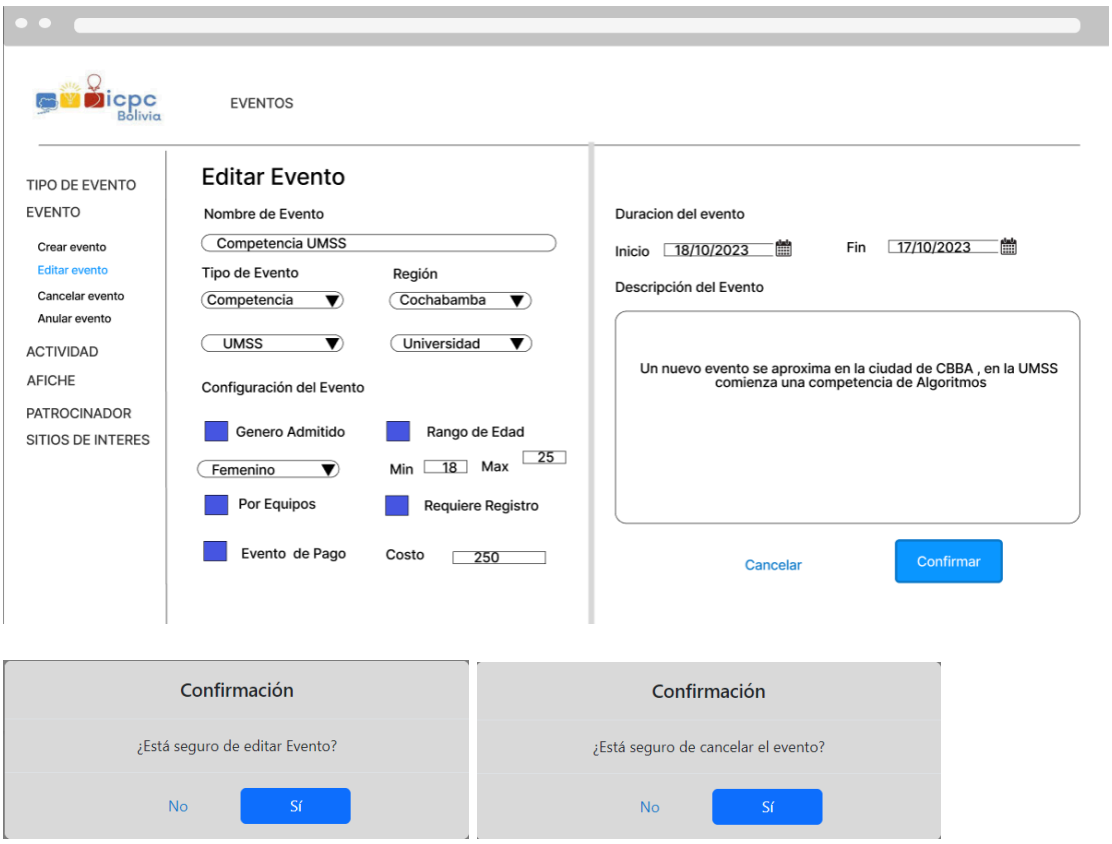




7.2.2. Editar evento

7.2.2.1. Historia de usuario de editar evento

Historia de Usuario	
Título: Editar evento	**ID: 10**
Estimación: 13	**Importancia: Alta**
Descripción Yo como usuario con privilegios para editar eventos, quiero editar la información de todos los detalles de un evento para mantener la información actualizada o corregir algún error.	
Mockups	



Criterios de aceptación

1. Al hacer clic en la sección “EVENTO” en el menú lateral y seleccionar la opción “Editar evento” se carga una tabla con los eventos creados previamente.
2. La tabla contiene las siguientes columnas “Nombre del evento”, “Tipo de evento”, “Fecha de inicio del evento”, “Fecha de fin del evento” y “Acción”.
3. La columna “Acción” tiene un botón “ ” para cada evento.
4. Solo se muestran eventos cuya fecha de finalización es mayor o igual a la fecha actual.
5. Al presionar el botón “ ” se redirige a una ventana con el formulario para editar un evento con los datos del evento registrado previamente.
6. El formulario tiene los siguientes campos: “Nombre del evento *”, “Tipo de evento”, “Región”, “Instituciones admitidas”, “Grado académico requerido”, Configuración del evento (“Género admitido”, “Rango de edad”, “Por equipos”, “Requiere talla de polera”, “Evento de pago”); Duración del evento *(“Inicio”, “Fin”), “Descripción del evento”; un interruptor “Enviar notificación de la edición”, los botones “Cancelar” y “Confirmar”.
7. El valor inicial del interruptor es el de encendido .

8. El campo “Nombre del evento *” contiene el nombre registrado al crear el evento.
9. El campo “Nombre del evento *” es obligatorio.
10. El campo “Nombre del evento *” es único.
11. El campo “Nombre del evento *” tiene como máximo 64 caracteres.
12. El campo “Nombre del evento *” está desactivado si la fecha de inicio del evento es menor a la fecha actual.
13. El campo “Tipo de evento” contiene el tipo de evento registrado al crear el evento.
14. El campo “Tipo de evento” está desactivado si la fecha de inicio del evento es menor a la fecha actual.
15. El campo “Tipo de evento” despliega los tipos de eventos ya registrados sólo si la fecha de inicio del evento creado previamente indica que el evento aún no empezó o no está en curso.
16. Si en la “Duración del evento *” el campo “Inicio” indica que el evento está en curso el mismo se encuentra desactivado.
17. Si el evento se encuentra en curso las opciones que inician marcadas del selector “Instituciones admitidas” se encuentran deshabilitados.
18. Si el evento se encuentra en curso las opciones que inician marcadas del selector “Grado académico requerido” se encuentran deshabilitados.
19. Si el evento se encuentra en curso los campos “Género admitido”, “Rango de edad”, “Por equipos”,”Requiere registro”, “Evento de pago” se encuentran deshabilitados.
20. El campo “Descripción del evento” tiene como máximo de 2048 caracteres
21. Los campos de “Duración de evento *” son obligatorios.
22. El apartado de “Duración del evento *” tiene 2 campos para fechas “Inicio” y “Fin”.
23. El apartado de “Duración de evento *” despliega calendarios para registrar las fechas.
24. Los campos “Inicio” y “Fin” contienen las fechas de inicio y fin registrados al crear el evento.
25. Los campos que son fechas tienen el formato DD-MM-AAAA y hh:mm.
26. Si la fecha del campo “Inicio” es menor a la fecha actual, se muestra el mensaje de “Seleccione una fecha correcta.” debajo del campo “Inicio”.
27. Si el evento se encuentra en curso la fecha del campo “Inicio” comienza inhabilitada.
28. Al ingresar una nueva fecha en campo “Inicio” que es mayor a la fecha actual, el campo de la fecha “Fin”se habilitará.
29. Al ingresar una nueva fecha en el campo “Inicio” que tiene un valor menor a la fecha actual se restablece el campo de la fecha “Fin” y se deshabilita.
30. Si la fecha del campo “Fin” es menor al campo de la fecha “Inicio”, se muestra el mensaje de “Seleccione una fecha correcta.”.
31. El campo “Rango de edad” es una casilla de verificación.
32. Al marcar la casilla “Rango de edad“ se muestran dos inputs con la edad mínima (Min) y la edad máxima (Max).
33. Al menos uno de los campos “Min” o “Max” requiere un valor.
34. El valor mínimo para el campo “Min” es 10.
35. El valor mínimo para el campo “Max” es 10.
36. El valor máximo para el campo “Min” es 99.

37. El valor máximo para el campo “Max” es 99.
38. Si el valor del campo “Min” o “Max” es menor al valor mínimo establecido, aparece un mensaje de validación “Rango de edades no válido”.
39. Si el valor del campo “Min” o “Max” es mayor al valor máximo establecido, aparece un mensaje de validación “Rango de edades no válido”.
40. Si el valor del campo “Min” es mayor que el valor del campo “Max” aparece un mensaje de validación “Rango de edades no válido”.
41. Si solo el campo “Min” tiene un valor, aparece un mensaje de validación “Edad válida desde los (valor del campo “Min”) años”.
42. Si solo el campo “Max” tiene un valor, aparece un mensaje de validación “Edad válida hasta los (valor del campo “Max”) años”.
43. El campo “Género admitido” es una casilla de verificación.
44. Al marcar la casilla “Género admitido” se muestra un selector.
45. El selector de “Género admitido” muestra dos opciones “Femenino” y “Masculino”.
46. Si la información del evento recuperado no tenía seleccionada el selector “Género admitido”, el selector de “Género admitido” está por defecto con la opción “Femenino”.
47. El campo “Evento de pago” es una casilla de verificación.
48. Al marcar la casilla “Evento de pago” se muestra un campo “Costo”.
49. El valor mínimo del campo “Costo” es de 1.
50. El valor máximo del campo “Costo” es de 5000.
51. Si se ingresa un valor superior al valor máximo en el campo “Costo” aparece un mensaje de validación “Monto máximo 5000 Bs.”.
52. Si se ingresa un valor inferior al valor mínimo en el campo “Costo” aparece un mensaje de validación “Monto mínimo 1 Bs.”.
53. El campo “Costo” solo admite un decimal.
54. Al marcar la casilla “Por equipos“ se muestran dos campos, “Min” y “Max”, para la cantidad de participantes admitidos por equipo.
55. El campo “Max” requiere un valor.
56. El campo “Min” puede estar vacío.
57. El valor mínimo para el campo “Min” es 2.
58. El valor del campo “Max” no puede ser menor al valor del campo “Min” .
59. El valor máximo para el campo “Min” es 50.
60. El valor máximo para el campo “Max” es 50.
61. El campo “Requiere talla de polera” es una casilla de verificación.
62. El campo “Grado académico requerido” es un menú desplegable.
63. El menú desplegable de “Grado académico requerido” tiene las opciones de “Todas”, “Primaria”, “Secundaria”, “Universidad”,”Licenciatura”,”Maestría” y ”Doctorado” los cuales son casillas de selección.
64. La casilla de selección “Todas” marca y desmarca a todas las casillas de selección del menú desplegable de “Grado académico requerido”.
65. El menú desplegable de “Instituciones admitidas” tiene los valores de “TODAS”, “UMSS”, “UMSA”, ”UPSA”, ”UCB”, “UPB” y ”UNIFRANZ” los cuales son casillas de selección.

66. La casilla de selección “TODAS” marca y desmarca a todas las casillas de selección del menú desplegable de “Instituciones admitidas”.
67. Al hacer clic en el botón “Cancelar” aparece un modal de confirmación con el mensaje de ¿Está seguro de cancelar la edición del evento?, con los botones de “No” y “Sí” .
68. Al hacer clic en el botón “Sí” se redirige a la sección Editar evento.
69. Al hacer clic en el botón “No” no se realiza ninguna acción.
70. Al hacer clic el botón “No” se cierra el modal.
71. Al hacer clic en el botón “Confirmar” aparece un modal de confirmación con el mensaje de “¿Está seguro de editar el evento?”, con los botones de “No” y “Sí”.
72. Si el campo “Nombre del evento *” ya fue registrado, al hacer clic en el botón “Si” se muestra una alerta temporal con el mensaje “El evento ya existe”.
73. Si el campo “Nombre del evento *” ya fue registrado, al hacer clic en el botón “Si” se muestra un mensaje de validación “El evento ya existe” debajo del campo “Nombre del evento *”.
74. Si el campo “Nombre del evento *” está vacío, al hacer clic en el botón “Si” se muestra un mensaje de validación “El nombre no puede estar vacío” debajo del campo “Nombre del evento *”.
75. Si el interruptor está encendido  al hacer clic en el botón “Sí”, se envía una notificación sobre los cambios al correo electrónico de todos los participantes inscritos en el evento.
76. Si los campos están correctamente llenados, al hacer clic en el botón “Sí” se muestra una alerta temporal con el mensaje “Actualizado exitosamente”.
77. Al finalizar el tiempo de la alerta temporal, se redirige a la sección Editar evento.

7.2.2.2. Código fuente de editar evento

Controlador de editar evento

El método ‘update’ recibe los datos del formulario mediante un objeto Request y el identificador único del evento a modificar. Comienza localizando el evento en la base de datos a través de su ID y guarda los atributos originales para su posterior uso en notificaciones. Luego, actualiza los atributos del evento con los datos provenientes del formulario. Si la opción de notificación está activada, invoca un método notificarCambios para informar sobre las modificaciones realizadas. El código maneja posibles excepciones de la base de datos, como la violación de unicidad (error 1062) o la longitud del campo (error 1406), proporcionando respuestas específicas en formato JSON en caso de error.

Ruta del archivo: app/Http/Controllers/EventoController.php

```php
/**
 * Actualiza la información de un evento existente.
 */
public function update(Request $request, $id)
{
    try {
        // Obtener el evento existente por su ID
        $evento = Evento::findOrFail($id);
        // Guardar los atributos antiguos para la notificación posterior (si se activa)
        $atributosAntiguos = $evento->getOriginal();
        // Actualizar los atributos del evento con los datos del formulario
        $evento->nombre             = $request->nombre;
        $evento->descripcion        = $request->descripcion;
        $evento->inicio_evento      = $request->inicio_evento;
        $evento->fin_evento         = $request->fin_evento;
        $evento->institucion        = $request->institucion;
        $evento->region             = $request->region;
        $evento->grado_academico    = $request->grado_academico;
        $evento->equipo_minimo      = $request->equipo_minimo;
        $evento->equipo_maximo      = $request->equipo_maximo;
        $evento->talla              = $request->talla;
        $evento->edad_minima        = $request->edad_minima;
        $evento->edad_maxima        = $request->edad_maxima;
        $evento->genero             = $request->genero;
        $evento->precio_inscripcion = $request->precio_inscripcion;
        $evento->id_tipo_evento     = $request->id_tipo_evento;
        // Guardar los cambios en la base de datos
        $evento->save();

        // Notificar cambios si la opción de notificación está activada
        if ($request->notificacion) {
            $this->notificarCambios($evento, $atributosAntiguos);
        }
        // Respuesta exitosa en formato JSON
        return response()->json(['mensaje' => 'Actualizado exitosamente', 'error' => false]);
    } catch (QueryException $e) {
        // Manejar excepciones de la base de datos
        if ($e->errorInfo[1] == 1062) {
            return response()->json(['mensaje' => 'El evento ya existe', 'error' => true]);
        } else {
            if ($e->errorInfo[1] == 1406) {
                return response()->json(['mensaje' => 'Campo demasiado grande', 'error' => true]);
            } else {
                return $e->getMessage();
            }
        }
    }
}
```

Código JavaScript de editar evento

Si es una edición, se inicializan los elementos de la interfaz de usuario con los valores actuales del evento seleccionado. Además, se establece un conjunto de variables booleanas que rastrean cambios en diferentes secciones del formulario, como la información de equipo, las fechas, el costo, y más. El formulario contiene campos relacionados con el género, rango de edad, costo, equipo, y otros detalles específicos del evento. Se utiliza la biblioteca Axios para realizar peticiones HTTP, en este caso, para actualizar la información del evento en el servidor. El código incluye funciones específicas para la gestión de eventos, como mostrarInput, que controla la visibilidad de ciertos campos dependiendo de las selecciones del usuario. Además, hay funciones dedicadas a la preparación de los datos del formulario y su posterior envío al servidor para la actualización del evento. Durante la edición, se implementan lógicas para

deshabilitar ciertos campos si la fecha actual es posterior a la fecha de inicio del evento, evitando así modificaciones en información pasada. También se manejan validaciones para evitar la repetición de nombres de eventos y se muestra una alerta si el nombre ya existe.

Ruta del archivo: public/js/Evento/crearEvento.js

```
// Declaración de variables que representan elementos del DOM
const inputGenero = document.getElementById("generoCheck");
const inputEdad = document.getElementById("edadCheck");
const inputCosto = document.getElementById("eventoPagoCheck");
const form = document.getElementById("formularioCrearEvento");
const fechaInicio = document.getElementById("fechaInicio");
const fechaFin = document.getElementById("fechaFin");
const edadMinima = document.getElementById("edadMinima");
const edadMaxima = document.getElementById("edadMaxima");
const chekcTodas = document.getElementById("check-institucion-TODAS");
const costo = document.getElementById("costoEvento");
const checkTodasRango = document.getElementById("input-grado-Todas");
const nombreEvento = document.getElementById("nombreDelEvento");
const mensajeNombre = document.getElementById("mensajeNombre");
const checkEquipo = document.getElementById("equipoCheck");
const checkNotificacion = document.getElementById("notificacion");
const equipoMaximo = document.getElementById("equipoMaximo");
const equipoMinimo = document.getElementById("equipoMinimo");

let boolCheckEquipo = true;
let boolMinEquipo = true;
let boolMaxEquipo = true;
let boolFecha = true;
let boolCosto = true;
let boolMinEdad = true;
let boolcheckEdad = true;
let boolMaxEdad = true;
let crear = true;
let datosActualizados = false;
let nombreAnterior = '';
let mensajeRepetido = '';

let fechaLocal = new Date();
fechaLocal.setHours(fechaLocal.getHours() - 4);
let laFecha = fechaLocal.toISOString().substring(0, 16);
```

```
// Función que muestra u oculta un elemento del formulario según el estado de una casilla de verificación
const mostrarInput = (indInput, check) => {
    let input = document.getElementById(indInput);
    if (!check) {
        input.style.display = "none";
    } else {
        input.style.display = "flex";
    }
};

// Función que prepara los datos del formulario para ser enviados mediante una solicitud
const prepararFormData = () => {
    if (!crear) {
        document.querySelectorAll(".fecha-editar").forEach(Element => {
            Element.disabled = false;
        });
    }
    const formData = new FormData(form);
    if (!inputEdad.checked) {
        formData.set("edad_minima", "");
        formData.set("edad_maxima", "");
    }
    if (!inputGenero.checked) {
        formData.set("genero", "");
    }
    if (!inputCosto.checked) {
        formData.set("precio_inscripcion", "");
    }
    if (!checkEquipo.checked) {
        formData.set("equipo_minimo", "");
        formData.set("equipo_maximo", "");
    }
    if (!document.getElementById("tallaCheck").checked) {
        formData.set("talla", "");
    }
    if (!crear) {
        if (!checkNotificacion.checked) {
            formData.set("notificacion", "");
        } else {
            formData.set("notificacion", "on");
```

```
            }
        }

        return formData;
};

// Función que realiza la edición de un evento mediante una solicitud a la API
const editarEvento = (formData) => {
    if (!datosActualizados) {
        window.location.href =
            "/eventos/" + document.getElementById("nombreDelEvento").value;
    }
    nombreAnterior = nombreEvento.value;
    axios
        .post("/api/evento/actualizar/" + formData.get("evento_id"), formData)
        .then(function (response) {
            mostrarAlerta(
                "Éxito",
                response.data.mensaje,
                response.data.error ? "danger" : "success"
            );
            mensajeRepetido = response.data.mensaje;
            if (mensajeRepetido !== 'El evento ya existe') {
                setTimeout(() => {
                    window.location.href = "/editarEvento";
                }, 1800);
                form.reset();
            } else {
                nombreEvento.classList.remove("is-valid");
                nombreEvento.classList.add("is-invalid");
                mensajeNombre.textContent = 'El evento ya existe.';
            }
        })
        .catch(function (error) {
            mostrarAlerta(
                "Error",
                "Hubo un error al guardar el tipo de evento",
                "danger"
            );
```

```
        });
};

// Función que realiza la creación de un evento mediante una solicitud a la API
const crearEvento = (formData) => {
    nombreAnterior = nombreEvento.value;
    axios
        .post("/api/evento", formData)
        .then(function (response) {
            mostrarAlerta(
                "Éxito",
                response.data.mensaje,
                response.data.error ? "danger" : "success"
            );
            mensajeRepetido = response.data.mensaje;
            if (mensajeRepetido === 'El evento ya existe') {
                nombreEvento.classList.remove("is-valid");
                nombreEvento.classList.add("is-invalid");
                mensajeNombre.textContent = 'El evento ya existe.';
            } else {
                setTimeout(() => {
                    window.location.href = "/eventos/";
                }, 1800);
            }
        })
        .catch(function (error) {
            mostrarAlerta(
                "Error",
                "Hubo un error al guardar el tipo de evento",
                "danger"
            );
        });
};

// Función que se ejecuta cuando se detecta un cambio en los datos del formulario
const datoCambiado = () => {
    if (!crear) {
        datosActualizados = true;
    }
```

```
};

// Inicialización de ciertas acciones cuando la ventana ha cargado completamente
window.addEventListener("load", () => {
    checkTodasRango.classList.remove("grado-requerido");
    chekcTodas.classList.remove("institucion");
    if (document.getElementById("nombreDelEvento").value != "") {
        crear = false;
        iniciarEditar();
    } else {
        fechasMin();
        fechaFin.disabled = true;
    }
    axios
        .get("/api/tipo-evento")
        .then(function (response) {
            const idTipoEvento = document.getElementById("tipoDelEvento").getAttribute("data-id");
            const select = document.getElementById("tipoDelEvento");
            const tiposDeEvento = response.data;
            tiposDeEvento.forEach(function (tipo) {
                if (idTipoEvento != tipo.id) {
                    const option = document.createElement("option");
                    option.value = tipo.id;
                    option.text = tipo.nombre;
                    select.appendChild(option);
                }
            });
            if (idTipoEvento != "") {
                select.value = idTipoEvento;
            }
        })
        .catch(function (error) {
            console.error(error);
        });
});

// Función que se ejecuta al cerrar el formulario de edición o creación de eventos
const cerrar = (edicion) => {
    if (crear) {
```

```
            location.reload();
            form.reset();
        } else {
            window.location.href = "/editarEvento/";
        }
};

// Event listener para el envío del formulario
form.addEventListener("submit", (event) => {
    event.preventDefault();
    form.querySelectorAll(".form-control, .form-select").forEach((Element) => {
        Element.dispatchEvent(new Event("change"));
    });
    if (!validar()) {
        $("#modalConfirmacion").modal("hide");
    } else {
        let aux = "";
        form.querySelectorAll(".institucion").forEach((Element) => {
            if (Element.checked) {
                aux = Element.value + "-" + aux;
            }
        });
        let insti = "";
        form.querySelectorAll(".grado-requerido").forEach((Element) => {
            if (Element.checked) {
                insti = Element.value + "-" + insti;
            }
        });
        const formData = prepararFormData();
        formData.set("institucion", aux.slice(0, -1));
        formData.set("grado_academico", insti.slice(0, -1));
        if (!crear) {
            editarEvento(formData);
        } else {
            crearEvento(formData);
        }
        $("#modalConfirmacion").modal("hide");
    }
});
```

```
// Función de validación del formulario
const validar = () => {
    if (form.querySelector(".is-invalid") === null) {
        return true;
    } else {
        return false;
    }
};

// Event listeners para cambios en las casillas de verificación y campos del formulario
inputGenero.addEventListener("change", () => {
    mostrarInput("genero", inputGenero.checked);
});
inputEdad.addEventListener("change", () => {
    mostrarInput("rangosDeEdad", inputEdad.checked);
});
inputCosto.addEventListener("change", () => {
    mostrarInput("eventoPago", inputCosto.checked);
});
checkEquipo.addEventListener("change", () => {
    mostrarInput("numero-integrantes", checkEquipo.checked);
});

nombreEvento.addEventListener("input", validarNombreRepetido);
nombreEvento.addEventListener("change", validarNombreRepetido);

// Función que se ejecuta al iniciar la edición de un evento existente
const iniciarEditar = () => {
    mostrarInput("genero", inputGenero.checked);
    mostrarInput("rangosDeEdad", inputEdad.checked);
    mostrarInput("eventoPago", inputCosto.checked);
    mostrarInput("numero-integrantes", checkEquipo.checked);
    let f1 = new Date(fechaInicio.value);
    let fl1 = new Date(laFecha);
    let boolGrado = true;
    let boolInstitucion = true;
    const instituciones = document.getElementById("ul-institucion").getAttribute("data-institucion");
    document.querySelectorAll(".institucion").forEach(Element => {
        if (instituciones.includes(Element.value)) {
```

```
                Element.checked = true;
                Element.classList.add("fecha-editar");
            } else {
                boolInstitucion = false;
            }
        });
        const grados = document.getElementById("ul-grado").getAttribute("data-grado");
        document.querySelectorAll(".grado-requerido").forEach(Element => {
            if (grados.includes(Element.value)) {
                Element.checked = true;
                Element.classList.add("fecha-editar");
            } else {
                boolGrado = false;
            }

        });
        checkTodasRango.checked = boolGrado;
        chekcTodas.checked = boolInstitucion;

        if (f1 <= fl1) {
            checkTodasRango.disabled = boolGrado;
            chekcTodas.disabled = boolInstitucion;

            if (!inputGenero.checked) {
                inputGenero.classList.add("fecha-editar");
            }
            document.getElementById("ul-institucion").classList.add("fecha-editar");
            document.getElementById("todas-grado").classList.add("fecha-editar");

            document.querySelectorAll(".fecha-editar").forEach(Element => {
                Element.disabled = true;
            });
            fechaFin.min = laFecha;
        } else {
            fechasMin();
        }
    }
```

```
// Función que establece fechas mínimas para los campos de fecha
const fechasMin = () => {
    let fechaLocal = new Date();
    fechaLocal.setHours(fechaLocal.getHours() - 4);
    let laFecha = fechaLocal.toISOString().substring(0, 16);
    fechaInicio.min = laFecha;
    fechaFin.min = laFecha;
};

// Función que valida si el nombre del evento es único
function validarNombreRepetido() {
    if (nombreAnterior === '') {
        if (nombreEvento.value === '') {
            nombreEvento.classList.remove("is-valid");
            nombreEvento.classList.add("is-invalid");
            mensajeNombre.textContent = "El nombre no puede estar vacío.";
        } else {
            nombreEvento.classList.remove("is-invalid");
            nombreEvento.classList.add("is-valid");
        }
    } else {
        if (nombreEvento.value !== nombreAnterior && nombreEvento.value !== '') {
            nombreEvento.classList.remove("is-invalid");
            nombreEvento.classList.add("is-valid");
        } else if (nombreEvento.value == '') {
            nombreEvento.classList.remove("is-valid");
            nombreEvento.classList.add("is-invalid");
            mensajeNombre.textContent = 'El nombre no puede estar vacío.';
        } else {
            nombreEvento.classList.remove("is-valid");
            nombreEvento.classList.add("is-invalid");
            mensajeNombre.textContent = 'La actividad ya existe';
        }
    }
}
```

Vista de editar evento

Este código corresponde a una vista de Laravel diseñada para la edición de eventos. En el contexto de una aplicación web, la vista utiliza el sistema de plantillas Blade de Laravel y extiende el diseño principal (layouts.app). La interfaz principal consta de un encabezado que indica la acción a realizar ("Editar Evento") y una tabla que muestra eventos disponibles para su edición. La tabla se presenta de manera tabular con columnas que muestran información clave de cada evento, como el nombre del evento, el tipo de evento, la fecha de inicio y la fecha de finalización. Se utiliza una condición @if para asegurarse de que solo se muestren eventos cuya fecha de finalización sea posterior a la fecha actual, evitando eventos pasados. Dentro de la tabla, se proporciona un botón de acción para cada evento que redirige a la página de edición específica del evento.

Ruta del archivo: resources/views/crear-evento/editarEvento.blade.php

```
@extends('layouts.app')

@section('content')
    <div class="container">
        <div class="row mb-2">
            <h2>Editar Evento</h2>
        </div>
        <div class="row g-5">
            <div class="col-sm-12">
                <!-- Tabla que muestra la lista de eventos para editar -->
                <table class="table table-responsive table-striped text-secondary  cursor" id="tablaEvento">
                    <caption>Eventos</caption>
                    <thead>
                        <!-- Encabezados de la tabla -->
                        <tr>
                            <th scope="col" class="col-sm-3 col-md-3">Nombre del evento</th>
                            <th scope="col" class="col-sm-2 col-md-2 text-center">Tipo de evento</th>
                            <th scope="col" class="col-sm-3 col-md-3 text-center">Fecha de inicio del evento</th>
                            <th scope="col" class="col-sm-3 col-md-3 text-center">Fecha de fin del evento</th>
                            <th scope="col" class="col-sm-1 col-md-1 text-center font-sm">Acción</th>
                        </tr>
                    </thead>
                    <tbody id="datosTabla">
                        <!-- Ciclo que recorre los eventos para mostrar en la tabla -->
                        @foreach ($eventos as $evento)
                            <!-- Verifica si la fecha de fin del evento es posterior a la fecha actual -->
                            @if (strtotime($evento->fin_evento) >= time())
                                <tr id="{{ $evento->id }}">
                                    <td>{{ $evento->nombre }}</td>
                                    <td class="text-center">{{ $evento->tipoEvento->nombre }}</td>

                                    <td class="text-center">{{ date('d-m-Y', strtotime($evento->inicio_evento)) }}</td>
                                    <td class="text-center">{{ date('d-m-Y', strtotime($evento->fin_evento)) }}</td>
                                    <td class="text-center">
                                        <!-- Botón para redireccionar a la página de edición del evento -->
                                        <button type="button" class="btn btn-primary btn-sm"
                                            onclick="redireccionarEditar('{{ route('evento.editar', ['id' => $evento->id]) }}')">
                                            <i class="fa-solid fa-pencil"></i>
                                        </button>
                                    </td>
                                </tr>
                            @endif
                        @endforeach
                    </tbody>
                </table>
            </div>
        </div>
    </div>

    <!-- Enlaces a las hojas de estilo y scripts necesarios para DataTables -->
    <link href="https://cdn.datatables.net/1.13.6/css/dataTables.bootstrap5.min.css" rel="stylesheet">
    <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.7.1/jquery.min.js"></script>
    <script src="https://cdn.datatables.net/1.13.6/js/jquery.dataTables.min.js"></script>
    <script src="https://cdn.datatables.net/1.13.6/js/dataTables.bootstrap5.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/moment.js/2.29.2/moment.min.js"></script>
    <!-- Script personalizado para la funcionalidad de la tabla de eventos -->
    <script src="{{ asset('js/Evento/tablaEditarEvento.js') }}" defer></script>
@endsection
```

7.2.2.3. Interfaz de usuario de editar evento

La interfaz de usuario para editar eventos ha sido diseñada para proporcionar una experiencia intuitiva y eficiente al usuario. A continuación, se detallan los elementos clave de la interfaz:

Encabezado:

En la parte superior, se encuentra un encabezado claro que indica la acción principal que se realizará en esta página, en este caso, "Editar Evento".

Tabla de Eventos:

La tabla presenta información tabular sobre los eventos disponibles para su edición.

Las columnas incluyen el nombre del evento, el tipo de evento, la fecha de inicio y la fecha de finalización.

Se ha aplicado una condición para mostrar solo los eventos cuya fecha de finalización sea posterior a la fecha actual, evitando la visualización de eventos pasados.

Formulario de Edición:

Al presionar el botón con icono de lápiz para un evento específico, se redirige al usuario a un formulario de edición detallado.

Este formulario incluye campos editables para el nombre del evento, tipo de evento, fecha de inicio, fecha de finalización, y cualquier otra información relevante.

Los campos del formulario están prellenados con la información existente del evento seleccionado, facilitando la modificación de datos existentes.

Botón de Confirmar:

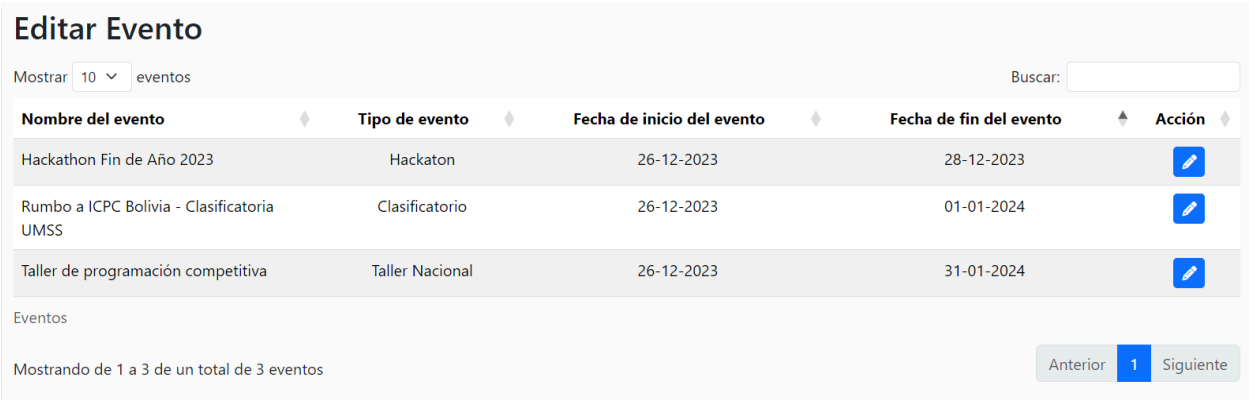
Después de realizar las ediciones necesarias, el usuario tiene la opción de guardar los cambios mediante un botón de “Confirmar”.

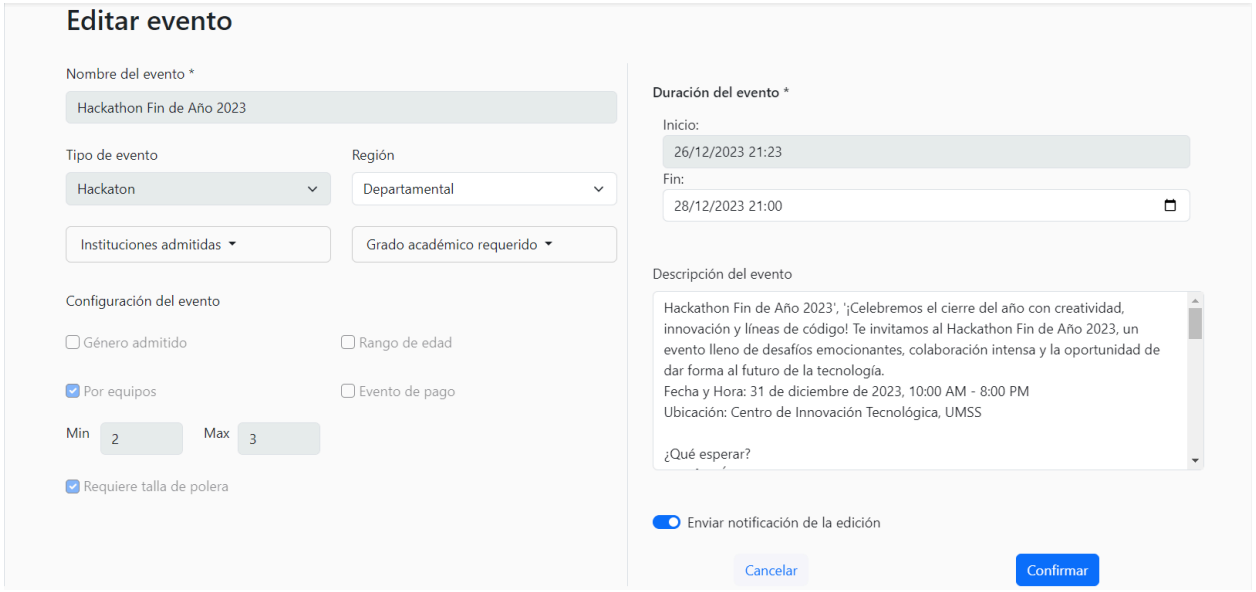
Al hacer clic en este botón, se procesa la actualización del evento con los nuevos datos ingresados en el formulario.

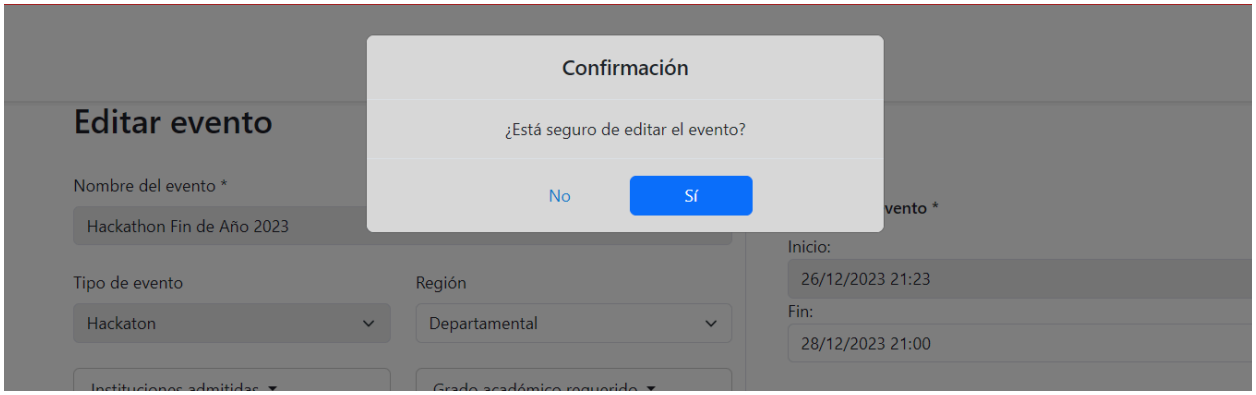
Estilo y Legibilidad:

Se ha aplicado un estilo de diseño limpio y legible, utilizando colores y tipografía coherentes con la identidad visual de la aplicación.

La información se presenta de manera organizada para facilitar la identificación y selección de eventos.



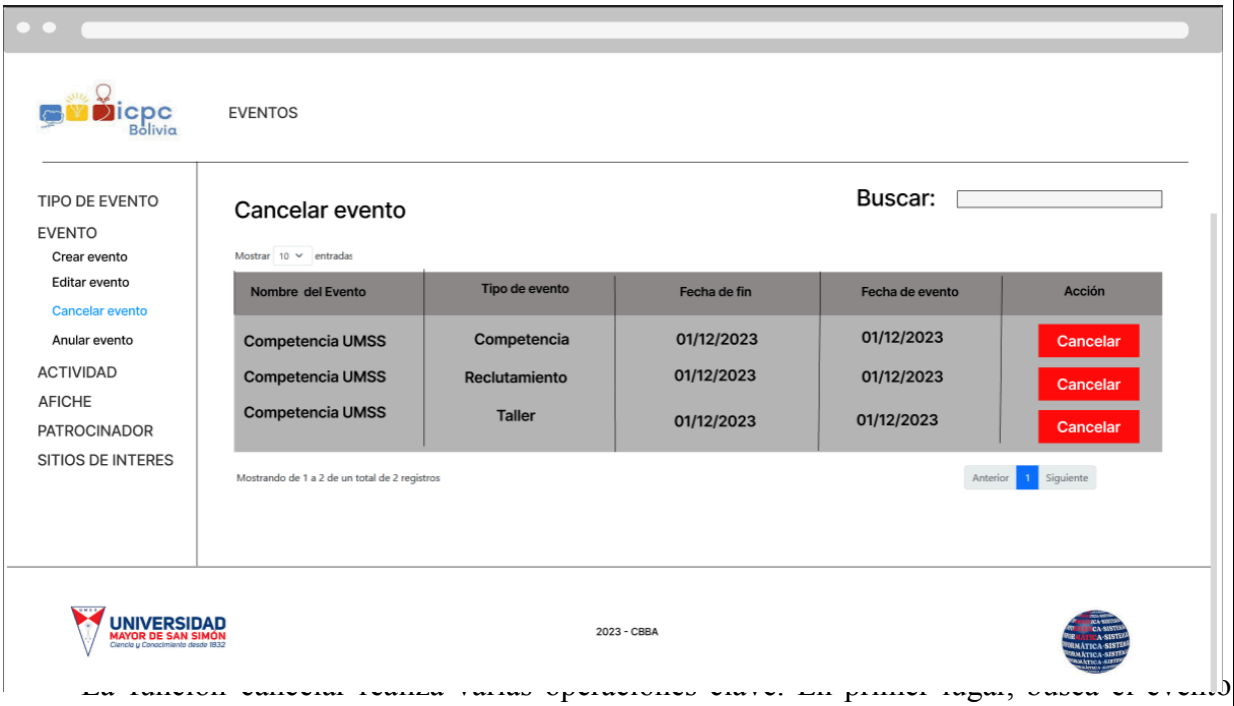


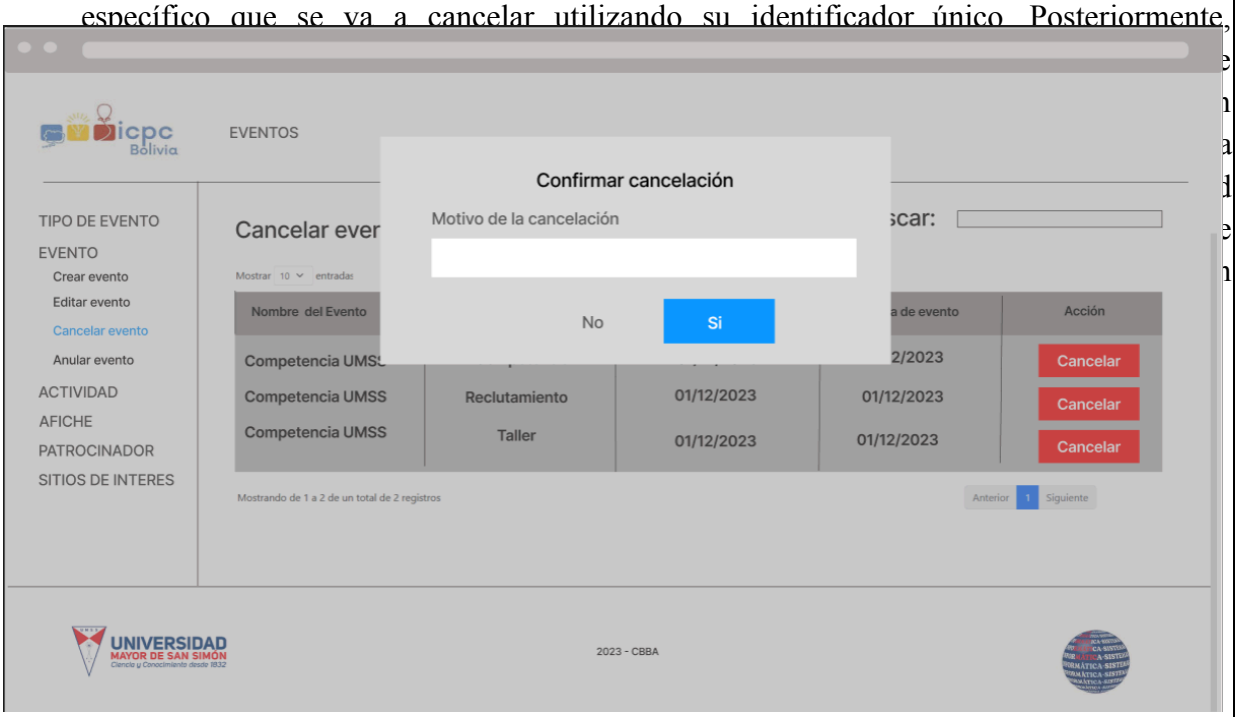


7.2.3. Cancelar evento

7.2.3.1. Historia de usuario de cancelar evento

Historia de Usuario	
Título: Cancelar evento	**ID: 9**
Estimación: 13	**Importancia: Media**
Descripción: Yo como usuario con privilegio de administrar eventos, quiero cancelar un evento por algún imprevisto durante la planificación del mismo.	
Mockups	





Criterios de aceptación.

1. Al hacer clic en la sección “EVENTO” en el menú lateral y seleccionar la opción “Cancelar evento” se muestra una tabla con los eventos creados previamente.
2. La tabla tiene las columnas “Nombre del evento”, “Tipo de evento”, “Fecha de inicio del evento ”, “Fecha de fin del evento” y “Acción”.

```php
/**
 * Cancela un evento, actualizando su estado y registrando el motivo de la cancelación.
 */
public function cancelar(Request $request, $id)
{
    // Buscar el evento por su identificador único
    $evento = Evento::find($id);

    // Actualizar el estado del evento a "Cancelado" (estado 1)
    $evento->estado = 1;
    $evento->save();

    // Crear una instancia de la clase Cancelado y registrar el motivo de la cancelación
    $cancelado = new Cancelado();
    $cancelado->motivo = $request->motivo;
    $cancelado->id_evento = $id;
    $cancelado->save();

    // Responder con un mensaje indicando el éxito de la cancelación
    return response()->json(['mensaje' => 'Cancelado exitosamente', 'error' => false]);
}
```

Código JavaScript de cancelar evento

El script comienza declarando variables globales como tablaDeTipos, tablaInicializada, idEvento, y contraseña. Luego, configura las opciones para la tabla DataTable, como la longitud de página, menú de longitud, orden y configuración de idioma. Al cargar la página, se ejecuta un evento que inicializa la DataTable y configura un patrón de contraseña en un campo específico. La inicialización de la DataTable se encarga de destruir cualquier instancia existente antes de crear una nueva, utilizando las opciones previamente definidas. La función setEventoId establece el ID del evento seleccionado y puede reiniciar formularios modales según si se está anulando o cancelando un evento. Las funciones cancelarEvento y anularEvento realizan operaciones de cancelación y anulación, respectivamente, utilizando datos del formulario y realizando solicitudes a través de Axios. Además, hay funciones para crear objetos FormData con datos específicos. La función validarAnulacion asegura que el formulario de anulación esté validado antes de enviar la solicitud. La función guardarForm maneja las solicitudes y muestra alertas de éxito o fracaso según la respuesta obtenida. Las funciones resetModalCancelar y resetModalAnular reinician los formularios modales de cancelación y anulación, respectivamente. La función recargarEventos redirige a la página de eventos después de un tiempo determinado.

Ruta del archivo: public/js/cancelarEvento.js

```
// Definición de variables globales
let tablaDeTipos;
let tablaInicializada = false;
let idEvento = null;
let contraseña = "admin123";

// Configuración de opciones para la tabla DataTable
const dataTableOptions = {
    pageLength: 10,
    lengthMenu: [5, 10, 15, 20],
    destroy: true,
    order: [[3, "desc"]],
    language: {
        // Configuración de idioma para la tabla
        // ... (se incluyen mensajes en español)
    },
};

// Evento que se ejecuta cuando la página se carga
window.addEventListener("load", () => {
    // Inicialización de la tabla DataTable
    initDataTable();
    // Configuración del patrón de la contraseña en un campo específico
    document.getElementById("contrasenia").pattern = contraseña;
});

// Función para inicializar la tabla DataTable
const initDataTable = async () => {
    // Verificación y destrucción de la tabla si ya está inicializada
    if (tablaInicializada) {
        tablaDeTipos.destroy();
    }
    // Inicialización de la tabla DataTable con opciones específicas
    DataTable.datetime("DD-MM-YYYY");
    tablaDeTipos = $("#tablaEvento").DataTable(dataTableOptions);
    tablaInicializada = true;
};
```

```
// Función para establecer el ID del evento seleccionado
const setEventoId = (id, anular) => {
    idEvento = id;
    // Reinicio de formularios modales según la acción (anular o cancelar)
    if (anular)
        resetModalAnular();
    else
        resetModalCancelar();
};

// Función para cancelar un evento
const cancelarEvento = async () => {
    // Creación del objeto FormData con los datos del formulario de cancelación
    let formData = crearFormDataCancelar();
    // Invocación de la función para realizar la solicitud de cancelación
    guardarForm(
        `/api/evento/cancelar/${idEvento}`,
        formData,
        "Error al cancelar el evento"
    );
    // Reinicio del formulario modal de cancelación
    resetModalCancelar();
    // Recarga de la página de eventos después de un tiempo
    recargarEventos(0);
};

// Función para crear un objeto FormData con datos para cancelar un evento
const crearFormDataCancelar = () => {
    const formData = new FormData();
    formData.append(
        "motivo",
        document.getElementById("motivoCancelacion").value
    );
    return formData;
};
```

```
// Función para anular un evento
const anularEvento = async () => {
    // Creación del objeto FormData con los datos del formulario de anulación
    let formData = crearFormDataAnular();
    // Validación del formulario de anulación antes de la solicitud
    if (validarAnulacion()) {
        // Invocación de la función para realizar la solicitud de anulación
        guardarForm(
            `/api/evento/anular/${idEvento}`,
            formData,
            "Error al anular el evento"
        );
        // Reinicio del formulario modal de anulación
        resetModalAnular();
        // Recarga de la página de eventos después de un tiempo
        recargarEventos(1);
    }
};

// Función para crear un objeto FormData con datos para anular un evento
const crearFormDataAnular = () => {
    const formData = new FormData();
    formData.append("motivo", document.getElementById("motivoAnulacion").value);
    formData.append(
        "descripcion",
        document.getElementById("descripcionAnulacion").value
    );
    formData.append(
        "archivos",
        document.getElementById("archivosRespaldo").files[0]
    );
    return formData;
};
```

```
// Función para validar el formulario de anulación
const validarAnulacion = () => {
    let form = document.getElementById("formularioAnulacion");
    if (form.checkValidity()) {
        form.classList.remove("was-validated");
        return true;
    }
    form.classList.add("was-validated");
    return false;
};

// Función para realizar una solicitud y manejar la respuesta
const guardarForm = async (ruta, formData, msgError) => {
    await axios
        .post(ruta, formData)
        .then((response) => {
            // Mostrar alerta con el resultado de la solicitud
            mostrarAlerta(
                "Éxito",
                response.data.mensaje,
                response.error ? "danger" : "success"
            );
        })
        .catch((error) => {
            // Mostrar alerta en caso de error en la solicitud
            mostrarAlerta("Fracaso", msgError, "danger");
        });
};

// Función para reiniciar el formulario modal de cancelación
const resetModalCancelar = () => {
    $("#modalCancelar").modal("hide");
    document.getElementById("motivoCancelacion").value = "";
};

// Función para reiniciar el formulario modal de anulación
const resetModalAnular = () => {
    $("#modalAnular").modal("hide");
    document.getElementById("formularioAnulacion").classList.remove("was-validated");
    document.getElementById("motivoAnulacion").value = "";
    document.getElementById("descripcionAnulacion").value = "";
    document.getElementById("archivosRespaldo").value = "";
    document.getElementById("contrasenia").value = "";
};

// Función para recargar la página de eventos después de un tiempo
const recargarEventos = (control) => {
    let ruta = control === 0 ? "/admin/eventos/cancelar-evento" : "/admin/eventos/anular-evento";
    setTimeout(() => {
        window.location.href = ruta;
    }, 1800);
};
```

Vista de cancelar evento

La vista utiliza la plantilla Blade de Laravel para generar dinámicamente la tabla de eventos con información como el nombre, tipo, fecha de inicio y fecha de finalización de cada evento. Se incluyen dos modales, uno para la cancelación de eventos y otro para la

anulación, que se activan al hacer clic en botones específicos de la tabla. El código JavaScript gestionado por cancelarEvento.js se encarga de inicializar y manejar eventos en la tabla utilizando DataTables, una biblioteca para tablas interactivas. Además, se implementan funciones para recopilar y validar datos del formulario antes de enviar solicitudes asincrónicas al servidor a través de Axios. También se incluyen funciones para restablecer los formularios de los modales y recargar la página después de realizar acciones de cancelación.

Ruta del archivo: resources/views/eventos/cancelarEvento.blade.php

```
@extends('layouts.app')

@section('content')
    <div class="container">
        <div class="row mb-2">
            <h2>{{ $anular ? 'Anular' : 'Cancelar' }} evento</h2>
        </div>
        <div class="row g-5">
            <div class="col-sm-12">
                <!-- Tabla para mostrar eventos -->
                <table class="table table-responsive table-striped text-secondary  cursor" id="tablaEvento">
                    <caption>Eventos</caption>
                    <thead>
                        <tr>
                            <th scope="col" class="col-sm-3 col-md-3">Nombre del evento</th>
                            <th scope="col" class="col-sm-2 col-md-2 text-center">Tipo de evento</th>
                            <th scope="col" class="col-sm-3 col-md-3 text-center">Fecha de inicio del evento</th>
                            <th scope="col" class="col-sm-3 col-md-3 text-center">Fecha de fin del evento</th>
                            <th scope="col" class="col-sm-1 col-md-1 text-center font-sm">Acción</th>
                        </tr>
                    </thead>
                    <tbody id="datosTabla">
                        @php
                            date_default_timezone_set('America/Caracas');
                        @endphp
                        @foreach ($eventos as $evento)
                            <!-- Verifica si se está anulando o cancelando y si el evento es válido -->
                            @if ($anular)
                                @if (strtotime($evento->inicio_evento) <= time() && strtotime($evento->fin_evento) >= time())
```

```
                        <!-- Fila de la tabla con información del evento -->
                        <tr id="{{ $evento->id }}">
                            <td>{{ $evento->nombre }}</td>
                            <td class="text-center">{{ $evento->tipoEvento->nombre }}</td>
                            <td class="text-center">{{ date('d-m-Y', strtotime($evento->inicio_evento)) }}</td>
                            <td class="text-center">{{ date('d-m-Y', strtotime($evento->fin_evento)) }}</td>
                            <td class="text-center">
                                <!-- Botón para anular el evento -->
                                <button type="button" class="btn btn-danger btn-sm"
                                    onclick="setEventoId({{ $evento->id }},true)" id="botonAccion"
                                    style="min-width: 80px;" data-bs-toggle="modal"
                                    data-bs-target={{ $anular ? '#modalAnular' : '#modalCancelar' }}>
                                    {{ $anular ? 'Anular' : 'Cancelar' }}
                                </button>
                            </td>
                        </tr>
                    @endif
                @else
                    <!-- Verifica si el evento es válido para cancelación -->
                    @if (strtotime($evento->inicio_evento) >= time() && $evento->inscritos->count() == 0)
                        <!-- Fila de la tabla con información del evento -->
                        <tr id="{{ $evento->id }}">
                            <td>{{ $evento->nombre }}</td>
                            <td class="text-center">{{ $evento->tipoEvento->nombre }}</td>
                            <td class="text-center">{{ date('d-m-Y', strtotime($evento->inicio_evento)) }}</td>
                            <td class="text-center">{{ date('d-m-Y', strtotime($evento->fin_evento)) }}</td>
                            <td class="text-center">
                                <!-- Botón para cancelar el evento -->
                                <button type="button" class="btn btn-danger btn-sm"
                                    onclick="setEventoId({{ $evento->id }}, false)"
                                    id="botonAccion" style="min-width: 80px;" data-bs-toggle="modal"
                                    data-bs-target={{ $anular ? '#modalAnular' : '#modalCancelar' }}>
                                    {{ $anular ? 'Anular' : 'Cancelar' }}
                                </button>
                            </td>
                        </tr>
                    @endif
                @endif
            @endforeach
        </tbody>
    </table>

    <!-- Modal para cancelar evento -->
    <div class="modal fade" id="modalCancelar" tabindex="-1" aria-labelledby="modalCancelarLabel"
        aria-hidden="true">
        <div class="modal-dialog">
            <div class="modal-content">
                <div class="modal-header">
                    <h5 class="modal-title" id="modalCancelarLabel">
                        Confirmar cancelación
                    </h5>
                </div>
                <div class="modal-body">
                    <div class="container px-4">
                        <!-- Formulario para ingresar motivo de cancelación -->
                        <label for="motivoCancelacion" class="form-label">
                            Motivo de la cancelación
                        </label>
                        <textarea rows="3" type="text" class="form-control" id="motivoCancelacion"
                            placeholder="Ingrese el motivo de la cancelación" value=""
```

```
                                maxlength="128" style="resize: none"></textarea>
                        </div>
                    </div>
                    <div class="modal-footer d-flex justify-content-center">
                        <!-- Botones para confirmar o cancelar cancelación -->
                        <button type="button" class="btn btn-secondary w-25 mx-8"
                            data-bs-dismiss="modal" onclick="resetModalCancelar()">No</button>
                        <button type="button" class="btn btn-danger w-25 mx-8"
                            onclick="cancelarEvento()">Sí</button>
                    </div>
                </div>
            </div>
        </div>

        <!-- Modal para anular evento -->
        <div class="modal fade" id="modalAnular" tabindex="-1" aria-labelledby="modalAnularLabel"
            aria-hidden="true">
            <div class="modal-dialog">
                <div class="modal-content">
                    <div class="modal-header">
                        <h5 class="modal-title" id="modalAnularLabel">
                            Confirmar anulación
                        </h5>
                    </div>
                    <form class="needs-validation" novalidate id="formularioAnulacion">
                        <div class="modal-body">
                            <div class="container px-5">
                                <!-- Campos para ingresar motivo, descripción, archivo de respaldo y contraseña -->
                                <div class="row">
                                    <label for="motivoAnulacion" class="form-label">
                                        Motivo de la anulación *
                                    </label>
                                    <input type="text" class="form-control" id="motivoAnulacion"
                                        placeholder="Ingrese el motivo de la anulación" value=""
                                        required maxlength="64">
                                    <div class="invalid-feedback">
                                        El motivo de la anulación no puede estar vacío.
                                    </div>
                                </div>

                                <div class="row mt-2">
                                    <label for="descripcionAnulacion" class="form-label">
                                        Descripción del motivo de la anulación
                                    </label>
                                    <textarea class="form-control" id="descripcionAnulacion" rows="3" style="resize: none;"
                                        placeholder="Ingrese una descripción..." maxlength="512">
                                    </textarea>
                                </div>

                                <div class="row mt-2">
                                    <label for="archivosRespaldo" class="form-label">
                                        Subir archivo de respaldo
                                    </label>
                                    <input type="file" class="form-control" id="archivosRespaldo">
                                </div>

                                <div class="row mt-2">
                                    <label for="contrasenia" class="form-label">
                                        Contraseña *
                                    </label>
```

```
                                    <input type="password" class="form-control" id="contrasenia"
                                        placeholder="Ingrese su contraseña" pattern="" required>
                                    <div class="invalid-feedback">
                                        Contraseña incorrecta.
                                    </div>
                                </div>
                            </div>
                        </div>
                    </form>
                    <div class="modal-footer d-flex justify-content-center">
                        <!-- Botones para confirmar o cancelar anulación -->
                        <button type="button" class="btn btn-secondary w-25 mx-8"
                            data-bs-dismiss="modal" onclick="resetModalAnular()">No</button>
                        <button type="button" class="btn btn-danger w-25 mx-8"
                            onclick="anularEvento()">Sí</button>
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>

<!-- Enlaces a recursos externos y scripts -->
<link href="https://cdn.datatables.net/1.13.6/css/dataTables.bootstrap5.min.css" rel="stylesheet">
<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.7.1/jquery.min.js"></script>
<script src="https://cdn.datatables.net/1.13.6/js/jquery.dataTables.min.js"></script>
<script src="https://cdn.datatables.net/1.13.6/js/dataTables.bootstrap5.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/moment.js/2.29.2/moment.min.js"></script>
<script src="{{ asset('js/cancelarEvento.js') }}" defer></script>
@endsection
```

7.2.3.3. Interfaz de usuario de cancelar evento

La interfaz de usuario destinada a la cancelación de eventos ha sido diseñada de manera intuitiva y eficiente, proporcionando una experiencia clara y consciente para llevar a cabo estas acciones. Aquí se describen los elementos clave de esta interfaz:

Tabla de Eventos:

La tabla presenta una visión general de los eventos existentes con las siguientes columnas:

Nombre del Evento: Identifica de manera única cada evento.

Tipo de Evento: Muestra el tipo al que pertenece el evento.

Fecha de Inicio y Fin: Indica el período durante el cual se llevará a cabo el evento.

Acción (Cancelar): Contiene botones con las etiquetas "Cancelar" que, al hacer clic, activa el respectivo modal de confirmación.

Modal de Cancelación:

Al hacer clic en el botón "Cancelar", se abre un modal con el mensaje "¿Está seguro de cancelar este evento?".

El modal presenta dos opciones: "No" para cancelar la acción y "Sí" para confirmar la cancelación del evento.

Incluye un campo para ingresar el motivo de la cancelación.

Interacción del Usuario:

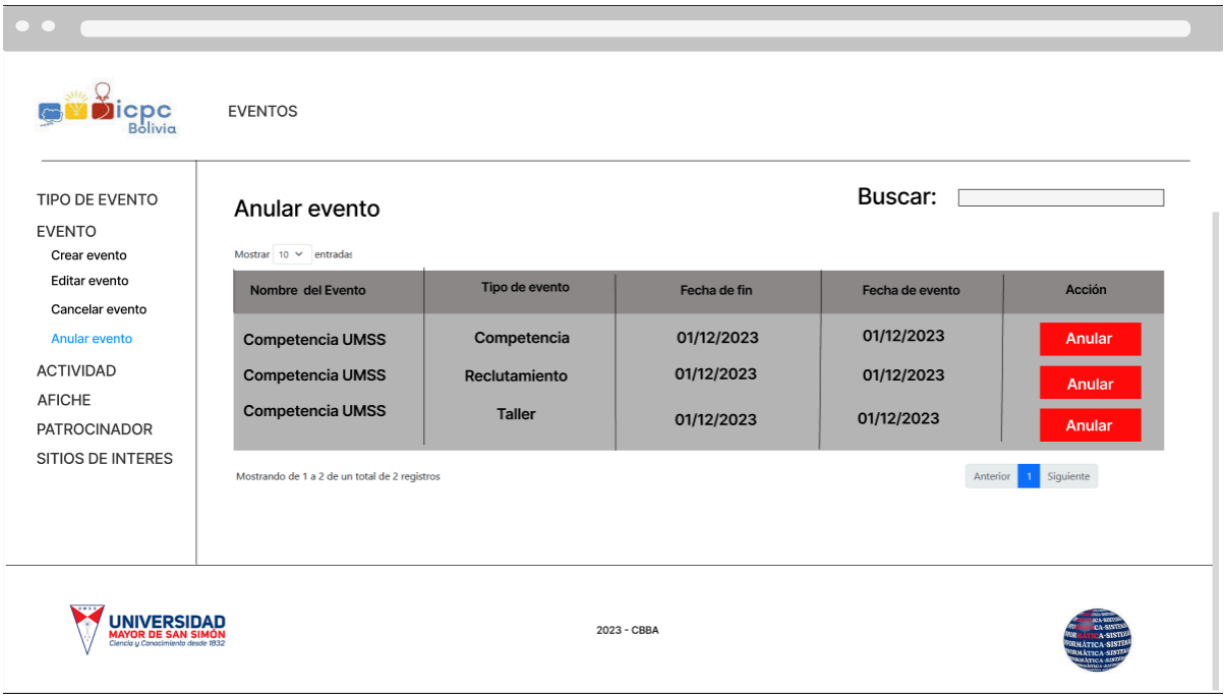
Los botones "Cancelar" en la tabla permite al usuario iniciar el proceso de cancelación con un solo clic.

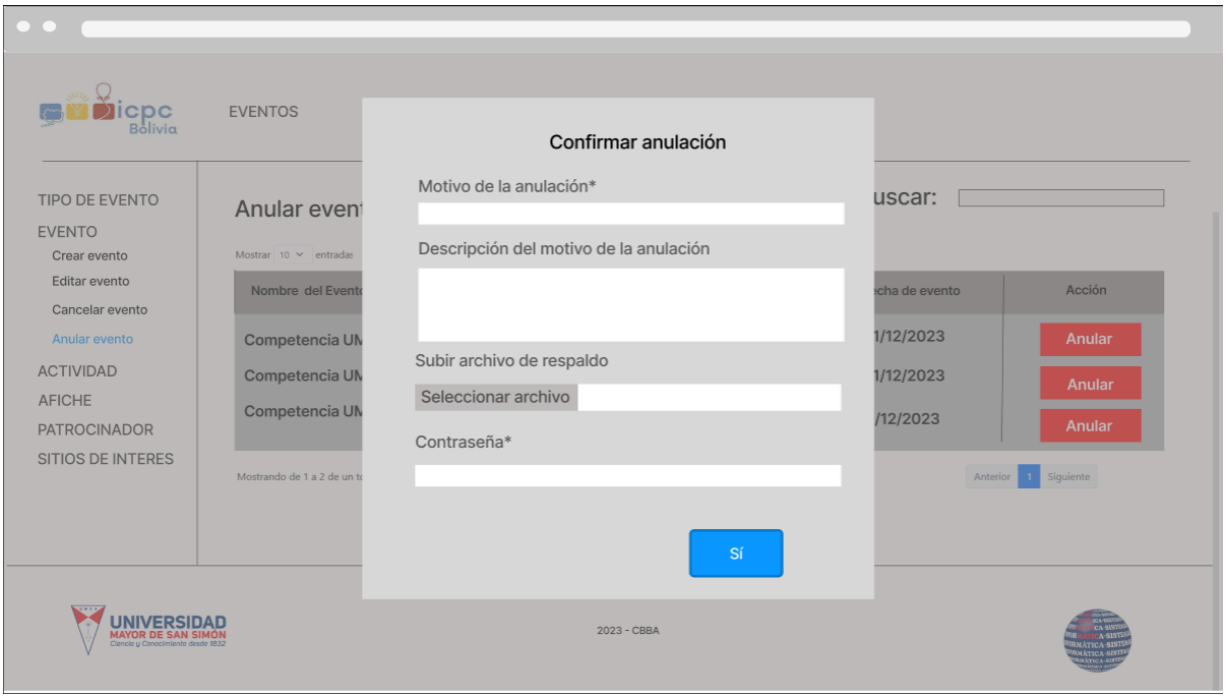
Los modales de confirmación aseguran que el usuario confirme su intención, evitando acciones accidentales.
La inclusión de campos específicos en los modales garantiza la recopilación de información relevante para la cancelación o anulación del evento.

7.2.4. Anular evento

7.2.4.1. Historia de usuario de anular evento

Historia de Usuario	
Título: Anular evento	**ID: 13**
Estimación: 8	**Importancia: Baja**
Descripción Como usuario con permisos para anular eventos, quiero poder anular un evento de manera eficiente para reflejar cambios inesperados o situaciones que lo requieran.	
Mockups	





Criterios de Aceptación

1. Al hacer clic en la sección “EVENTO” en el menú lateral y seleccionar la opción “Anular evento” se despliega una tabla con los eventos creados.
2. La tabla contiene los campos “Nombre del evento”, “Tipo de evento”, “Fecha de inicio del evento”, “Fecha de fin del evento”,”Acción”.

3. En la tabla se muestran solo los eventos que están en curso.
4. La columna “Acción” de la tabla, contiene el botón “Anular” para cada evento.
5. Al hacer clic en el botón “Anular” ,se muestra un modal con un formulario para anular un evento.
6. El formulario contiene los siguientes campos “Motivo de la anulación *”, “Descripción del motivo de la anulación”, “Subir archivo de respaldo”, “Contraseña *” y los botones “No” y “Sí”.
7. El campo “Motivo de la anulación *” es obligatorio.
8. El campo “Motivo de la anulación *” acepta un máximo de 64 caracteres.
9. El campo “Descripción del motivo de la anulación” es opcional.
10. El campo “Descripción del motivo de la anulación” acepta un máximo de 512 caracteres.
11. El campo “Subir archivo de respaldo” es un fichero de entrada.
12. El campo “Subir archivo de respaldo” es opcional.
13. Al hacer clic en el botón “Seleccionar archivo” dentro del campo “Subir archivo de respaldo”, se muestra el administrador de archivos del sistema operativo.
14. El campo “Contraseña *” es obligatorio.
15. El valor del campo “Contraseña *” está oculto con símbolos de puntos.
16. Si los campos en el formulario están correctamente llenados,al hacer clic en el botón “Si” se anula el evento.
17. Al anular un evento, aparece una alerta temporal con el mensaje “Anulado exitosamente”.
18. Al hacer clic en el botón “No” se cierra el modal y no se efectúa ningún cambio.

7.2.4.2. Código fuente de anular evento

Controlador de anular evento

Inicia recuperando el evento mediante su identificador único en la base de datos y procede a actualizar su estado, marcándolo como "Anulado". Posteriormente, crea una instancia de la clase Anulado para almacenar información detallada sobre la anulación, como el motivo y una descripción proporcionados en la solicitud. Además, la función gestiona la subida de archivos de respaldo mediante un método llamado subirRespaldo() y almacena el nombre del archivo correspondiente en el campo "archivos". Finalmente, se guarda la información relacionada con la anulación en la base de datos y se responde con un mensaje de éxito en formato JSON, indicando que el proceso de anulación se ha completado satisfactoriamente.

Ruta del archivo: app/Http/Controllers/EventoController.php

```php
/**
 * Anula un evento y guarda la información relacionada en la base de datos.
 */
public function anular(Request $request, $id)
{
    // Buscar el evento por su ID en la base de datos.
    $evento = Evento::find($id);

    // Actualizar el estado del evento a "Anulado" (estado 2).
    $evento->estado = 2;
    $evento->save();

    // Crear una nueva instancia de Anulado para almacenar información adicional.
    $anulado = new Anulado();

    // Asignar el motivo de la anulación desde la solicitud.
    $anulado->motivo = $request->motivo;

    // Asignar la descripción de la anulación desde la solicitud.
    $anulado->descripcion = $request->descripcion;

    // Subir y asignar los archivos de respaldo, gestionando la lógica en el método subirRespaldo().
    $anulado->archivos = $this->subirRespaldo($request);

    // Asignar el ID del evento anulado.
    $anulado->id_evento = $id;

    // Guardar la información de la anulación en la base de datos.
    $anulado->save();

    // Responder con un mensaje de éxito en formato JSON.
    return response()->json(['mensaje' => 'Anulado exitosamente', 'error' => false]);
}
```

Código JavaScript de anular evento

La función está diseñada para llevar a cabo la anulación de un evento. Inicia recuperando el evento específico de la base de datos utilizando su identificador único. Luego, actualiza el estado del evento a "Anulado" mediante la modificación del campo "estado" en el registro correspondiente de la tabla de eventos. A continuación, crea una instancia del modelo Anulado para registrar información adicional sobre la anulación, como el motivo proporcionado en la solicitud. También, gestionará la subida de archivos de respaldo utilizando el método subirRespaldo(). La función procede a almacenar la información de la anulación, incluyendo el motivo y, si es proporcionado, una descripción y archivos de respaldo, en la base de datos. Finalmente, responde con un mensaje en formato JSON indicando que la anulación se ha llevado a cabo con éxito. Este código demuestra una lógica de anulación que incluye la actualización del estado del evento y el registro de detalles específicos relacionados con la anulación.

Ruta del archivo: public/js/cancelarEvento.js

```
// Definición de variables globales
let tablaDeTipos;
let tablaInicializada = false;
let idEvento = null;
let contraseña = "admin123";

// Configuración de opciones para la tabla DataTable
const dataTableOptions = {
    pageLength: 10,
    lengthMenu: [5, 10, 15, 20],
    destroy: true,
    order: [[3, "desc"]],
    language: {
        // Configuración de idioma para la tabla
        // ... (se incluyen mensajes en español)
    },
};

// Evento que se ejecuta cuando la página se carga
window.addEventListener("load", () => {
    // Inicialización de la tabla DataTable
    initDataTable();
    // Configuración del patrón de la contraseña en un campo específico
    document.getElementById("contrasenia").pattern = contraseña;
});

// Función para inicializar la tabla DataTable
const initDataTable = async () => {
    // Verificación y destrucción de la tabla si ya está inicializada
    if (tablaInicializada) {
        tablaDeTipos.destroy();
    }
    // Inicialización de la tabla DataTable con opciones específicas
    DataTable.datetime("DD-MM-YYYY");
    tablaDeTipos = $("#tablaEvento").DataTable(dataTableOptions);
    tablaInicializada = true;
};
```

```
// Función para establecer el ID del evento seleccionado
const setEventoId = (id, anular) => {
    idEvento = id;
    // Reinicio de formularios modales según la acción (anular o cancelar)
    if (anular)
        resetModalAnular();
    else
        resetModalCancelar();
};

// Función para cancelar un evento
const cancelarEvento = async () => {
    // Creación del objeto FormData con los datos del formulario de cancelación
    let formData = crearFormDataCancelar();
    // Invocación de la función para realizar la solicitud de cancelación
    guardarForm(
        `/api/evento/cancelar/${idEvento}`,
        formData,
        "Error al cancelar el evento"
    );
    // Reinicio del formulario modal de cancelación
    resetModalCancelar();
    // Recarga de la página de eventos después de un tiempo
    recargarEventos(0);
};

// Función para crear un objeto FormData con datos para cancelar un evento
const crearFormDataCancelar = () => {
    const formData = new FormData();
    formData.append(
        "motivo",
        document.getElementById("motivoCancelacion").value
    );
    return formData;
};
```

```
// Función para anular un evento
const anularEvento = async () => {
    // Creación del objeto FormData con los datos del formulario de anulación
    let formData = crearFormDataAnular();
    // Validación del formulario de anulación antes de la solicitud
    if (validarAnulacion()) {
        // Invocación de la función para realizar la solicitud de anulación
        guardarForm(
            `/api/evento/anular/${idEvento}`,
            formData,
            "Error al anular el evento"
        );
        // Reinicio del formulario modal de anulación
        resetModalAnular();
        // Recarga de la página de eventos después de un tiempo
        recargarEventos(1);
    }
};

// Función para crear un objeto FormData con datos para anular un evento
const crearFormDataAnular = () => {
    const formData = new FormData();
    formData.append("motivo", document.getElementById("motivoAnulacion").value);
    formData.append(
        "descripcion",
        document.getElementById("descripcionAnulacion").value
    );
    formData.append(
        "archivos",
        document.getElementById("archivosRespaldo").files[0]
    );
    return formData;
};
```

```
// Función para validar el formulario de anulación
const validarAnulacion = () => {
    let form = document.getElementById("formularioAnulacion");
    if (form.checkValidity()) {
        form.classList.remove("was-validated");
        return true;
    }
    form.classList.add("was-validated");
    return false;
};

// Función para realizar una solicitud y manejar la respuesta
const guardarForm = async (ruta, formData, msgError) => {
    await axios
        .post(ruta, formData)
        .then((response) => {
            // Mostrar alerta con el resultado de la solicitud
            mostrarAlerta(
                "Éxito",
                response.data.mensaje,
                response.error ? "danger" : "success"
            );
        })
        .catch((error) => {
            // Mostrar alerta en caso de error en la solicitud
            mostrarAlerta("Fracaso", msgError, "danger");
        });
};

// Función para reiniciar el formulario modal de cancelación
const resetModalCancelar = () => {
    $("#modalCancelar").modal("hide");
    document.getElementById("motivoCancelacion").value = "";
};

// Función para reiniciar el formulario modal de anulación
const resetModalAnular = () => {
    $("#modalAnular").modal("hide");
    document.getElementById("formularioAnulacion").classList.remove("was-validated");
    document.getElementById("motivoAnulacion").value = "";
    document.getElementById("descripcionAnulacion").value = "";
    document.getElementById("archivosRespaldo").value = "";
    document.getElementById("contrasenia").value = "";
};

// Función para recargar la página de eventos después de un tiempo
const recargarEventos = (control) => {
    let ruta = control === 0 ? "/admin/eventos/cancelar-evento" : "/admin/eventos/anular-evento";
    setTimeout(() => {
        window.location.href = ruta;
    }, 1800);
};
```

Vista de anular evento

La vista presenta una tabla que muestra una lista de eventos, con columnas que incluyen el nombre del evento, el tipo de evento, la fecha de inicio y la fecha de fin. Además, hay una columna de acción con botones para anular o cancelar cada evento, según el contexto

definido por la variable $anular. El diseño garantiza que solo se muestren los eventos elegibles para la acción seleccionada. Para anular un evento, el usuario puede hacer clic en el botón "Anular", lo que activará un modal de confirmación. Este modal solicita al usuario ingresar el motivo de la anulación, una descripción opcional y proporcionar una contraseña para confirmar la acción. También permite adjuntar archivos de respaldo relacionados con la anulación. La vista utiliza DataTables, una biblioteca de jQuery, para mejorar la funcionalidad de la tabla, permitiendo la paginación y búsqueda de eventos.

Ruta del archivo: resources/views/eventos/cancelarEvento.blade.php

```
@extends('layouts.app')

@section('content')
    <div class="container">
        <div class="row mb-2">
            <h2>{{ $anular ? 'Anular' : 'Cancelar' }} evento</h2>
        </div>
        <div class="row g-5">
            <div class="col-sm-12">
                <!-- Tabla para mostrar eventos -->
                <table class="table table-responsive table-striped text-secondary  cursor" id="tablaEvento">
                    <caption>Eventos</caption>
                    <thead>
                        <tr>
                            <th scope="col" class="col-sm-3 col-md-3">Nombre del evento</th>
                            <th scope="col" class="col-sm-2 col-md-2 text-center">Tipo de evento</th>
                            <th scope="col" class="col-sm-3 col-md-3 text-center">Fecha de inicio del evento</th>
                            <th scope="col" class="col-sm-3 col-md-3 text-center">Fecha de fin del evento</th>
                            <th scope="col" class="col-sm-1 col-md-1 text-center font-sm">Acción</th>
                        </tr>
                    </thead>
                    <tbody id="datosTabla">
                        @php
                            date_default_timezone_set('America/Caracas');
                        @endphp
                        @foreach ($eventos as $evento)
                            <!-- Verifica si se está anulando o cancelando y si el evento es válido -->
                            @if ($anular)
                                @if (strtotime($evento->inicio_evento) <= time() && strtotime($evento->fin_evento) >= time())
```

```
                        <!-- Fila de la tabla con información del evento -->
                        <tr id="{{ $evento->id }}">
                            <td>{{ $evento->nombre }}</td>
                            <td class="text-center">{{ $evento->tipoEvento->nombre }}</td>
                            <td class="text-center">{{ date('d-m-Y', strtotime($evento->inicio_evento)) }}</td>
                            <td class="text-center">{{ date('d-m-Y', strtotime($evento->fin_evento)) }}</td>
                            <td class="text-center">
                                <!-- Botón para anular el evento -->
                                <button type="button" class="btn btn-danger btn-sm"
                                    onclick="setEventoId({{ $evento->id }},true)" id="botonAccion"
                                    style="min-width: 80px;" data-bs-toggle="modal"
                                    data-bs-target={{ $anular ? '#modalAnular' : '#modalCancelar' }}>
                                    {{ $anular ? 'Anular' : 'Cancelar' }}
                                </button>
                            </td>
                        </tr>
                    @endif
                @else
                    <!-- Verifica si el evento es válido para cancelación -->
                    @if (strtotime($evento->inicio_evento) >= time() && $evento->inscritos->count() == 0)
                        <!-- Fila de la tabla con información del evento -->
                        <tr id="{{ $evento->id }}">
                            <td>{{ $evento->nombre }}</td>
                            <td class="text-center">{{ $evento->tipoEvento->nombre }}</td>
                            <td class="text-center">{{ date('d-m-Y', strtotime($evento->inicio_evento)) }}</td>
                            <td class="text-center">{{ date('d-m-Y', strtotime($evento->fin_evento)) }}</td>
                            <td class="text-center">
                                <!-- Botón para cancelar el evento -->
                                <button type="button" class="btn btn-danger btn-sm"
                                    onclick="setEventoId({{ $evento->id }}, false)"
                                    id="botonAccion" style="min-width: 80px;" data-bs-toggle="modal"
                                    data-bs-target={{ $anular ? '#modalAnular' : '#modalCancelar' }}>
                                    {{ $anular ? 'Anular' : 'Cancelar' }}
                                </button>
                            </td>
                        </tr>
                    @endif
                @endif
            @endforeach
        </tbody>
    </table>

    <!-- Modal para cancelar evento -->
    <div class="modal fade" id="modalCancelar" tabindex="-1" aria-labelledby="modalCancelarLabel"
        aria-hidden="true">
        <div class="modal-dialog">
            <div class="modal-content">
                <div class="modal-header">
                    <h5 class="modal-title" id="modalCancelarLabel">
                        Confirmar cancelación
                    </h5>
                </div>
                <div class="modal-body">
                    <div class="container px-4">
                        <!-- Formulario para ingresar motivo de cancelación -->
                        <label for="motivoCancelacion" class="form-label">
                            Motivo de la cancelación
                        </label>
                        <textarea rows="3" type="text" class="form-control" id="motivoCancelacion"
                            placeholder="Ingrese el motivo de la cancelación" value=""
```

```
                                maxlength="128" style="resize: none"></textarea>
                        </div>
                    </div>
                    <div class="modal-footer d-flex justify-content-center">
                        <!-- Botones para confirmar o cancelar cancelación -->
                        <button type="button" class="btn btn-secondary w-25 mx-8"
                            data-bs-dismiss="modal" onclick="resetModalCancelar()">No</button>
                        <button type="button" class="btn btn-danger w-25 mx-8"
                            onclick="cancelarEvento()">Sí</button>
                    </div>
                </div>
            </div>
        </div>

        <!-- Modal para anular evento -->
        <div class="modal fade" id="modalAnular" tabindex="-1" aria-labelledby="modalAnularLabel"
            aria-hidden="true">
            <div class="modal-dialog">
                <div class="modal-content">
                    <div class="modal-header">
                        <h5 class="modal-title" id="modalAnularLabel">
                            Confirmar anulación
                        </h5>
                    </div>
                    <form class="needs-validation" novalidate id="formularioAnulacion">
                        <div class="modal-body">
                            <div class="container px-5">
                                <!-- Campos para ingresar motivo, descripción, archivo de respaldo y contraseña -->
                                <div class="row">
                                    <label for="motivoAnulacion" class="form-label">
                                        Motivo de la anulación *
                                    </label>
                                    <input type="text" class="form-control" id="motivoAnulacion"
                                        placeholder="Ingrese el motivo de la anulación" value=""
                                        required maxlength="64">
                                    <div class="invalid-feedback">
                                        El motivo de la anulación no puede estar vacío.
                                    </div>
                                </div>

                                <div class="row mt-2">
                                    <label for="descripcionAnulacion" class="form-label">
                                        Descripción del motivo de la anulación
                                    </label>
                                    <textarea class="form-control" id="descripcionAnulacion" rows="3" style="resize: none;"
                                        placeholder="Ingrese una descripción..." maxlength="512">
                                    </textarea>
                                </div>

                                <div class="row mt-2">
                                    <label for="archivosRespaldo" class="form-label">
                                        Subir archivo de respaldo
                                    </label>
                                    <input type="file" class="form-control" id="archivosRespaldo">
                                </div>

                                <div class="row mt-2">
                                    <label for="contrasenia" class="form-label">
                                        Contraseña *
                                    </label>
```

```
                                    <input type="password" class="form-control" id="contrasenia"
                                        placeholder="Ingrese su contraseña" pattern="" required>
                                    <div class="invalid-feedback">
                                        Contraseña incorrecta.
                                    </div>
                                </div>
                            </div>
                        </div>
                    </form>
                    <div class="modal-footer d-flex justify-content-center">
                        <!-- Botones para confirmar o cancelar anulación -->
                        <button type="button" class="btn btn-secondary w-25 mx-8"
                            data-bs-dismiss="modal" onclick="resetModalAnular()">No</button>
                        <button type="button" class="btn btn-danger w-25 mx-8"
                            onclick="anularEvento()">Sí</button>
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>

<!-- Enlaces a recursos externos y scripts -->
<link href="https://cdn.datatables.net/1.13.6/css/dataTables.bootstrap5.min.css" rel="stylesheet">
<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.7.1/jquery.min.js"></script>
<script src="https://cdn.datatables.net/1.13.6/js/jquery.dataTables.min.js"></script>
<script src="https://cdn.datatables.net/1.13.6/js/dataTables.bootstrap5.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/moment.js/2.29.2/moment.min.js"></script>
<script src="{{ asset('js/cancelarEvento.js') }}" defer></script>
@endsection
```

7.2.4.3. Interfaz de usuario de anular evento

La interfaz de usuario destinada a la anulación de eventos ofrece un entorno claro y eficiente para llevar a cabo esta acción de manera deliberada. A continuación, se detallan los elementos clave de la interfaz:

Tabla de Eventos:

La tabla exhibe una lista de eventos con columnas informativas, como el nombre del evento, el tipo de evento, la fecha de inicio y la fecha de fin. La columna "Acción" contiene botones identificados como "Anular" para los eventos seleccionados según la lógica establecida por la variable $anular.

Modal de Anulación:

Al hacer clic en el botón "Anular" de un evento específico, se desencadena un modal de confirmación. Este modal solicita al usuario que ingrese el motivo de la anulación, proporcionando una opción para agregar una descripción detallada y adjuntar archivos de respaldo relacionados con la anulación. También se requiere la contraseña del usuario como medida adicional de seguridad.

Interacción del Usuario:

El botón "Anular" en la tabla permite al usuario iniciar el proceso de anulación con un solo clic.

El modal de confirmación asegura que el usuario confirme su intención de anular el evento, evitando acciones accidentales.

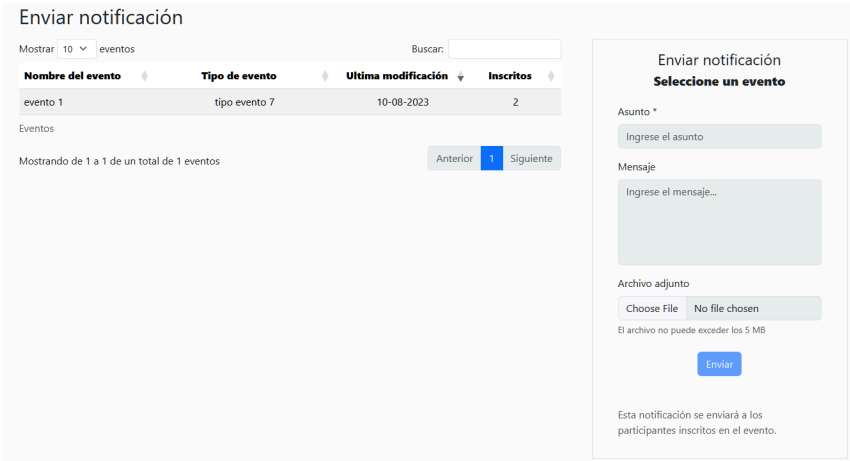
La interfaz prioriza la claridad al presentar la información y guía al usuario a través de los pasos necesarios para completar la anulación de manera efectiva.

7.2.5. Notificar evento

7.2.5.1. Historia de usuario de notificar evento

Historia de Usuario	
Título: Envío de notificaciones de evento	**ID: 30**
Estimación: 8	**Importancia: Media**
Descripción: Como usuario con privilegios de enviar notificaciones de evento, quiero enviar notificaciones personalizadas a todos los participantes inscritos en un evento.	

MockUps



Criterios de aceptación

1. Al hacer clic en la sección “EVENTO” en el menú lateral y seleccionar la opción “Notificar evento” se muestra una tabla con los eventos que tienen participantes inscritos.
2. La tabla tiene las siguientes columnas “Nombre del evento”, “Tipo de evento”, “Ultima modificación”, “Inscritos”.
3. La columna “Inscritos” de la tabla, muestra la cantidad de participantes inscritos que tiene cada uno de los eventos.
4. Al hacer clic sobre un evento se habilita el formulario “Enviar notificación” con los siguiente campos: “Asunto *”, “Mensaje”, “Archivo adjunto”, y el botón “Enviar”.
5. El campo “Asunto *”, es obligatorio.
6. El tamaño máximo del archivo admitido en el campo “Archivo adjunto” es 5 Mb.
7. El campo “Archivo adjunto”, solo admite un archivo.
8. El campo “Archivo adjunto”, admite cualquier formato de archivo.
9. Al hacer clic en el botón “Enviar”, Si el campo “Asunto *” está vacío, sale un mensaje de validación “El asunto no puede estar vacío”.
10. Al hacer clic en el botón “Enviar”, sale un modal “Confirmación”, con el mensaje “¿Está seguro de enviar la notificación?”, y los botones “No” y “Sí”.
11. Al hacer clic en el botón “No”, los campos regresan a los valores por defecto.
12. Al hacer clic en el botón “No”, el modal se cierra.
13. Al hacer clic en el botón “Sí”, sale una alerta temporal con el mensaje “Notificación enviada correctamente”.
14. Al hacer clic en el botón “Sí”, los campos del formulario vuelven a los valores por defecto.
15. Al hacer clic en el botón “Sí”, el modal se cierra.

7.2.5.2. Código fuente de notificar evento

Controlador de notificar evento

El método tablaEventos prepara la vista para enviar notificaciones, obteniendo información detallada sobre los eventos disponibles. La función storeArchivo almacena archivos adjuntos, y getParticipantes y getEquipos recuperan participantes y equipos inscritos en un evento, respectivamente. El método principal, notificar, procesa la solicitud de notificación, almacenando archivos adjuntos si existen, preparando los datos necesarios y enviando notificaciones a todos los participantes y equipos del evento. Se incluyen manejo de errores y registros de log para una gestión robusta.

Ruta del archivo: app/Http/Controllers/NotificacionController.php

```php
<?php

namespace App\Http\Controllers;

use App\Models\Equipo;
use App\Models\Evento;
use App\Models\Participante;
use Illuminate\Http\Request;
use App\Notifications\NotificacionEvento;
use Illuminate\Support\Facades\Log;

class NotificacionController extends Controller
{
    /**
     * Muestra la vista para enviar notificaciones con la lista de eventos disponibles.
     */
    public function tablaEventos()
    {
        // Selecciona los eventos activos con información relacionada.
        $eventos = Evento::select(
            'id',
            'nombre',
            'updated_at',
            'id_tipo_evento'
        )
            ->with(['tipoEvento' => function ($q) {
                $q->select('id', 'nombre')
                    ->withTrashed();
            }])
            ->withCount(['inscritos as cantidad_inscritos' => function ($q) {
                $q->whereHas('participante', function ($q) {
                    $q->where('correo_confirmado', 1);
                });
            }])
            ->withCount(['equiposInscrito as cantidad_equipos' => function ($q) {
                $q->whereHas('equipos', function ($q) {
                    $q->where('correo_verificado', 1);
                });
            }])
```

```
            ->where('estado', 0)
            ->orderBy('eventos.updated_at', 'desc')
            ->get();

        return view('notificaciones.enviar', ['eventos' => $eventos]);
    }
    /**
     * Almacena el archivo adjunto y devuelve su ruta.
     */
    public function storeArchivo(Request $request)
    {
        try {
            // Verifica si se recibió un archivo.
            if ($request->hasFile('archivo')) {
                $archivo = $request->file('archivo');
                $nombreArchivo = $archivo->getClientOriginalName();
                $ruta = $archivo->storeAs('public/archivo_adjunto', $nombreArchivo);
                $urlArchivo = $ruta;
                return $urlArchivo;
            } else {
                return null;
            }
        } catch (\Throwable $e) {
            // Captura cualquier excepción y la registra en los logs.
            return $e->getMessage();
        }
    }
    /**
     * Obtiene los participantes inscritos en un evento.
     */
    public function getParticipantes($id_evento)
    {
        // Obtiene participantes que están inscritos en el evento y han confirmado su correo.
        $participantes = Participante::whereHas('inscritos', function ($q) use ($id_evento) {
            $q->where('id_evento', $id_evento);
        })->where('correo_confirmado', 1)->get();

        return $participantes;
    }
```

```
    /**
     * Envía notificaciones a participantes y equipos de un evento.
     */
    public function notificar(Request $request)
    {
        try {
            // Almacena la ruta del archivo adjunto, si se proporciona.
            $adjunto = $request->hasFile('archivo') ? $this->storeArchivo($request) : null;

            // Prepara los datos para la notificación.
            $datos = [
                'asunto' => $request->asunto,
                'mensaje' => $request->mensaje,
                'adjunto' => $adjunto,
                'id_evento' => $request->id_evento,
                'nombre' => $request->nombre
            ];

            // Obtiene participantes y equipos del evento.
            $participantes = $this->getParticipantes($request->id_evento);
            $equipos = $this->getEquipos($request->id_evento);

            // Notifica a cada equipo del evento.
            foreach ($equipos as $equipo) {
                $equipo->notify(new NotificacionEvento($datos));
            }

            // Notifica a cada participante del evento.
            foreach ($participantes as $participante) {
                $participante->notify(new NotificacionEvento($datos));
            }

            return response()->json(['mensaje' => 'Notificación enviada correctamente', 'error' =>
            false]);
        } catch (\Exception $e) {
            // Registra un mensaje de error en los logs.
            Log::error('Error al enviar la notificación: ' . $e->getMessage());
            return 'Error al enviar la notificación.';
        }
    }

    /**
     * Obtiene los equipos inscritos en un evento.
     */
    public function getEquipos($id_evento)
    {
        // Obtiene equipos que están inscritos en el evento y han verificado su correo.
        $equipos = Equipo::whereHas('equipoInscrito', function ($q) use ($id_evento) {
            $q->where('id_evento', $id_evento);
        })->where('correo_verificado', 1)->get();

        return $equipos;
    }
}
```

Código JavaScript de notificar evento

Se hace uso de la librería DataTable para la presentación tabular de eventos, y Axios para gestionar las peticiones HTTP a la API correspondiente. La lógica se organiza en funciones claramente definidas: La inicialización de la DataTable se realiza al cargar la página, con opciones de configuración como el número de registros por página, menús de longitud y ordenamiento descendente por fecha. La función initDataTable se encarga de

inicializar o reiniciar la tabla según sea necesario. El código gestiona la habilitación y deshabilitación del formulario, asegurando que solo se pueda interactuar con él cuando se ha seleccionado un evento. Se utiliza una serie de funciones, como inhabilitarForm, habilitarForm, y seleccionarEvento, para controlar el estado del formulario y resaltar visualmente el evento seleccionado. La validación del formulario se realiza antes de mostrar un modal de confirmación para enviar la notificación. Se comprueba la validez del formulario utilizando la propiedad checkValidity y se muestra el modal si los datos son correctos. La función enviar realiza una petición POST a la API de notificaciones con los datos del formulario. Luego, se oculta el modal de confirmación y se muestra una alerta con el resultado de la operación. Finalmente, hay funciones para resetear los campos del formulario (resetInputs), manejar cambios en la selección de archivos (onchangeArchivo), y algunas variables globales para el manejo de eventos y la tabla DataTable.

Ruta del archivo: public/js/Notificacion/enviar.js

```
// Variables globales para la gestión de la tabla de eventos y su inicialización
let tablaDeTipos;
let tablaInicializada = false;

// Elementos del formulario y del evento seleccionado
const eventoSeleccionado = document.getElementById("nombreEvento");
const form = document.getElementById("formNotificacion");
const asunto = document.getElementById("asunto");
const mensaje = document.getElementById("mensaje");
const archivo = document.getElementById("archivo");

// Opciones para la configuración de la DataTable
const dataTableOptions = {
    pageLength: 10,
    lengthMenu: [5, 10, 15, 20],
    destroy: true,
    order: [2, "desc"],
    ordering: true,
    language: {
        lengthMenu: "Mostrar _MENU_ eventos",
        zeroRecords: "Ningún evento encontrado",
        info: "Mostrando de _START_ a _END_ de un total de _TOTAL_ eventos",
        infoEmpty: "Ningún evento encontrado",
        infoFiltered: "(filtrados desde _MAX_ eventos totales)",
        search: "Buscar:",
        loadingRecords: "Cargando...",
        paginate: {
            first: "Primero",
            last: "Último",
            next: "Siguiente",
            previous: "Anterior",
        },
    },
};

// Inicialización de la DataTable al cargar la página
const initDataTable = async () => {
    if (tablaInicializada) {
        tablaDeTipos.destroy();
    }
```

```
    DataTable.datetime("DD-MM-YYYY");
    tablaDeTipos = $("#tablaEvento").DataTable(dataTableOptions);
    tablaInicializada = true;
};

// Evento de carga de la página que inicia la DataTable
window.addEventListener("load", () => {
    initDataTable();
    // Si no hay un evento seleccionado, se inhabilita el formulario
    if (!seleccionado) {
        inhabilitarForm();
        eventoSeleccionado.textContent = "Seleccione un evento";
    }
});

// Función para inhabilitar los campos del formulario
const inhabilitarForm = () => {
    asunto.disabled = true;
    mensaje.disabled = true;
    archivo.disabled = true;
    document.getElementById("botonEnviar").disabled = true;
};

// Función para habilitar los campos del formulario
const habilitarForm = () => {
    asunto.disabled = false;
    mensaje.disabled = false;
    archivo.disabled = false;
    document.getElementById("botonEnviar").disabled = false;
};

// Variables para el evento seleccionado y su ID
let seleccionado;
let idSeleccionado;

// Función para seleccionar un evento
const seleccionarEvento = async (id, nombre) => {
    // Si ya hay un evento seleccionado, se quita la clase de resaltado
    if (seleccionado) {
        seleccionado.classList.remove("table-primary");
```

```
    }
    // Se habilita el formulario y se resalta el evento seleccionado
    habilitarForm();
    seleccionado = document.getElementById(id);
    seleccionado.classList.add("table-primary");
    idSeleccionado = id;
    eventoSeleccionado.textContent = nombre;
};

// Función para validar el formulario antes de mostrar el modal de confirmación
const validarForm = () => {
    if (form.checkValidity()) {
        form.classList.remove("was-validated");
        // Se muestra el modal de confirmación
        $("#modalEnviarNotificacion").modal("show");
    }
    form.classList.add("was-validated");
};

// Función para enviar la notificación al hacer clic en el botón de confirmación
const enviar = async () => {
    // Se crea un objeto FormData con los datos del formulario
    const formulario = new FormData(form);
    formulario.append("id_evento", idSeleccionado);
    formulario.append("nombre", eventoSeleccionado.textContent);
    // Se realiza una petición POST a la API de notificaciones
    const res = await axios.post("/api/notificacion/", formulario);
    // Se oculta el modal de confirmación
    $("#modalEnviarNotificacion").modal("hide");
    // Se muestra una alerta con el resultado de la operación
    mostrarAlerta(
        res.data.error ? "Peligro" : "Éxito",
        res.data.mensaje,
        res.data.error ? "danger" : "success"
    );
    // Se resetean los campos del formulario
    resetInputs();
};
```

```
// Función para resetear los campos del formulario
const resetInputs = () => {
    form.reset();
    form.classList.remove("was-validated");

    asunto.value = "";
    mensaje.value = "";
    archivo.value = "";
};

// Función que se ejecuta al cambiar el archivo seleccionado
const onchangeArchivo = () => {
    const inputArchivo = document.getElementById("archivo");
    const archivo = inputArchivo.files[0];

    if (archivo) {
        // Se verifica el tamaño del archivo
        const maxSize = 5 * 1024 * 1024; // 5 megabytes
        if (archivo.size > maxSize) {
            // Si el archivo es demasiado grande, se muestra una alerta y se limpia el input
            mostrarAlerta(
                "danger",
                "El archivo no debe superar los 5 MB",
                "danger"
            );
            inputArchivo.value = "";
        }
    }
};
```

Vista de notificar evento

La vista está dividida en dos secciones principales: la tabla de eventos y el formulario de notificación. La tabla de eventos, construida con DataTables, presenta información relevante, como el nombre del evento, tipo de evento, última modificación y el número de inscritos o equipos. Solo se muestran eventos con participantes o equipos inscritos, permitiendo la selección de un evento específico al hacer clic en él. El formulario de notificación, ubicado en la parte derecha de la página, se habilita al seleccionar un evento. Permite al usuario ingresar un asunto, un mensaje y adjuntar un archivo. Además, se incluye una funcionalidad para validar el formulario antes de enviar la notificación.

El código hace uso de componentes Blade, como un modal de confirmación para asegurar la intención de enviar la notificación.

Ruta del archivo: resources/views/notificaciones/enviar.blade.php

```
@extends('layouts.app')

@section('content')
    <div class="container">
        <div class="row mb-2">
            <h2>Enviar notificación</h2>
        </div>
        <div class="row g-5">
            <div class="col-sm-12 col-md-12 col-lg-8">
                <!-- Tabla de Eventos -->
                <table class="table table-responsive table-striped text-secondary" id="tablaEvento">
                    <caption>Eventos</caption>
                    <thead>
                        <tr>
                            <th scope="col" class="col-sm-3 col-md-3">Nombre del evento</th>
                            <th scope="col" class="col-sm-4 col-md-4 text-center">Tipo de evento</th>
                            <th scope="col" class="col-sm-3 col-md-3 text-center">Ultima modificación</th>
                            <th scope="col" class="col-sm-3 col-md-3 text-center">Inscritos</th>
                        </tr>
                    </thead>
                    <tbody id="datosTabla">
                        @foreach ($eventos as $evento)
                            <!-- Solo mostrar eventos con participantes o equipos inscritos -->
                            @if($evento->cantidad_inscritos > 0 or $evento->cantidad_equipos > 0)
                            <tr onclick="seleccionarEvento({{ $evento->id }}, '{{ $evento->nombre }}', event)"
                                id="{{ $evento->id }}">
                                <td>{{ $evento->nombre }}</td>
                                <td class="text-center">{{ $evento->tipoEvento->nombre }}</td>
                                <td class="text-center">{{ date('d-m-Y', strtotime($evento->updated_at)) }}</td>
                                <td class="text-center" id="contadorActividades{{ $evento->id }}">
                                    {{ $evento->cantidad_inscritos > 0 ? $evento->cantidad_inscritos : $evento->cantidad_equipos }}</td>
                            </tr>
                            @endif
                        @endforeach
                    </tbody>
                </table>
            </div>

            <div class="col-sm-12 col-md-12 col-lg-4">
                <!-- Formulario de Notificación -->
                <div class="container d-flex flex-column justify-content-center align-items-center border p-3 container-image">
                    <div class="col-12 d-flex justify-content-center align-items-center">
                        <h4>Enviar notificación</h4>
                    </div>
                    <h5 id="nombreEvento" class="text-center fw-bold"></h5>
                    <div class="container g-5">
                        <form class="needs-validation" novalidate id="formNotificacion">
                            <!-- Campo de Asunto -->
                            <label for="asunto" class="form-label mt-3">Asunto *</label>
                            <input type="text" class="form-control custom-input" name="asunto"
                                placeholder="Ingrese el asunto" id="asunto" maxlength="64" required>
                            <div class="invalid-feedback">
                                El asunto no puede estar vacío
                            </div>

                            <!-- Campo de Mensaje -->
                            <label for="mensaje" class="form-label mt-3">Mensaje</label>
                            <textarea name="mensaje" id="mensaje" cols="30" rows="5" class="form-control custom-input"
                                style="resize: none" placeholder="Ingrese el mensaje..."></textarea>
```

```
                    <!-- Campo de Archivo Adjunto -->
                    <label for="archivo" class="mt-3 form-label">Archivo adjunto</label>
                    <input type="file" name="archivo" id="archivo" class="form-control custom-input"
                        onchange="onchangeArchivo()">
                    <p class="text-muted mt-1" style="font-size: 14px">
                        El archivo no puede exceder los 5 MB
                    </p>
                </form>
                <div class="d-flex justify-content-center mt-4">
                    <button class="btn btn-primary" onclick="validarForm()" id="botonEnviar">Enviar</button>
                </div>
                <p class="text-muted mt-5">
                    Esta notificación se enviará a los participantes inscritos en el evento.
                </p>
                <!-- Modal de Confirmación -->
                @component('components.modal')
                @slot('modalId', 'modalEnviarNotificacion')
                @slot('modalTitle', 'Confirmación')
                @slot('modalContent')
                    ¿Está seguro de enviar la notificación?
                @endslot
                @slot('modalButton')
                    <button type="button" class="btn btn-secondary w-25 mx-8" data-bs-dismiss="modal" onclick="resetInputs()">No</button>
                    <button type="reset" class="btn btn-primary w-25 mx-8" data-bs-dismiss="modal"
                        onclick="enviar()">Sí</button>
                @endslot
            @endcomponent

            </div>
        </div>
```

```
        </div>
    </div>
</div>

<!-- Enlaces a recursos externos -->
<link href="https://cdn.datatables.net/1.13.6/css/dataTables.bootstrap5.min.css" rel="stylesheet">
<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.7.1/jquery.min.js"></script>
<script src="https://cdn.datatables.net/1.13.6/js/jquery.dataTables.min.js"></script>
<script src="https://cdn.datatables.net/1.13.6/js/dataTables.bootstrap5.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/moment.js/2.29.2/moment.min.js"></script>
<script src="{{ asset('js/Notificacion/enviar.js') }}" defer></script>
@endsection
```

7.2.5.3. Interfaz de usuario de notificar evento

La interfaz de usuario destinada a la notificación de eventos ha sido diseñada de manera clara y funcional para facilitar el proceso de envío de comunicaciones a participantes y equipos. A continuación, se detallan los elementos clave de la interfaz:

Tabla de Eventos:

La tabla presenta una visión general de los eventos disponibles, incluyendo información relevante como el nombre del evento, tipo de evento, última modificación y el número de inscritos o equipos. Solo se muestran eventos con participantes o equipos inscritos. Al hacer clic en un evento específico, se resalta y habilita el formulario de notificación.

Formulario de Notificación:

Ubicado en la parte derecha de la página, este formulario se activa al seleccionar un evento en la tabla. Los elementos incluyen:

Asunto: Campo de texto para ingresar el asunto de la notificación, con una validación que asegura que no esté vacío.

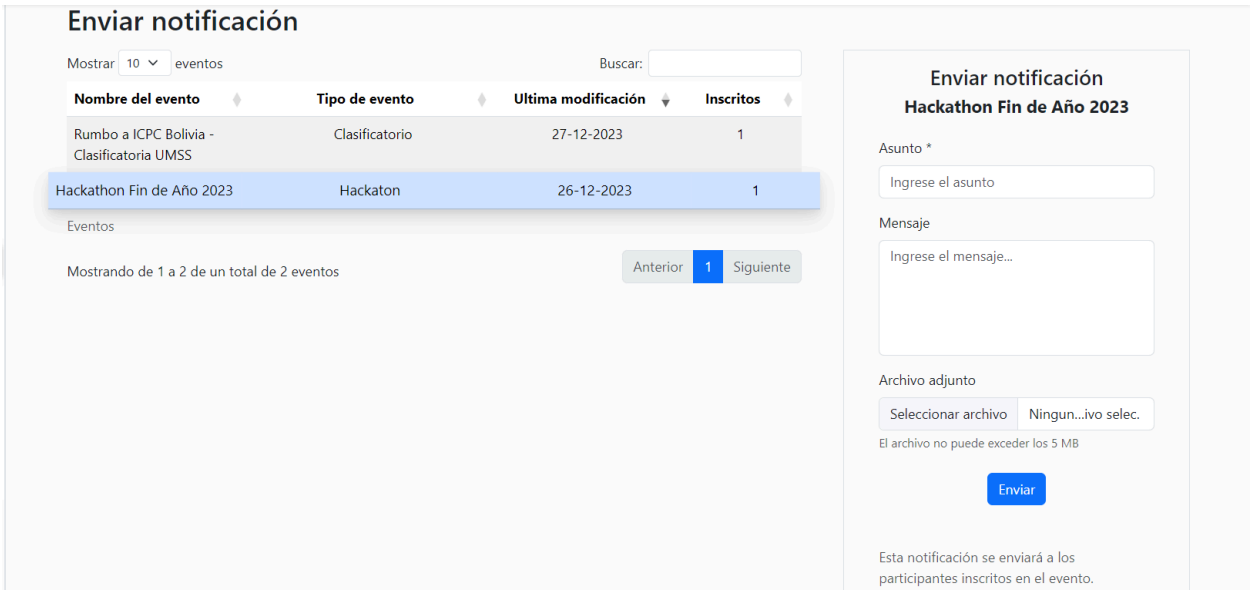
Mensaje: Área de texto para redactar el mensaje de la notificación.

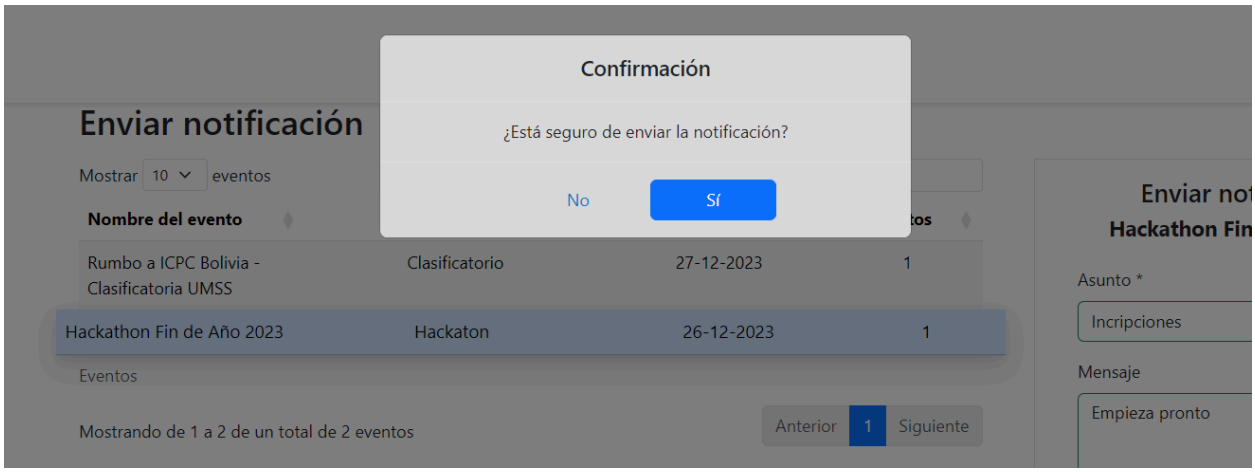
Archivo Adjunto: Permite adjuntar un archivo, con una restricción de tamaño máximo de 5 MB.

Botón de Enviar: Botón que desencadena la validación del formulario y muestra un modal de confirmación antes de enviar la notificación.

Modal de Confirmación:

Al hacer clic en "Enviar", se activa un modal que pregunta al usuario si está seguro de enviar la notificación. Este modal proporciona botones "Sí" y "No" para confirmar o cancelar la operación, respectivamente.

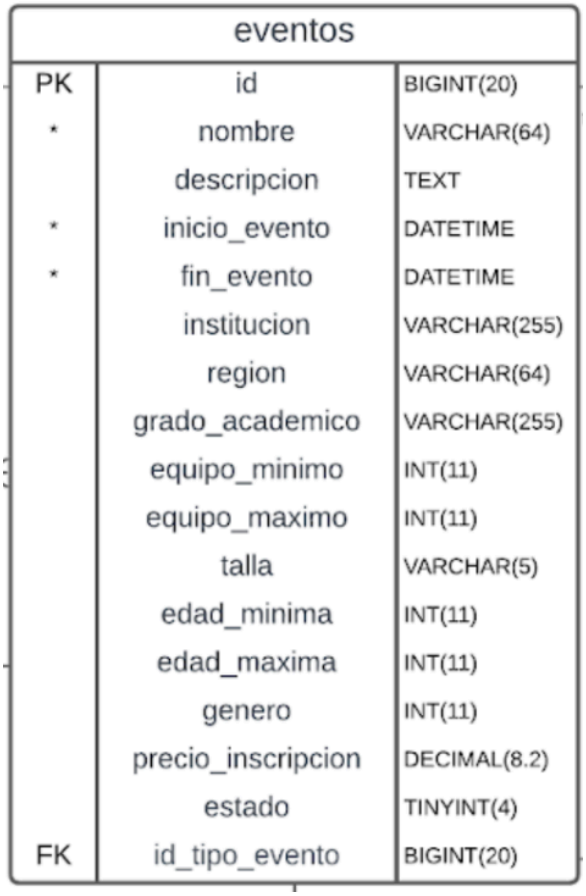




7.2.6. Tabla de base de datos de eventos

La tabla 'eventos' es central en nuestra base de datos, almacenando detalles cruciales sobre cada evento. Con campos como 'id' para identificación única, 'nombre' para descripción concisa, 'descripcion' para información detallada, 'inicio_evento' y 'fin_evento' para fechas de inicio y finalización, y 'institucion', 'region', 'grado_academico', 'equipo_minimo', 'equipo_maximo', 'talla', 'edad_minima',

'edad_maxima', 'genero', 'precio_inscripcion', 'estado', e 'id_tipo_evento' para características específicas.



7.3. ACTIVIDAD

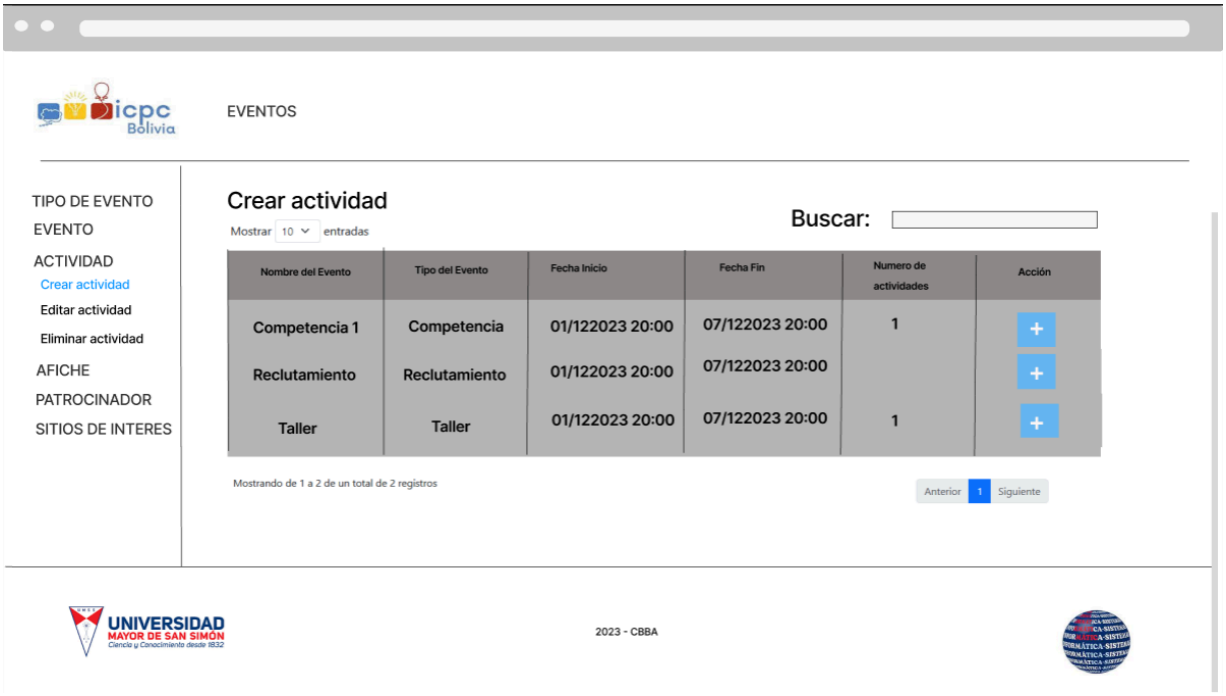
7.3.1. Crear actividad

7.3.1.1. Historia de usuario de crear actividad

Historia de Usuario	
Título: Crear actividad	**ID: 45**
Estimación: 5	**Importancia: Alta**
Descripción	

Como usuario con privilegios de creación de actividades, quiero crear una actividad en un evento para dar a los usuarios que se inscriban a un evento información adicional sobre lo que se va a realizar en el evento y en qué horario será.

Mockups



Criterios de Aceptación

1. Al hacer clic en la sección “ACTIVIDAD” en el menú lateral y seleccionar la opción “Crear actividad” se despliega una tabla con los eventos creados que están en curso o aún no empezaron.
2. La tabla tiene las siguientes columnas: “Nombre del evento”,”Tipo de evento”,”Fecha de inicio del evento”,”Fecha de fin del evento”,”Cantidad de actividades”,”Acción”.
3. Al hacer clic en el botón “ + ” de un evento seleccionado se redirige a una vista con un formulario para crear una actividad.
4. El formulario “Crear actividad” tiene los siguientes campos: “Nombre de la actividad *”, ”Duración de la actividad *” (“Inicio”, “Fin”), “Descripción de la actividad” y los botones “Cancelar” y “Crear”.
5. El campo “Nombre de la actividad *” es obligatorio.
6. El campo “Nombre de la actividad *” es único para cada evento.
7. Los campos “Inicio” y “Fin” de Duración de la actividad son obligatorios.
8. El campo “Inicio” de Duración de la actividad tiene como valor mínimo la fecha del inicio del evento, si el evento aún no comenzó.
9. El campo “Inicio” de Duración de la actividad tiene como valor mínimo la fecha actual

si el evento ya comenzó.
10. El campo “Fin” de Duración de la actividad tiene como valor mínimo la fecha del inicio del evento si el evento aún no comenzó.
11. El campo “Fin” de Duración de la actividad tiene como valor mínimo la fecha actual si el evento ya comenzó.
12. Si la fecha y hora del campo “Inicio” de Duración de la actividad es menor a la fecha de inicio del evento. aparece un mensaje de validación “Rango de fechas no válido.”, debajo del campo “Inicio”.
13. Si la fecha y hora del campo “Inicio” de Duración de la actividad es mayor a la fecha del campo “Fin” de Duración de la actividad, aparece un mensaje de validación “Rango de fechas no válido.”, debajo del campo “Inicio”.
14. Si la fecha y hora del campo “Inicio” de Duración de la actividad es mayor a la fecha de finalización del evento. aparece un mensaje de validación “Rango de fechas no válido.”, debajo del campo “Inicio”.
15. Sí la fecha y hora del campo “Fin” de Duración de la actividad es mayor a la fecha de fin del evento, aparece un mensaje de validación “Rango de fechas no válido.”, debajo del campo “Fin”.
16. Si la fecha y hora del campo “Fin” de Duración de la actividad es menor a la fecha del campo “Inicio”, aparece un mensaje de validación “Rango de fechas no válido.”, debajo del campo “Fin”.
17. Si la fecha y hora del campo “Fin” de Duración es menor a la fecha de inicio del evento. aparece un mensaje de validación “Rango de fechas no válido.”, debajo del campo “Fin”.
18. El campo “Descripción de la actividad” tiene un máximo de 1000 caracteres.
19. Cuando se presiona en el botón “Crear” sale una alerta temporal con el mensaje “Actividad creada exitosamente”.
20. Cuando se presiona sobre “Cancelar” se limpian los campos y regresa a su estado inicial.
21. Si se ingresa el nombre de una actividad ya existente en el campo “Nombre de la actividad *” y se hace clic en el botón “Crear” se muestra una alerta temporal con el mensaje “La actividad ya existe”

7.3.1.2. Código fuente de crear actividad

Controlador de crear actividad

La función store en el controlador ActividadController se encarga de gestionar la creación de una nueva actividad. En primer lugar, se crea una instancia de la clase Actividad y se asignan los valores necesarios provenientes de la solicitud HTTP. Luego, se realiza una verificación para asegurarse de que el nombre de la actividad no exista ya en el mismo evento, evitando así duplicados. En caso de encontrar un nombre duplicado, se devuelve un mensaje de error. Si la verificación es exitosa, la actividad se guarda en la base de datos y se notifica sobre la creación mediante la función notificar. Finalmente, la

función retorna un mensaje indicando el resultado de la operación, ya sea éxito o error, junto con información adicional si es necesario.

Ruta del archivo: app/Http/Controllers/ActividadController.php

```php
public function store(Request $request)
{
    try {
        // Crear una nueva instancia de Actividad
        $actividad = new Actividad();

        // Asignar valores desde la solicitud
        $actividad->nombre = $request->nombre;
        $actividad->inicio_actividad = $request->inicio_evento;
        $actividad->fin_actividad = $request->fin_evento;
        $actividad->descripcion = $request->descripcion;
        $actividad->inscripcion = $request->inscripcion;
        $actividad->id_evento = $request->evento_id;

        // Verificar si el nombre de la actividad ya existe en el mismo evento
        $nombreExistente = Actividad::where('id_evento', $actividad->id_evento)
            ->where('nombre', $actividad->nombre)
            ->first();

        // Retornar un mensaje de error si el nombre ya existe
        if ($nombreExistente) {
            return response()->json(['mensaje' => 'La actividad ya existe', 'error' => true]);
        }

        // Guardar la actividad en la base de datos
        $actividad->save();

        // Notificar sobre la creación de la actividad
        $this->notificar($actividad, $actividad->created_at);

        // Retornar un mensaje de éxito
        return response()->json(['mensaje' => 'Actividad creada exitosamente', 'error' => false]);
    } catch (QueryException $e) {
        // Capturar y retornar mensajes de error en caso de excepción
        return response()->json(['mensaje' => $e->getMessage(), 'error' => true]);
    }
}
```

Código JavaScript de crear actividad

Se encarga de gestionar la creación de actividades a través de un formulario en una aplicación web. Comienza definiendo variables que hacen referencia a elementos del formulario, como el nombre de la actividad, el interruptor de inscripción y otros elementos relacionados. Además, se inicializan variables para almacenar el valor del interruptor y el nombre anterior de la actividad para validaciones posteriores. Presenta una función llamada crearActividad que se encarga de enviar una solicitud Axios al servidor para crear una nueva actividad. Esta función toma el formulario como parámetro y adjunta el valor del interruptor al formulario antes de realizar la solicitud. Después de recibir la respuesta del servidor, muestra una alerta de éxito o error y realiza acciones adicionales según la respuesta. El código incluye también validaciones en tiempo real para el nombre de la actividad, realizando comprobaciones y proporcionando retroalimentación visual. La función validarNombreRepetido verifica si el nombre ya existe y realiza validaciones en consecuencia. Además, hay manejo de eventos, como la

escucha del evento de envío del formulario, eventos de cambio en el nombre de la actividad, y funciones para quitar todas las validaciones visuales del formulario y redirigir al usuario después de realizar la creación.

Ruta del archivo: public/js/Actividad/crearActividad.js

```
// Obtener referencias a elementos del formulario y otros elementos relevantes
const form = document.getElementById("formularioActividad");
const inputNombre = document.getElementById('nombreActividad');
const mensajeNombre = document.getElementById('mensajeNombre');
const switchInscripcion = document.getElementById("inscripcion");
let valorCheckbox = 0; // Valor predeterminado
let nombreAnterior = ''; // Almacena el nombre anterior para validación

/**PETICIONES a AXIOS**/
/**CREAR ACTIVIDAD**/
const crearActividad = (formData) => {
    // Almacenar el nombre actual antes de la creación para validación posterior
    nombreAnterior = inputNombre.value;

    // Adjuntar el valor del interruptor al formulario si está presente
    if (switchInscripcion) {
        const estaActivado = switchInscripcion.checked;
        valorCheckbox = estaActivado ? 1 : 0;
    }

    formData.append("inscripcion", valorCheckbox);

    axios.post("/api/actividad", formData)
        .then(function (response) {
            const mensaje = response.data.mensaje;
            const nombreIgual = 'La actividad ya existe';

            // Mostrar alerta de éxito o error según la respuesta del servidor
            mostrarAlerta(
                "Éxito",
                mensaje,
                response.data.error ? "danger" : "success"
            );
```

```
            if (mensaje === nombreIgual) {
                // Validación especial para el nombre duplicado
                inputNombre.classList.remove('is-valid');
                inputNombre.classList.add('is-invalid');
                mensajeNombre.textContent = 'La actividad ya existe';
            } else {
                // Restablecer el formulario y redirigir si no hay error de "danger"
                form.querySelectorAll(".form-control, .form-select").forEach(
                    (Element) => {
                        Element.classList.remove("is-valid");
                    }
                );
                form.reset();
                if (response.data.error != "danger") {
                    setTimeout(() => {
                        window.location.href = "/admin/actividad";
                    }, 1800);
                }
            }
        })
        .catch(function (error) {
            // Manejar error en la solicitud
            mostrarAlerta("Error", "Hubo un error al guardar la actividad", "danger");
        });
};

// Escuchar el evento de envío del formulario
form.addEventListener("submit", (event) => {
    event.preventDefault();

    // Despachar un evento de cambio para todos los elementos del formulario
    form.querySelectorAll(".form-control, .form-select").forEach((Element) => {
        Element.dispatchEvent(new Event("change"));
    });

    // Validar y realizar la creación de la actividad si es válido
    if (validar()) {
        const formData = new FormData(form);
        crearActividad(formData);
    }
```

```
    // Ocultar el modal de confirmación
    $("#modalConfirmacion").modal("hide");
});

// Función para validar que no haya campos inválidos
const validar = () => {
    return form.querySelector(".is-invalid") === null;
};

/**Validacion para el input nombre**/
inputNombre.addEventListener("input", validarNombreRepetido);
inputNombre.addEventListener("change", validarNombreRepetido);

// Función para validar el nombre de la actividad y proporcionar retroalimentación visual
function validarNombreRepetido() {
    if (nombreAnterior === '') {
        if (inputNombre.value === '') {
            // Validación si el nombre está vacío
            inputNombre.classList.remove("is-valid");
            inputNombre.classList.add("is-invalid");
            mensajeNombre.textContent = "El nombre no puede estar vacío.";
        } else {
            // Validación si el nombre no está vacío
            inputNombre.classList.remove("is-invalid");
            inputNombre.classList.add("is-valid");
        }
    } else {
        if (inputNombre.value !== nombreAnterior && inputNombre.value !== '') {
            // Validación si el nombre es diferente al anterior y no está vacío
            inputNombre.classList.remove("is-invalid");
            inputNombre.classList.add("is-valid");
        } else if (inputNombre.value == '') {
            // Validación si el nombre está vacío
            inputNombre.classList.remove("is-valid");
            inputNombre.classList.add("is-invalid");
            mensajeNombre.textContent = 'El nombre no puede estar vacío.';
        } else {
            // Validación si el nombre es igual al anterior
            inputNombre.classList.remove("is-valid");
            inputNombre.classList.add("is-invalid");
            mensajeNombre.textContent = 'La actividad ya existe.';
        }
    }
}

// Función para quitar todas las validaciones visuales del formulario
function quitarValidacion() {
    form.querySelectorAll(".form-control, .form-select").forEach(
        (Element) => {
            Element.classList.remove("is-valid");
            Element.classList.remove("is-invalid");
        }
    );
    // Redirigir a la página de administración de actividades
    window.location.href = "/admin/actividad";
}
```

Vista de crear actividad

Primera vista.- La vista incluye una tabla que muestra información relevante sobre los eventos disponibles, como el nombre, el tipo de evento, la fecha de inicio y fin, y la cantidad de actividades asociadas. Solo se muestran eventos futuros. Cada fila de la tabla cuenta con un botón que redirige al usuario a la página de creación de actividades para el evento específico. Además, se incluyen enlaces a bibliotecas y scripts externos para mejorar la funcionalidad, como el manejo de datos tabulares y la integración de iconos Bootstrap.

Ruta del archivo: resources/views/actividad/crearActividad.blade.php

```
@extends('layouts.app')

@section('content')
    <div class="container">
      <div class="row mb-2">
            <h2>Crear actividad</h2>
        </div>
        <div class="row g-5">
            <div class="col-sm-12 col-md-12">
                <!-- Tabla de Eventos -->
                <table class="table table-responsive table-striped text-secondary" id="tablaEvento">
                    <caption>Eventos</caption>
                    <thead>
                        <tr>
                            <th scope="col" class="col-sm-3 col-md-3">Nombre del evento</th>
                            <th scope="col" class="col-sm-0 col-md-3 text-center">Tipo de evento</th>
                            <th scope="col" class="col-sm-3 col-md-2 text-center">Fecha de inicio del evento</th>
                            <th scope="col" class="col-sm-3 col-md-2 text-center">Fecha de fin del evento</th>
                            <th scope="col" class="col-sm-3 col-md-3 text-center font-sm">Cantidad de actividades</th>
                            <th scope="col" class="col-sm-3 col-md-2 text-center font-sm">Acción</th>
                        </tr>
                    </thead>
                    <tbody id="datosTabla">
                        @foreach ($actividades as $actividad)
                            <!-- Mostrar solo actividades futuras -->
                            @if (strtotime($actividad->fin_evento) >= time())
                                <tr>
                                    <td>{{ $actividad->nombre }}</td>
                                    <td class="text-center">{{ $actividad->tipoEvento->nombre }}</td>
                                    <td class="text-center">{{ date('d-m-Y H:i', strtotime($actividad->inicio_evento)) }}</td>
                                    <td class="text-center">{{ date('d-m-Y H:i', strtotime($actividad->fin_evento)) }}</td>
                                    <td class="text-center" id="contadorRecursos{{ $actividad->id }}">
                                        {{ $actividad->actividades->count() }}</td>
                                    <td class="text-center">
                                        <!-- Botón para crear actividad asociada al evento -->
                                        <button type="button" class="btn btn-primary btn-sm"
                                            onclick="window.location.href='/admin/actividad/crear-actividad/{{ $actividad->id }}'"
                                        >
                                            <i class="bi bi-plus"></i>
                                        </button>
                                    </td>
                                </tr>
                            @endif
                        @endforeach
                    </tbody>
                </table>
            </div>
        </div>
    </div>

    <!-- Enlaces a bibliotecas y scripts externos -->
    <link href="https://cdn.datatables.net/1.13.6/css/dataTables.bootstrap5.min.css" rel="stylesheet">
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.11.1/font/bootstrap-icons.css">
    <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.7.1/jquery.min.js"></script>
    <script src="https://cdn.datatables.net/1.13.6/js/jquery.dataTables.min.js"></script>
    <script src="https://cdn.datatables.net/1.13.6/js/dataTables.bootstrap5.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/moment.js/2.29.2/moment.min.js"></script>
    <!-- Script personalizado para ver actividades -->
    <script src="{{ asset('js/Actividad/verActividad.js') }}" defer></script>
@endsection
```

Segunda vista.- La vista está diseñada para permitir a los usuarios crear nuevas actividades asociadas a un evento específico. El formulario incluye campos para ingresar el nombre de la actividad, la duración del evento, la duración específica de la actividad, y una descripción opcional. El formulario también incorpora lógica para la gestión de actividades de inscripción, presentando un interruptor (checkbox) en caso de que no haya actividades de inscripción previas en el evento. Además, se incluyen campos ocultos para almacenar información relevante del evento, como sus fechas de inicio y fin. La vista integra validaciones tanto del lado del cliente como del lado del servidor para garantizar la integridad de los datos ingresados. Se han añadido botones para confirmar la creación de la actividad o cancelar la operación, cada uno de los cuales desencadena un modal de confirmación para evitar acciones no deseadas.

Ruta del archivo: resources/views/actividad/formCrearActividad.blade.php

```
@extends('layouts.app')

@section('content')
    <div class="container">
        <div class="row justify-content-center">
            <div class="col-md-7">
                <h2 class="text-center mb-3">Crear actividad</h2>
                <!-- Formulario para la creación de actividades -->
                <form id="formularioActividad" class="needs-validation" novalidate>
                    @csrf
                    <!-- Campos ocultos para información del evento -->
                    <input type="hidden" name="evento_id" value="{{ $evento->id }}">
                    <input type="hidden" id="fecha_evento_inicio" value="{{ $evento->inicio_evento }}">
                    <input type="hidden" id="fecha_evento_fin" value="{{ $evento->fin_evento }}">

                    <div class="container">
                        <div class="row">
                            <div class="col-md-12">
                                <!-- Campo de nombre de la actividad -->
                                <label for="nombreActividad" class="form-label">Nombre de la actividad *</label>
                                <input name="nombre" type="text" class="form-control custom-input" id="nombreActividad"
                                    value="" required placeholder="Ingrese el nombre de la actividad" maxlength="64"
                                    autocomplete="off">
                                <div id="mensajeNombre" class="invalid-feedback">
                                    El nombre no puede estar vacío.
                                </div>
                            </div>
                        </div>
```

```
<div class="row">
    <!-- Campo de inscripción si no hay actividades de inscripción previas -->
    @if (
        $evento->actividades->where('inscripcion', 1)->filter(function ($actividad) {
            return strtotime($actividad->fin_actividad) >= strtotime('today');
        })->count() > 0)
    @else
        <div class="form-check form-switch mt-4">
            <input class="form-check-input" type="checkbox" role="switch" id="inscripcion" checked>
            <label class="form-check-label" for="inscripcion">¿Desea que sea una actividad de
                inscripción?</label>
        </div>
    @endif
</div>

<div class="row">
    <div class="col-md-12 mt-4 ">
        <!-- Duración del evento -->
        <label class="form-label">Duración del evento</label>
    </div>

    <div class="row justify-content-between pr-0">
        <div class=" col-sm-12 col-md-12  col-lg-5"><strong>Inicio: </strong>
            {{ date('d-m-Y H:i', strtotime($evento->inicio_evento)) }} </div>
        <div class="col-sm-12 col-md-12 col-lg-7"><strong>Fin: </strong>
            {{ date('d-m-Y H:i', strtotime($evento->fin_evento)) }} </div>
    </div>
</div>
```

```
<div class="row">
    <div class="col-md-12 mt-4 ">
        <!-- Duración de la actividad -->
        <label class="form-label">Duración de la actividad *</label>
    </div>

    <div class="row mt-3 justify-content-between pr-0">
        <div class="col-md-5">Inicio</div>
        <div class="col-md-7 p-0">
            <!-- Campo de fecha y hora de inicio -->
            <input name="inicio_evento" id="fechaInicio" class="form-control" type="datetime-local"
                min="{{ date('Y-m-d\TH:i', strtotime($evento->inicio_evento)) }}"
                max="{{ date('Y-m-d\TH:i', strtotime($evento->fin_evento)) }}" required />
            <div id="mensajeFechaInicio" class="invalid-feedback"></div>
        </div>
    </div>

    <div class="row mt-3 justify-content-between">
        <div class="col-md-4">Fin</div>
        <div class="col-md-7 p-0">
            <!-- Campo de fecha y hora de fin -->
            <input name="fin_evento" id="fechaFin" class="form-control" type="datetime-local"
                min="{{ max(date('Y-m-d\TH:i', strtotime($evento->inicio_evento)), date('Y-m-d\TH:i')) }}"
                max="{{ date('Y-m-d\TH:i', strtotime($evento->fin_evento)) }}" disabled required />
            <div id="mensajeFechaFin" class="invalid-feedback">
                <!-- Aquí entran los mensajes de validación de fecha -->
            </div>
            <div class="col-md-1"></div>
        </div>
    </div>
```

```
                </div>

                <div class="row">
                    <div class="col-md-12 mt-4">
                        <!-- Campo de descripción de la actividad -->
                        <label for="detalleActividad" class="form-label">Descripción de la actividad</label>
                        <textarea name="descripcion" class="form-control" id="detalleActividad" rows="3" style="resize: none;"
                            placeholder="Ingrese una descripción..." maxlength="1000"></textarea>
                    </div>
                </div>

                <div class="text-center my-4">
                    <!-- Botones de acción y modales de confirmación -->
                    <button type="button" class="btn btn-secondary mx-5" data-bs-toggle="modal"
                        data-bs-target="#modalCancelar">Cancelar</button>
                    @component('components.modal')
                        @slot('modalId', 'modalCancelar')
                        @slot('modalTitle', 'Confirmación')
                        @slot('modalContent')
                            ¿Está seguro de cancelar la creación de la actividad?
                        @endslot
                        @slot('modalButton')
                            <button type="button" class="btn btn-secondary w-25 mx-8" data-bs-dismiss="modal">No</button>
                            <button type="reset" class="btn btn-primary w-25 mx-8" data-bs-dismiss="modal"
                                onClick="quitarValidacion()">Sí</button>
                        @endslot
                    @endcomponent
                    <button type="button" id="confirmarBoton" class="btn btn-primary mx-5" data-bs-toggle="modal"
                        data-bs-target="#modalConfirmacion">Crear</button>
                    @component('components.modal')
                        @slot('modalId', 'modalConfirmacion')
                        @slot('modalTitle', 'Confirmación')
                        @slot('modalContent')
                            ¿Está seguro de crear la actividad?
                        @endslot
                        @slot('modalButton')
                            <button type="button" class="btn btn-secondary w-25 mx-8" data-bs-dismiss="modal">No</button>
                            <button type="submit" class="btn btn-primary w-25 mx-8">Sí</button>
                        @endslot
                    @endcomponent
                </div>
            </form>
        </div>
    </div>
</div>
<!-- Scripts JavaScript -->
<script src="{{ asset('js/Actividad/crearActividad.js') }}" defer></script>
<script src="{{ asset('js/Actividad/validarActividad.js') }}" defer></script>
@endsection
```

7.3.1.3. Interfaz de usuario de crear actividad

La interfaz de usuario para visualizar y seleccionar eventos antes de crear una actividad proporciona una vista clara y accesible de los eventos existentes. Aquí se detallan los elementos clave de la interfaz:

Tabla de Eventos:

Nombre del Evento: Identifica el nombre del evento.

Tipo de Evento: Muestra el tipo de evento asociado al evento.

Fecha de Inicio y Fin: Presenta las fechas y horas de inicio y fin del evento en formato "d-m-Y H:i".

Cantidad de Actividades: Indica la cantidad de actividades asociadas al evento.

Acción (Agregar Actividad): Contiene botones con un ícono de suma para añadir una nueva actividad al evento seleccionado.

Botones de Acción:

Los botones de acción, representados por un ícono de suma, permiten al usuario agregar actividades a eventos específicos de manera rápida y directa.

Al hacer clic en el botón de acción, se redirige al usuario al formulario de creación de actividad, preconfigurado con la información del evento seleccionado.

Formulario de Creación de Actividad:

El formulario incluye campos esenciales para ingresar información sobre la nueva actividad, como el nombre, la descripción, la duración y la opción de actividad de inscripción.

Se utiliza el atributo "required" en los campos necesarios para garantizar que se ingresen datos válidos antes de enviar el formulario.

Se proporcionan mensajes de validación en tiempo real para informar al usuario sobre posibles errores y guiarlo en la corrección.

Interruptor de Inscripción:

Se incluye un interruptor que permite al usuario especificar si la actividad requiere inscripción.

Si hay actividades existentes con inscripción asociada al evento, se muestra un mensaje indicando que ya existen actividades con inscripción y no se muestra el interruptor.

Visualización de la Duración del Evento:

Se presenta una sección que muestra la duración del evento en la parte superior del formulario para contextualizar la programación de la nueva actividad.

Botones de Acción:

El formulario incluye botones para "Cancelar" y "Crear" la actividad.

Se implementa un modal de confirmación adicional antes de crear la actividad para asegurar que el usuario esté seguro de realizar la acción.

Interacción del Usuario:

La tabla facilita una identificación rápida de eventos disponibles.

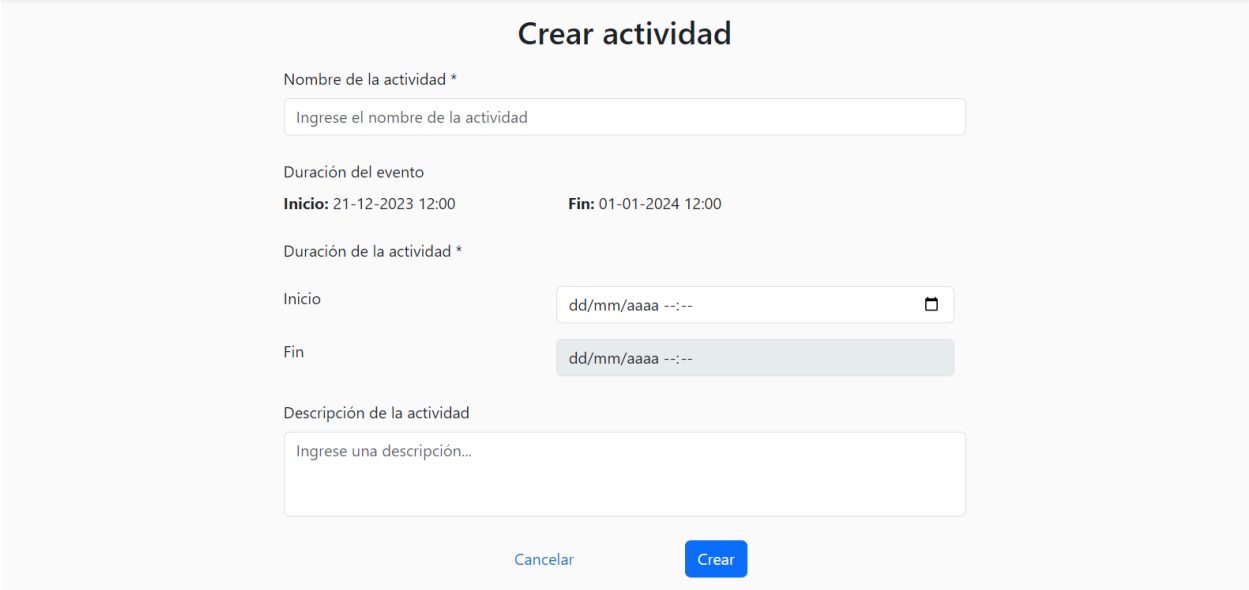
Los botones de acción ofrecen una navegación eficiente al formulario de creación de actividad, ya prellenado con los detalles del evento seleccionado.

La información clara y organizada permite al usuario tomar decisiones informadas al agregar nuevas actividades a eventos específicos.

El diseño claro y organizado del formulario facilita la entrada de datos.

Los mensajes de validación en tiempo real ayudan al usuario a corregir posibles errores antes de enviar el formulario.

El modal de confirmación antes de crear la actividad garantiza que el usuario esté consciente de la acción a realizar, evitando acciones no deseadas.



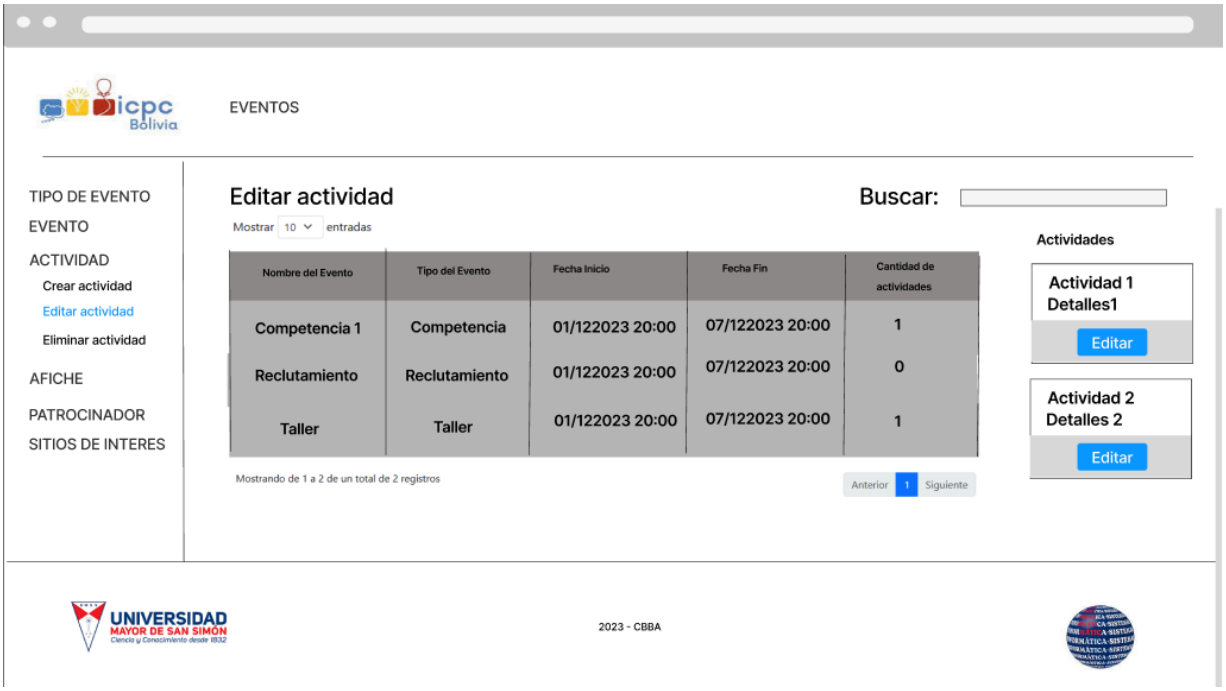
7.3.2. Editar actividad

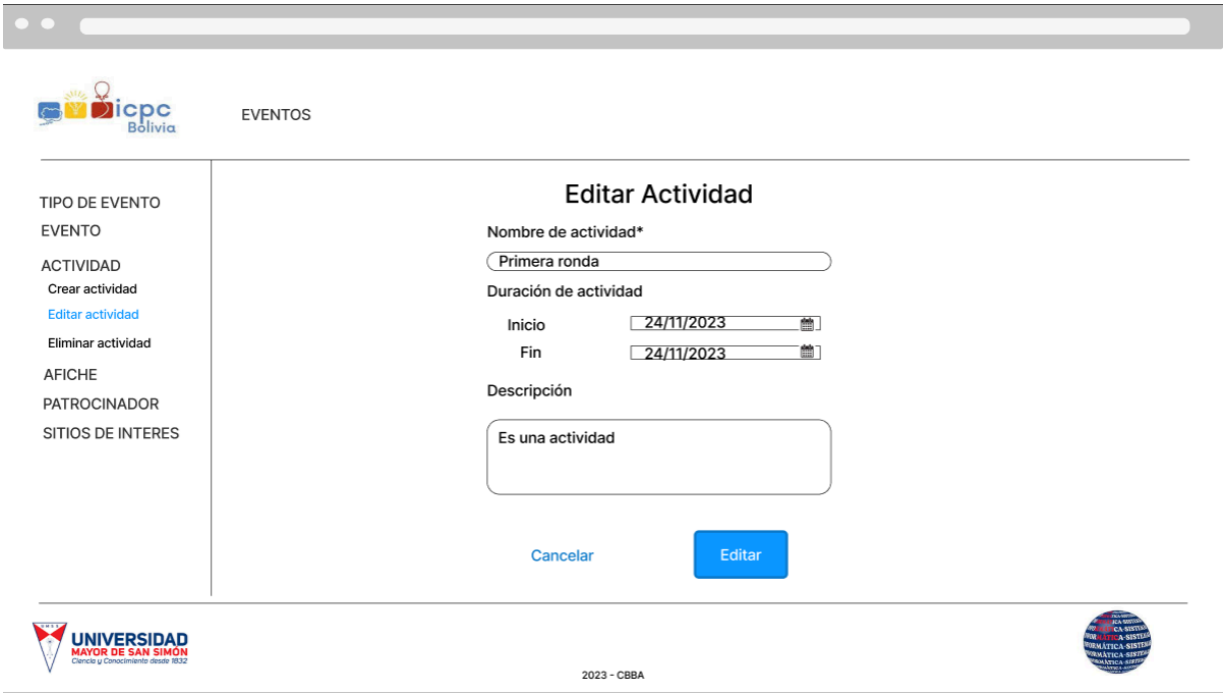
7.3.2.1. Historia de usuario de editar actividad

Historia de Usuario	
Título: Editar actividad	**ID: 46**
Estimación: 3	**Importancia: Media**
Descripción	

Como usuario con permisos para editar actividad, quiero poder editar una actividad para actualizar y ajustar la información según sea necesario y así mantener la información de las actividades al día.

Mockups





Criterios de Aceptación

1. Al hacer clic en la sección “ACTIVIDAD” en el menú lateral y seleccionar la opción “Editar actividad” se despliega una tabla.
2. La tabla tiene las siguientes columnas “Nombre del evento”, “Tipo de evento”, “Fecha de inicio del evento”, “Fecha de fin del evento”, “Cantidad de actividades”.
3. En la columna “Cantidad de actividades” se muestra la cantidad de actividades que tiene cada uno de los eventos existentes.
4. Al hacer clic sobre alguna fila de un evento existente de la tabla, en la columna derecha se muestran las actividades que tiene el evento.
5. Cada actividad se muestra en tarjetas con los siguientes detalles: nombre, descripción de la actividad, fecha y hora de inicio de la actividad, fecha y hora de finalización de la actividad y un botón de “Editar”.
6. Cuando se presione el botón de “Editar” se muestra un formulario.
7. El formulario contiene los siguientes campos “Nombre de la actividad *”, “Duración del evento” (“Inicio”, “Fin”), “Duración de la actividad *” ("Inicio ","Fin "), “Descripción de la actividad”, un interruptor  ”¿Desea enviar una notificación sobre los cambios a los usuarios?”, y los botones "Cancelar" y "Editar ".
8. En el campo de “Duración del evento” (“Inicio”, “Fin”) se muestra información de la fecha y horario de inicio y fin del evento.
9. Cada uno de los campos se muestran con la información de la actividad seleccionada.
10. El campo “Nombre de actividad *” es obligatorio.
11. El campo “Nombre de la actividad” no se repite para otra actividad en el mismo evento.
12. No se permite modificar el campo “Nombre de la actividad*” si la actividad ya inició.
13. Se permite modificar el campo 'Inicio' solo si la actividad aún no comenzó.
14. Se permite modificar el campo “Fin” solo si la actividad aún no terminó.
15. El campo “Inicio” de Duración de la actividad tiene como valor mínimo la fecha del inicio del evento, si el evento aún no comenzó.
16. El campo “Inicio” de Duración de la actividad tiene como valor mínimo la fecha actual si el evento ya comenzó.
17. El campo “Fin” de Duración de la actividad tiene como valor mínimo la fecha del inicio del evento si el evento aún no comenzó.
18. El campo “Fin” de Duración de la actividad tiene como valor mínimo la fecha actual si el evento ya comenzó.
19. Si la fecha y hora del campo “Inicio” de Duración de la actividad es menor a la fecha de inicio del evento. aparece un mensaje de validación “Rango de fechas no válido.”, debajo del campo “Inicio”.
20. Si la fecha y hora del campo “Inicio” de Duración de la actividad es mayor a la fecha del campo “Fin” de Duración de la actividad, aparece un mensaje de validación “Rango de fechas no válido.”, debajo del campo “Inicio”.
21. Si la fecha y hora del campo “Inicio” de Duración de la actividad es mayor a la fecha de finalización del evento. aparece un mensaje de validación “Rango de fechas no válido.”, debajo del campo “Inicio”.
22. Sí la fecha y hora del campo “Fin” de Duración de la actividad es mayor a la fecha de

fin del evento, aparece un mensaje de validación “Rango de fechas no válido.”, debajo del campo “Fin”.

23. Si la fecha y hora del campo “Fin” de Duración de la actividad es menor a la fecha del campo “Inicio”, aparece un mensaje de validación “Rango de fechas no válido.”, debajo del campo “Fin”.
24. Si la fecha y hora del campo “Fin” de Duración es menor a la fecha de inicio del evento. aparece un mensaje de validación “Rango de fechas no válido.”, debajo del campo “Fin”.
25. El campo “Descripción de la actividad” tiene un máximo de 1000 caracteres.
26. Si el interruptor está encendido  , al momento de hacer clic en el botón “Editar”, se envía una notificación de los cambios al correo electrónico de todos los inscritos en el evento al que pertenece la actividad.
27. Cuando se presiona “Cancelar” no se realiza ningún cambio.
28. Cuando se presiona “Editar” se muestra una alerta temporal con el mensaje “Actividad actualizada exitosamente”.

7.3.2.2. Código fuente de editar actividad

Controlador de editar actividad

En la función editarActividad, se recibe el identificador de una actividad, se obtienen los datos de la actividad y el evento asociado, y se devuelve la vista de edición con esta información. Por otro lado, la función update se encarga de procesar las solicitudes de actualización de datos de una actividad específica. Recibe los nuevos datos de la actividad a través de la solicitud HTTP, realiza validaciones, actualiza los campos pertinentes y, finalmente, notifica a los participantes y equipos asociados a la actividad si se ha solicitado. La respuesta JSON indica si la operación fue exitosa o si se produjo algún error durante el proceso.

Ruta del archivo: app/Http/Controllers/ActividadController.php

```php
/**
 * Muestra la vista para editar una actividad.
 */
public function editarActividad($id)
{
    // Obtener la actividad y el evento asociado
    $actividad = Actividad::find($id);
    $evento = Evento::find($actividad->id_evento);

    // Retornar la vista con los datos de la actividad y el evento
    return view('actividad.editarActividad', ['actividad' => $actividad, 'evento' => $evento]);
}
```

```php
/**
 * Actualiza la información de una actividad.
 */
public function update(Request $request, $id)
{
    try {
        /**Obtenemos la actividad con ID**/
        $actividad = Actividad::find($id);

        // Actualizar los campos de la actividad con los nuevos datos
        $actividad->nombre = $request->nombre;
        $actividad->inicio_actividad = $request->inicio_evento;
        $actividad->fin_actividad = $request->fin_evento;
        $actividad->descripcion = $request->descripcion;

        /**Antes de guardar debemos revisar si el nombre ya existe y que no sea el id del evento
         * Para eso obtenemos el id del evento al que pertenece esta actividad
         **/
        $nombreExistente = Actividad::where('id_evento', $actividad->id_evento)
            ->where('id', '!=', $id)
            ->where('nombre', $actividad->nombre)
            ->first();

        if ($nombreExistente) {
            return response()->json(['mensaje' => 'La actividad ya existe', 'error' => true]);
        }

        // Guardar los cambios en la actividad
        $actividad->save();

        if ($request->notificacion) {
            // Enviar notificación si está activada
            $this->notificar($actividad, $actividad->updated_at);
        }

        return response()->json(['mensaje' => 'Actividad actualizada exitosamente', 'error' => false]);
    } catch (QueryException $e) {
        return $e->getMessage();
    }
}
```

Código JavaScript de editar actividad

Se encarga de gestionar la edición de actividades mediante un formulario dinámico. Se definen constantes que representan elementos HTML relevantes del formulario, como campos de entrada, mensajes de validación y elementos de fecha. Además, se inicializan variables para almacenar información anterior y estados de checkbox. El código utiliza la biblioteca Axios para realizar peticiones HTTP, especialmente para la edición de actividades. La función editarActividad recopila los datos del formulario y realiza una solicitud POST al servidor. Se manejan las respuestas del servidor, mostrando alertas de éxito o error y realizando acciones adicionales, como redireccionamiento en caso de éxito. El formulario cuenta con validaciones en tiempo real para campos obligatorios, fechas de inicio y fin, y el nombre de la actividad. Estas validaciones se realizan al cambiar el contenido de los campos correspondientes, y se reflejan visualmente en el formulario. El código también contiene funciones para validar duplicados en el nombre de la actividad, ajustar las fechas mínimas y máximas según la selección del usuario, y deshabilitar la fecha de fin si no se cumple con ciertas condiciones.

Ruta del archivo: public/js/Actividad/crearActividad.js

```
// Constantes para elementos del formulario
const form = document.getElementById("formularioActividad");
const inputNombre = document.getElementById("nombreActividad");
const mensajeNombre = document.getElementById("mensajeNombre");
const fechaEventoInicio = document.getElementById("fechaEventoInicio");
const fechaEventoFin = document.getElementById("fechaEventoFin");
const fechaInicio = document.getElementById("fechaInicio");
const fechaFin = document.getElementById("fechaFin");
const mensajeFechaInicio = document.getElementById("mensajeFechaInicio");
const mensajeFechaFin = document.getElementById("mensajeFechaFin");
const switchMensaje = document.getElementById("notificacion");
let nombreAnterior = "";
let valorCheckbox = "";

/**PETICIONES a AXIOS**/

/**EDITAR ACTIVIDAD**/
const editarActividad = (formData) => {
    const estaActivado = switchMensaje.checked;
    valorCheckbox = estaActivado
        ? formData.append("notificacion", true)
        : "";

    axios
        .post("/api/actividad/" + formData.get("id"), formData)
        .then(function (response) {
            const mensaje = response.data.mensaje;
            const nombreIgual = "La actividad ya existe";

            // Muestra una alerta con el resultado
            mostrarAlerta(
                "Éxito",
                mensaje,
                response.data.error ? "danger" : "success"
            );
```

```
            if (mensaje === nombreIgual) {
                // Actualiza el nombre anterior si existe
                nombreAnterior = inputNombre.value;
                inputNombre.classList.remove("is-valid");
                inputNombre.classList.add("is-invalid");
                mensajeNombre.innerHTML = "La actividad ya existe";
            } else if (response.data.error != "danger") {
                // Redirecciona después de un tiempo si no hay errores graves
                setTimeout(() => {
                    window.location.href = "/admin/actividad/editar-actividad";
                }, 1800);
            }
        })
        .catch(function (error) {
            // Muestra una alerta en caso de error
            mostrarAlerta(
                "Error",
                "Hubo un error al editar la actividad",
                "danger"
            );
        });
};

// Maneja el envío del formulario
form.addEventListener("submit", (event) => {
    event.preventDefault();

    // Dispara eventos de cambio en todos los campos del formulario
    form
        .querySelectorAll(".form-control, .form-select")
        .forEach((Element) => {
            Element.dispatchEvent(new Event("change"));
        });
```

```
    // Valida el formulario antes de enviar
    if (validar()) {
        // Habilita los campos del formulario y realiza la edición
        form
            .querySelectorAll(".form-control, .form-select")
            .forEach((Element) => {
                if (Element.disabled) Element.disabled = false;
            });

        // Obtiene los datos del formulario y llama a la función para editar la actividad
        const formData = new FormData(form);
        editarActividad(formData);
    }

    // Cierra el modal de confirmación
    $("#modalConfirmacion").modal("hide");
});

// Función de validación del formulario
const validar = () => {
    return form.querySelector(".is-invalid") === null;
};

// Agregar validación a los inputs
form
    .querySelectorAll(".form-control, .form-select")
    .forEach((Element) => {
        Element.addEventListener("change", () => {
            if (
                Element.hasAttribute("required") &&
                Element.value === ""
            ) {
                Element.classList.remove("is-valid");
                Element.classList.add("is-invalid");
            } else {
                Element.classList.remove("is-invalid");
                Element.classList.add("is-valid");
            }
        });
```

```
    });

/**Validación para el input nombre**/
inputNombre.addEventListener("input", validarNombreRepetido);
inputNombre.addEventListener("change", validarNombreRepetido);

/**Validaciones para fecha INICIO**/
fechaInicio.addEventListener("change", () => {
    // Obtiene las fechas como objetos Date
    const fechaInicioSeleccionada = new Date(fechaInicio.value);
    const fechaMin = new Date(fechaInicio.min);
    const fechaMax = new Date(fechaInicio.max);

    if (fechaInicio.disabled == false) {
        // Verifica si la fecha está dentro del rango permitido
        if (
            fechaInicioSeleccionada < fechaMin ||
            fechaInicioSeleccionada > fechaMax ||
            (fechaInicio.value > fechaFin.value && fechaFin.value !== "")
        ) {
            isValid(fechaInicio, false);
            mensajeFechaInicio.innerHTML = "Rango de fechas no válido.";
            deshabilitar();
        } else if (fechaInicio.value == "") {
            isValid(fechaInicio, false);
            mensajeFechaInicio.innerHTML = "Seleccione una fecha y hora.";
            deshabilitar();
        } else {
            // Quita todos los mensajes y valida
            mensajeFechaInicio.innerHTML = "";
            isValid(fechaInicio, true);

            // Establece el valor mínimo como la fecha de inicio de la actividad
            fechaFin.disabled = false;
            fechaFin.min = fechaInicio.value;
            fechaFin.dispatchEvent(new Event("change"));
        }
    } else {
        isValid(fechaInicio, true);
    }
```

```
});

/**Validaciones para fecha FIN**/
fechaFin.addEventListener("change", () => {
    // Obtiene las fechas como objetos Date
    const fechaFinSeleccionada = new Date(fechaFin.value);
    const fechaMin = new Date(fechaFin.min);
    const fechaMax = new Date(fechaFin.max);

    // Verifica si la fecha está dentro del rango permitido
    if (fechaFin.value == "") {
        isValid(fechaFin, false);
        mensajeFechaFin.innerHTML = "Seleccione una fecha y hora.";
    } else if (
        fechaFinSeleccionada < fechaMin ||
        fechaFinSeleccionada > fechaMax ||
        (fechaFin.value < fechaInicio.value && fechaInicio.value !== "")
    ) {
        isValid(fechaFin, false);
        mensajeFechaFin.innerHTML = "Rango de fechas no válido.";
    } else {
        // Quita todos los mensajes y valida
        mensajeFechaFin.innerHTML = "";
        isValid(fechaFin, true);
    }
});

// Función para establecer el estado de validez en un componente
const isValid = (componente, bandera) => {
    if (bandera) {
        componente.classList.remove("is-invalid");
        componente.classList.add("is-valid");
    } else {
        componente.classList.remove("is-valid");
        componente.classList.add("is-invalid");
    }
};
```

```
// Función para deshabilitar el campo de fecha de fin
const deshabilitar = () => {
    fechaFin.disabled = true;
    fechaFin.value = "";
    fechaFin.classList.remove("is-valid");
    fechaFin.classList.remove("is-invalid");
};

// Evento de cambio en el input nombre para validar duplicados
function validarNombreRepetido() {
    if (nombreAnterior === "") {
        if (inputNombre.value === "") {
            inputNombre.classList.remove("is-valid");
            inputNombre.classList.add("is-invalid");
            mensajeNombre.textContent = "El nombre no puede estar vacío.";
        } else {
            inputNombre.classList.remove("is-invalid");
            inputNombre.classList.add("is-valid");
        }
    } else {
        if (inputNombre.value !== nombreAnterior && inputNombre.value !== "") {
            inputNombre.classList.remove("is-invalid");
            inputNombre.classList.add("is-valid");
        } else if (inputNombre.value == "") {
            inputNombre.classList.remove("is-valid");
            inputNombre.classList.add("is-invalid");
            mensajeNombre.textContent = "El nombre no puede estar vacío.";
        } else {
            inputNombre.classList.remove("is-valid");
            inputNombre.classList.add("is-invalid");
            mensajeNombre.textContent = "La actividad ya existe.";
        }
    }
}

// Función para quitar la validación
function quitarValidacion() {
    form
        .querySelectorAll(".form-control, .form-select")
        .forEach((Element) => {
            Element.classList.remove("is-valid");
            Element.classList.remove("is-invalid");
        });
}
```

Vista de editar actividad

La interfaz presenta un formulario que permite al usuario modificar información clave de una actividad, tales como el nombre, la duración, y la descripción. Además, se visualiza la duración del evento asociado, brindando contexto al usuario. Se incorpora un interruptor para activar o desactivar la opción de enviar notificaciones sobre los cambios a los usuarios. El formulario incluye validaciones en tiempo real, destacando la obligatoriedad de ciertos campos y proporcionando mensajes de error correspondientes. Asimismo, se implementan componentes modales para confirmar acciones críticas, como

la cancelación o confirmación de la edición de la actividad, asegurando que el usuario tome decisiones conscientes.

Ruta del archivo: resources/views/actividad/editarActividad.blade.php

```
@extends('layouts.app')

@section('content')
    <div class="container">
        <div class="row justify-content-center">
            <div class="col-md-7 " style="min-height: 500px">
                <!-- Encabezado de la página -->
                <h2 class="text-center mb-3 ">Editar actividad</h2>

                <!-- Formulario de edición de actividad -->
                <form id="formularioActividad" class="needs-validation" novalidate>
                    @csrf
                    <input type="hidden" name="id" value="{{ $actividad->id }}">

                    <!-- Campo de nombre de la actividad -->
                    <div class="container">
                        <div class="row">
                            <div class="col-md-12">
                                <label for="nombreActividad" class="form-label">Nombre de la actividad*</label>
                                <input name="nombre" type="text" class="form-control custom-input" id="nombreActividad"
                                    value="{{ $actividad->nombre }}" required
                                    placeholder="Ingrese el nombre de la actividad" maxlength="64"
                                    @if (strtotime($actividad->inicio_actividad) <= time()) disabled @endif>
                                <div id="mensajeNombre" class="invalid-feedback">
                                    El nombre no puede estar vacío.
                                </div>
                            </div>
                        </div>
                    </div>

                    <!-- Sección de duración del evento -->
                    <div class="row">
                        <div class="col-md-12 mt-4 ">
                            <label class="form-label">Duración del evento</label>
                        </div>

                        <!-- Visualización de la duración del evento -->
                        <div class="row justify-content-between pr-0">
                            <div class="col-sm-12 col-md-12 col-lg-5"><strong>Inicio: </strong> {{date('d-m-Y H:i', strtotime($evento->inicio_evento))}} </div>
                            <div class="col-sm-12 col-md-12 col-lg-7"><strong>Fin: </strong> {{date ('d-m-Y H:i', strtotime($evento->fin_evento))}} </div>
                        </div>
                    </div>

                    <!-- Sección de duración de la actividad -->
                    <div class="row">
                        <div class="col-md-12 mt-4 ">
                            <label class="form-label">Duración de la actividad *</label>
                        </div>

                        <!-- Campos de inicio y fin de la actividad -->
                        <div class="row mt-3 justify-content-between pr-0">
                            <div class="col-md-5">Inicio</div>
                            <div class="col-md-7 p-0">
                                <input name="inicio_evento" id="fechaInicio" class="form-control" type="datetime-local"
                                    @if (strtotime($actividad->inicio_actividad) <= time()) min="{{ $evento->inicio_evento }}"
                                    @else
                                        min="{{ date('Y-m-d\TH:i') }}" @endif
                                    max="{{ $evento->fin_evento }}" value="{{ $actividad->inicio_actividad }}" required
                                    @if (strtotime($actividad->inicio_actividad) <= time()) disabled @endif />
                                <div id="mensajeFechaInicio" class="invalid-feedback"></div>
```

```
                </div>
            </div>

            <div class="row mt-3 justify-content-between">
                <div class="col-md-4">Fin</div>
                <div class="col-md-7 p-0">
                    <input name="fin_evento" id="fechaFin" class="form-control" type="datetime-local"
                        value="{{ $actividad->fin_actividad }}"
                        @if (strtotime($actividad->inicio_actividad) <= time()) min="{{ date('Y-m-d\TH:i') }}"
                    @else
                        min="{{ $evento->inicio_evento }}" @endif
                        max="{{ $evento->fin_evento }}" required />
                    <div id="mensajeFechaFin" class="invalid-feedback">
                        <!-- Aquí entran los mensajes de validación de fecha -->
                    </div>
                    <div class="col-md-1"></div>
                </div>
            </div>
        </div>

        <!-- Campo de descripción de la actividad -->
        <div class="row">
            <div class="col-md-12 mt-4">
                <label for="detalleActividad" class="form-label">Descripción de la actividad</label>
                <textarea name="descripcion" class="form-control" id="detalleActividad" rows="3" style="resize: none;"
                    placeholder="Ingrese una descripción..." maxlength="1000">{{ $actividad->descripcion }}</textarea>
            </div>
        </div>
```

```
        <!-- Interruptor para enviar notificación -->
        <div class="form-check form-switch mt-4 d-flex justify-content-center">
            <input class="form-check-input" type="checkbox" role="switch" id="notificacion" checked>
            <label class="form-check-label" for="notificacion">¿Desea enviar una notificación sobre los cambios a los usuarios?</label>
        </div>

        <!-- Botones de acción -->
        <div class="text-center my-4">
            <button type="button" class="btn btn-secondary mx-5" data-bs-toggle="modal" data-bs-target="#modalCancelar">Cancelar</button>
            @component('components.modal')
                @slot('modalId', 'modalCancelar')
                @slot('modalTitle', 'Confirmación')
                @slot('modalContent')
                    ¿Está seguro de cancelar la edición de la actividad?
                @endslot
                @slot('modalButton')
                    <button type="button" class="btn btn-secondary w-25 mx-8"
                        data-bs-dismiss="modal">No</button>
                    <button type="reset" class="btn btn-primary w-25 mx-8" data-bs-dismiss="modal"
                        onClick="quitarValidacion()">Sí</button>
                @endslot
            @endcomponent

            <button id="confirmarBoton" type="button" class="btn btn-primary mx-5" data-bs-toggle="modal" data-bs-target="#modalConfirmacion">
            Editar
            </button>
            @component('components.modal')
                @slot('modalId', 'modalConfirmacion')
                @slot('modalTitle', 'Confirmación')
                @slot('modalContent')
```

```
                    ¿Está seguro de editar la actividad?
                @endslot
                @slot('modalButton')
                    <button type="button" class="btn btn-secondary w-25 mx-8" data-bs-dismiss="modal">No</button>
                    <button type="submit" class="btn btn-primary w-25 mx-8">Sí</button>
                @endslot
            @endcomponent
        </div>
    </form>
</div>
</div>
</div>
<!-- Script para la validación del formulario -->
<script src="{{ asset('js/Actividad/editarActividad.js') }}" defer></script>
@endsection
```

7.3.2.3. Interfaz de usuario de editar actividad

La interfaz de usuario ofrece una experiencia completa y eficiente para la visualización y edición de eventos y sus respectivas actividades. A continuación, se detallan los elementos clave de la interfaz:

Tabla de Eventos:

La tabla presenta una vista resumida de los eventos existentes con las siguientes columnas: Nombre del evento, Tipo de evento, Fecha de inicio del evento, Fecha de fin del evento y Cantidad de actividades.

Al hacer clic en un evento específico en la columna derecha, se activa una función que muestra tarjetas detalladas por cada actividad asociada al evento.

Tarjetas de Actividades:

Por cada actividad, se presenta una tarjeta con información detallada, incluyendo el Nombre de la actividad, Fecha de inicio y fin.

Si la actividad requiere inscripción, se muestra la etiqueta "Inscripción"; de lo contrario, no se incluye información adicional.

Cada tarjeta incluye un botón "Editar" en la parte inferior, permitiendo al usuario acceder rápidamente al formulario de edición de la actividad correspondiente.

Botón "Editar": Al hacer clic en el botón "Editar" en la tarjeta de actividad, se redirige al usuario al formulario de edición de actividad, proporcionando una transición fluida y directa para realizar cambios en la información de la actividad seleccionada.

Formulario de Edición de Actividad:

El formulario de edición de actividad presenta un diseño intuitivo y estructurado que permite al usuario modificar la información de la actividad de manera eficiente.

Incluye campos esenciales para editar el Nombre, la Descripción, y la Duración de la actividad.

Se utiliza el atributo "required" en los campos necesarios para garantizar la validez de los datos ingresados.

Se proporcionan mensajes de validación en tiempo real para guiar al usuario en la corrección de posibles errores.

Un interruptor permite especificar si la actividad requiere inscripción, con un mensaje contextual si ya existen actividades con inscripción asociadas al evento.

Botones de acción "Cancelar" y "Editar" facilitan el manejo del formulario.

Se implementa un modal de confirmación adicional antes de editar la actividad para asegurar que el usuario esté seguro de realizar la acción.

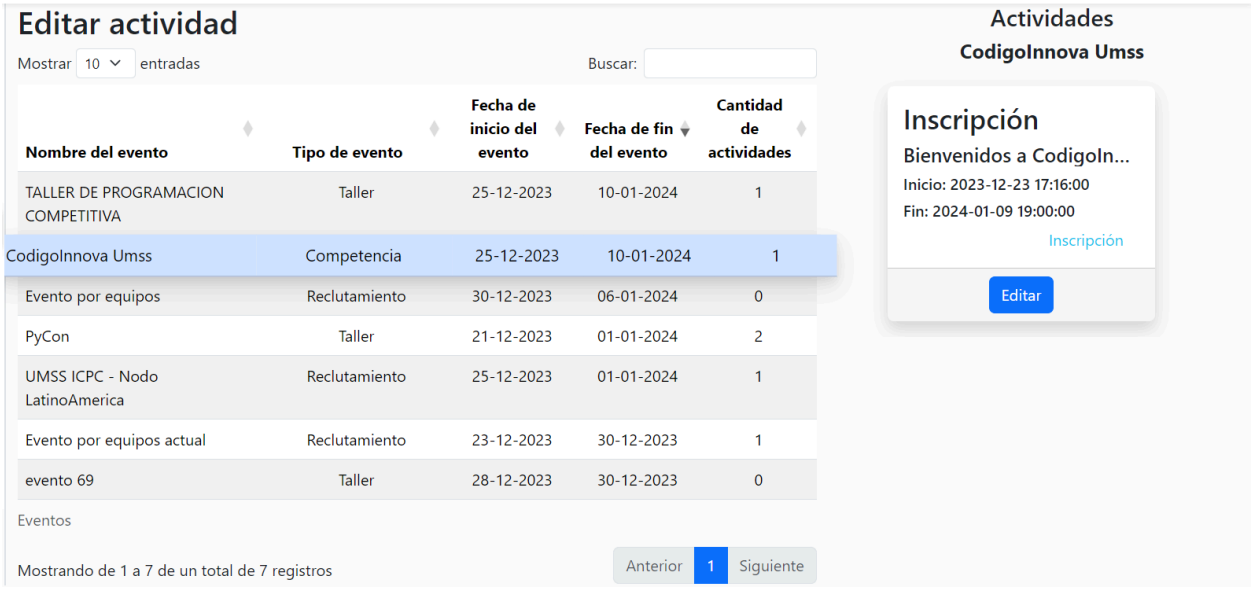
Interacción del Usuario:

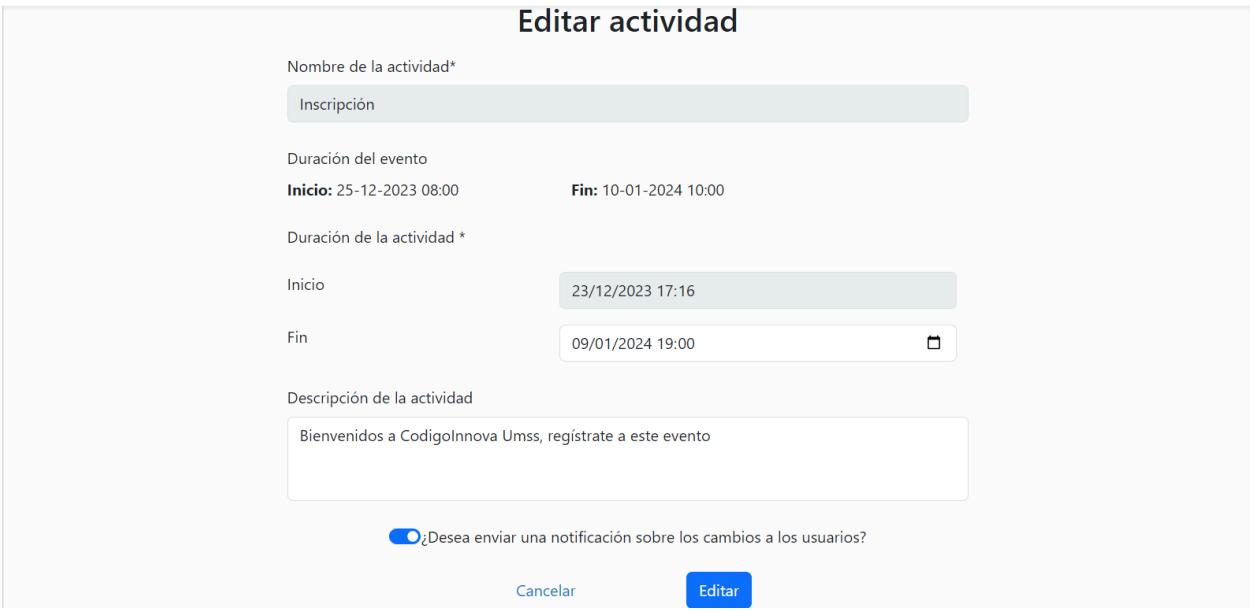
La disposición clara y organizada de la tabla de eventos facilita la identificación rápida de la información clave.

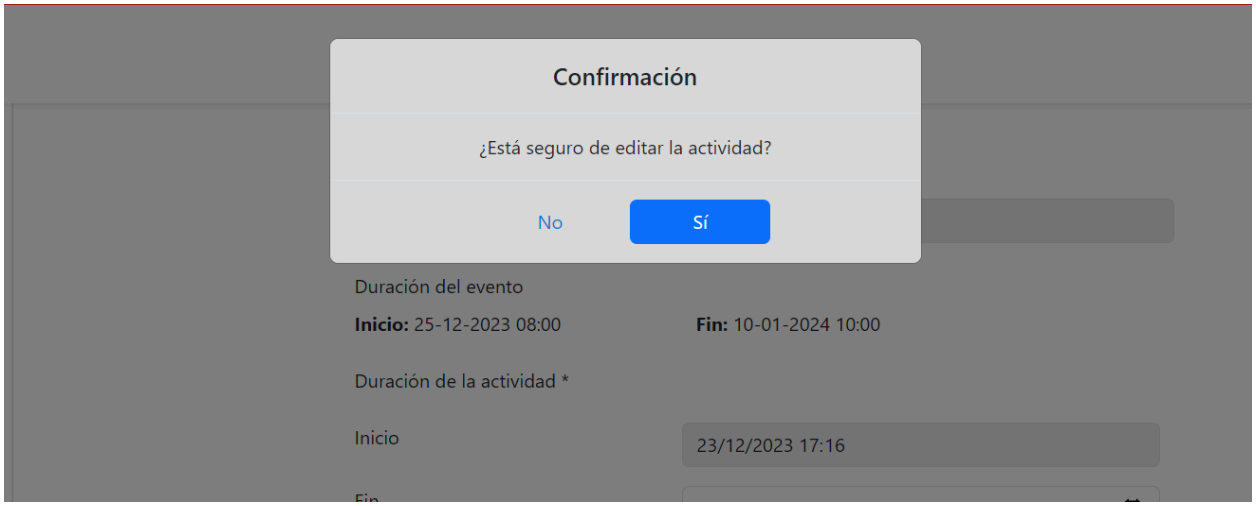
La función de tarjetas de actividades ofrece una vista detallada y accesible de cada actividad dentro de un evento, permitiendo una exploración eficiente.

El botón "Editar" en cada tarjeta agiliza el proceso de edición, brindando al usuario un acceso directo y conveniente al formulario de edición correspondiente.

La interactividad intuitiva entre la tabla y las tarjetas optimiza la experiencia del usuario al proporcionar información detallada cuando sea necesario y mantener la navegación eficiente.



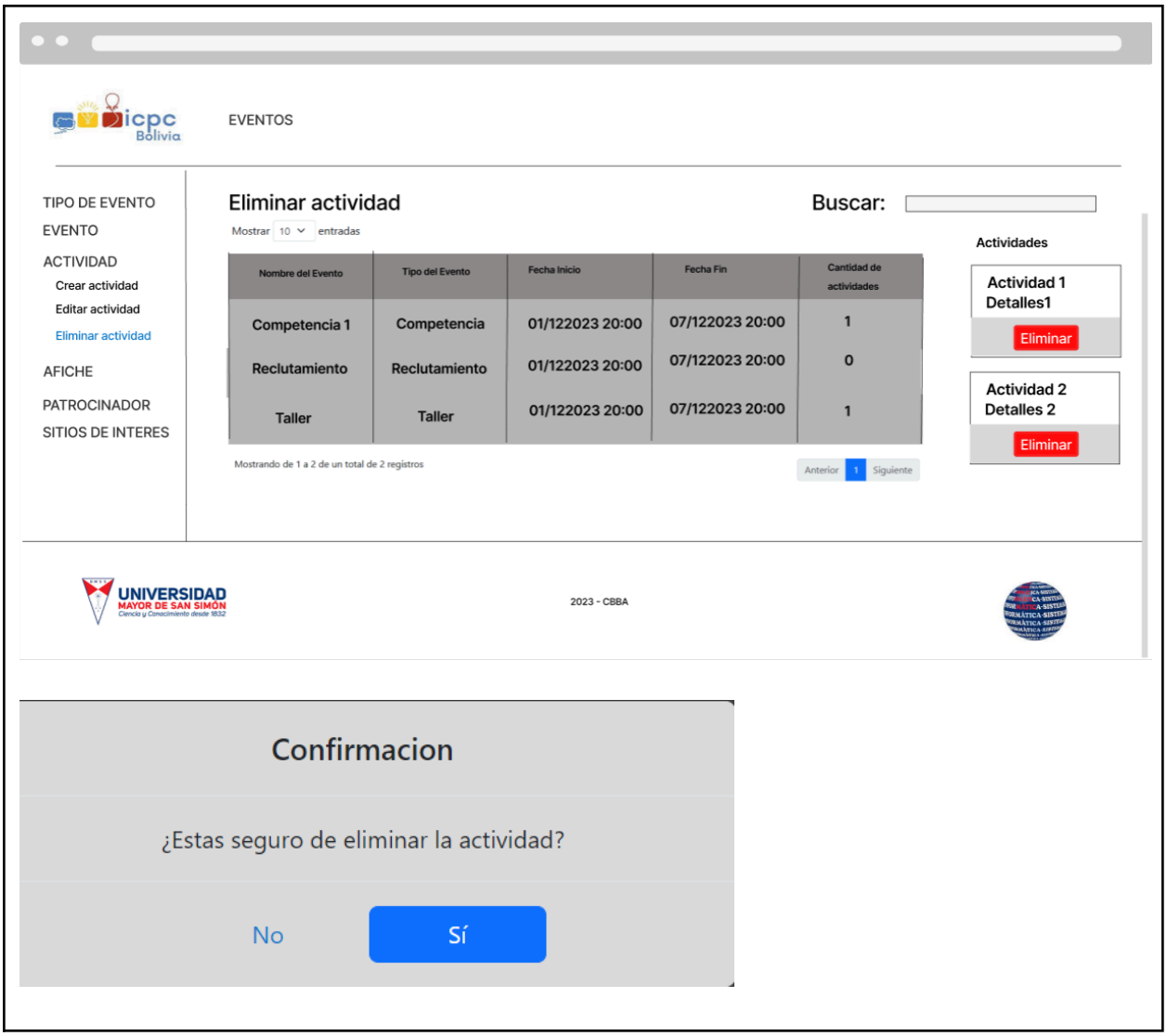




7.3.3. Eliminar actividad

7.3.3.1. Historia de usuario de eliminar actividad

Historia de Usuario	
Título: Eliminar actividad	**ID: 47**
Estimación: 3	**Importancia: Media**
Descripción: Como usuario con privilegios de eliminar actividades, quiero eliminar una actividad en caso de que mi evento cambie o surja alguna nueva necesidad que requiera el descarte de las actividades previamente creadas.	
MockUps	


icpc Bolivia
EVENTOS
TIPO DE EVENTO
EVENTO
ACTIVIDAD
Crear actividad
Editar actividad
Eliminar actividad
AFICHE
PATROCINADOR
SITIOS DE INTERES
Eliminar actividad
Buscar:
Mostrar 10 entradas
Nombre del Evento
Tipo del Evento
Fecha Inicio
Fecha Fin
Cantidad de actividades
Competencia 1
Competencia
01/122023 20:00
07/122023 20:00
1
Reclutamiento
Reclutamiento
01/122023 20:00
07/122023 20:00
0
Taller
Taller
01/122023 20:00
07/122023 20:00
1
Mostrando de 1 a 2 de un total de 2 registros
Anterior 1 Siguiente
Actividades
Actividad 1
Detalles1
Eliminar
Actividad 2
Detalles 2
Eliminar
UNIVERSIDAD MAYOR DE SAN SIMÓN
Ciencia y Conocimiento desde 1832
2023 - CBBA
Confirmacion
¿Estas seguro de eliminar la actividad?
No
Sí

Criterios de aceptación

1. Al hacer clic en la sección “ACTIVIDAD” en el menú lateral y seleccionar la opción “Eliminar actividad” se muestra una tabla con los eventos creados que están en curso o aún no empezaron.
2. La tabla tiene las siguientes columnas “Nombre del evento”, “Tipo de evento”, “Fecha de inicio del evento”, “Fecha de fin del evento”, “Cantidad de actividades”
3. La columna “Cantidad de actividades” de la tabla, muestra la cantidad de actividades que tiene cada uno de los eventos.
4. Al hacer clic sobre un evento se mostrarán todas las actividades asociadas a ese evento.
5. No se puede eliminar actividades que estén en curso.
6. No se puede eliminar actividades que terminaron.
7. Las actividades ya iniciadas o terminadas no tienen el botón de “Eliminar” solo es visible su información.
8. Al hacer clic sobre el botón “Eliminar” de alguna actividad se muestra un modal con el mensaje “¿Está seguro de eliminar la actividad?” y los botones de “No” y “Sí”.
9. Al hacer clic en el botón “No” se cierra el modal y no se elimina la actividad.
10. Al hacer clic en el botón “Si” del modal de confirmación se elimina la actividad seleccionada y se muestra una alerta temporal con el mensaje “Eliminada exitosamente”.
11. Cuando se elimina una actividad, la lista de actividades de un evento se actualiza.

7.3.3.2. Código fuente de eliminar actividad

Controlador de eliminar actividad

La función destroy en el controlador ActividadController desempeña un papel crucial al eliminar una actividad específica, identificada mediante su ID. Comienza buscando la actividad correspondiente en la base de datos. Posteriormente, notifica a los participantes y equipos asociados sobre la eliminación para mantenerlos informados de los cambios. Después de este paso, procede a eliminar la actividad de la base de datos. En caso de que surja algún error durante este proceso, como una excepción de consulta, la función maneja esta situación y devuelve el mensaje de error correspondiente. El objetivo principal es asegurar la integridad de los datos y facilitar una comunicación transparente con los usuarios afectados. La respuesta JSON generada indica el éxito del proceso de eliminación. La notificación a los participantes y equipos se realiza mediante el método notificar.

Ruta del archivo: app/Http/Controllers/ActividadController.php

```php
/**
 * Elimina una actividad específica.
 */
public function destroy($id)
{
    try {
        // Buscar la actividad utilizando el ID proporcionado
        $actividad = Actividad::find($id);

        // Notificar a los participantes y equipos asociados sobre la eliminación
        $this->notificar($actividad, now());

        // Eliminar la actividad
        $actividad->delete();

        // Retornar una respuesta JSON indicando la eliminación exitosa
        return response()->json(['mensaje' => 'Eliminada exitosamente', 'error' => false]);
    } catch (QueryException $e) {
        // Manejar cualquier error de consulta y devolver el mensaje de error asociado
        return $e->getMessage();
    }
}
```

Código JavaScript de eliminar actividad

Comienza declarando variables para referenciar elementos HTML, inicializando un DataTable con opciones de configuración, y gestionando el estado de eventos y actividades seleccionadas. La función seleccionarEvento se encarga de resaltar el evento seleccionado y mostrar sus actividades en tarjetas. Además, se proporciona una función eliminarActividad que elimina la actividad seleccionada mediante una solicitud a la API, actualizando la interfaz en consecuencia. La función cargarActividades se utiliza para cargar y mostrar dinámicamente las actividades asociadas al evento seleccionado, ocultando aquellas que ya existen y mostrando las nuevas. Este código se integra con la API para interactuar con el backend y proporciona una interfaz de usuario interactiva y dinámica para la gestión de eventos y actividades.

Ruta del archivo: public/js/Actividad/eliminarActividad.js

```
// Obtener referencias a elementos HTML
const eventoSeleccionado = document.getElementById("nombreEvento");
const contenedorAsignar = document.getElementById("contenedorAsignar");
// Variables para DataTable
let tablaDeTipos;
let tablaInicializada = false;
// Opciones de configuración para DataTable
const dataTableOptions = {
    pageLength: 10,
    lengthMenu: [5, 10, 15, 20],
    destroy: true,
    order: [[3, "desc"]],
    language: {
        lengthMenu: "Mostrar _MENU_ entradas",
        zeroRecords: "Ningún evento encontrado",
        info: "Mostrando de _START_ a _END_ de un total de _TOTAL_ registros",
        infoEmpty: "Ningún evento encontrado",
        infoFiltered: "(filtrados desde _MAX_ registros totales)",
        search: "Buscar:",
        loadingRecords: "Cargando...",
        paginate: {
            first: "Primero",
            last: "Último",
            next: "Siguiente",
            previous: "Anterior",
        },
    },
};
// Inicializar DataTable al cargar la página
const initDataTable = async () => {
    if (tablaInicializada) {
        tablaDeTipos.destroy();
    }
    DataTable.datetime("DD-MM-YYYY");
    tablaDeTipos = $("#tablaEvento").DataTable(dataTableOptions);
    tablaInicializada = true;
};
```

```
// Manejar el evento de carga de la página
window.addEventListener("load", () => {
    initDataTable();
    if (!seleccionado) {
        eventoSeleccionado.textContent = "Selecciona un evento";
    }
});

// Variables para gestionar el estado del evento seleccionado y la actividad seleccionada
let seleccionado;
let idEvento;
let eventoAux;
let actividadSeleccionada;

// Función para seleccionar un evento y mostrar sus actividades
const seleccionarEvento = (evento) => {
    if (seleccionado) {
        seleccionado.classList.remove("table-primary");
        contenedorAsignar.innerHTML = "";
    }
    seleccionado = document.getElementById(evento.id);
    seleccionado.classList.add("table-primary");
    idEvento = evento.id;
    eventoSeleccionado.textContent = evento.nombre;
    cambiarEvento(evento);
    eventoAux = evento;
    cargarActividades();
};

// Función para cambiar el evento seleccionado y actualizar la lista de actividades
const cambiarEvento = (evento) => {
    // Limpiar el contenedor antes de agregar nuevas actividades
    contenedorAsignar.innerHTML = "";

    // Crear un contenedor div para las actividades con una altura fija y overflow-y: auto
    const actividadesContainer = document.createElement("div");
    actividadesContainer.style.maxHeight = "350px"; // Ajusta la altura según sea necesario
    actividadesContainer.style.overflowY = "auto";
    contenedorAsignar.appendChild(actividadesContainer);
```

```
// Iterar sobre las actividades del evento y crear tarjetas para cada una
evento.actividades.forEach((actividad) => {
    const actividadDiv = document.createElement("div");
    actividadDiv.classList.add("col-auto");
    actividadDiv.id = `tarjetaActividad${actividad.id}`;

    // Verificar si la fecha de inicio es mayor al día de hoy
    const fechaInicio = new Date(actividad.inicio_actividad);
    const hoy = new Date();

    if (fechaInicio > hoy) {
        // Agregar contenido HTML a la tarjeta de actividad con el botón de eliminar
        actividadDiv.innerHTML = `
            <div class="container col-12 col-md-12 col-lg-12">
                <div class="card w-100 my-3 shadow" style="min-height: 100px; width: 17rem !important">
                    <div class="card-body">
                        <h3 class="card-title">${actividad.nombre}</h3>
                        <h5 class="card-text text-truncate d-block" style="max-width: 300px;" title=
                        "${actividad.descripcion}">${actividad.descripcion}</h5>
                        <h6 class="card-text text-truncate d-block" style="max-width: 300px;">Inicio:
                        ${actividad.inicio_actividad}</h6>
                        <h6 class="card-text text-truncate d-block" style="max-width: 300px;">Fin:
                        ${actividad.fin_actividad}</h6>
                    </div>
                    <div class="card-footer d-flex justify-content-center">
                        <a href="#" class="btn btn-danger btn-sm" data-bs-toggle="modal"
                            data-bs-target="#modalEliminarActividad" onclick="seleccionarActividad(
                            ${actividad.id})">Eliminar
                        </a>
                    </div>
                </div>
            </div>
        `;
    } else {
        // Agregar contenido HTML a la tarjeta de actividad sin el botón de eliminar
        actividadDiv.innerHTML = `
            <div class="container col-12 col-md-12 col-lg-12">
                <div class="card w-100 my-3 shadow" style="min-height: 100px; width: 17rem !important">
                    <div class="card-body">
```

```
                        <h3 class="card-title">${actividad.nombre}</h3>
                        <h5 class="card-text text-truncate d-block" style="max-width: 300px;" title=
                        "${actividad.descripcion}">${actividad.descripcion}</h5>
                        <h6 class="card-text text-truncate d-block" style="max-width: 300px;">Inicio:
                        ${actividad.inicio_actividad}</h6>
                        <h6 class="card-text text-truncate d-block" style="max-width: 300px;">Fin:
                        ${actividad.fin_actividad}</h6>
                    </div>
                </div>
            </div>
        `;
    }

    // Agregar la tarjeta de actividad al contenedor de actividades
    actividadesContainer.appendChild(actividadDiv);
  });
};

// Función para seleccionar una actividad
const seleccionarActividad = (id) => {
  actividadSeleccionada = id;
};

// Función para eliminar una actividad
const eliminarActividad = async () => {
  if (actividadSeleccionada) {
    await axios
      .delete(`/api/actividad/${actividadSeleccionada}`)
      .then((response) => {
        // Manejar la respuesta después de eliminar la actividad
        mostrarAlerta(
          "Éxito",
          response.data.mensaje,
          response.error ? "danger" : "success"
        );
```

```
                // Actualizar el contador de actividades en la tabla
                const contadorActividadesElement = document.getElementById(`contadorActividades
                ${idEvento}`);
                if (contadorActividadesElement) {
                    const newCount = parseInt(contadorActividadesElement.textContent) - 1;
                    contadorActividadesElement.textContent = newCount;
                }

                // Recargar las actividades después de eliminar
                cargarActividades();
            });
    }
};

// Función para cargar y mostrar las actividades del evento seleccionado
const cargarActividades = async () => {
    const tarjetasActividad = document.querySelectorAll('[id^="tarjetaActividad"]');

    // Ocultar todas las actividades existentes
    tarjetasActividad.forEach((tarjetaActividad) => {
        tarjetaActividad.style.display = 'none';
    });

    // Obtener las actividades del evento seleccionado mediante una solicitud a la API
    await axios.get(`/api/actividad/${idEvento}`).then((response) => {
        // Iterar sobre las actividades obtenidas
        response.data.forEach((actividad) => {
            const actividadDiv = document.getElementById(`tarjetaActividad${actividad.id}`);

            if (actividadDiv) {
                // Si la actividad ya existe, mostrarla
                actividadDiv.style.display = 'block';
            } else {
                // Si la actividad no existe, crearla y agregarla al contenedor
                cambiarEvento(eventoAux);
            }
        });

        // Actualizar el contador de actividades en la tabla después de cargar
        const contadorActividadesElement = document.getElementById(`contadorActividades
        ${idEvento}`);
        if (contadorActividadesElement) {
            contadorActividadesElement.textContent = response.data.length;
        }
    });
};
```

7.3.3.3. Interfaz de usuario de eliminar actividad

La interfaz de usuario para la eliminación de actividades ofrece una experiencia completa y eficiente. Aquí se describen los elementos y la interacción del usuario:

Tabla de Eventos:

La tabla presenta una visión general de los eventos existentes con detalles como el Nombre del evento, Tipo de evento, Fecha de inicio, Fecha de fin y Cantidad de actividades.

Al seleccionar un evento en la columna derecha, se activa una función que muestra tarjetas detalladas por cada actividad vinculada al evento.

Tarjetas de Actividades:

Cada actividad se presenta en una tarjeta con información detallada, incluyendo el Nombre de la actividad, descripción y Fecha de inicio y fin.

Si la actividad requiere inscripción, se indica con la etiqueta "Inscripción"; de lo contrario, no se incluye información adicional.

Cada tarjeta tiene un botón "Eliminar" en la parte inferior.

Modal de Confirmación para Eliminar Actividad:

Al hacer clic en el botón "Eliminar" en la tarjeta de actividad, se desencadena un modal de confirmación.

El modal muestra un mensaje claro preguntando al usuario si está seguro de eliminar la actividad.

Se proporcionan botones "No" y "Sí" para permitir al usuario confirmar o cancelar la eliminación.

Proceso de Eliminación: Si el usuario hace clic en "Sí" en el modal de confirmación, se activa la eliminación de la actividad seleccionada.

Se muestra una alerta indicando el éxito o cualquier error durante el proceso de eliminación.

Interacción del Usuario:

La tabla de eventos facilita la identificación rápida de la información clave.

Las tarjetas de actividades ofrecen detalles específicos, y el botón "Eliminar" proporciona una opción directa para eliminar la actividad.

La inclusión de un modal de confirmación garantiza que el usuario tome decisiones informadas antes de proceder con la eliminación.

7.3.4. Tabla de base de datos de actividades

La tabla 'actividades' almacena información esencial sobre cada actividad planificada. Cada registro cuenta con un identificador único ('id') para referencia, así como con campos como 'nombre' y 'descripcion' para describir la actividad de manera concisa y detallada respectivamente. Los campos 'inicio_actividad' y 'fin_actividad' indican las fechas de inicio y finalización, mientras que 'inscripcion' rastrea los detalles sobre la inscripción. La relación con el evento principal se establece a través del campo 'id_evento', vinculando cada actividad a su correspondiente evento en el sistema."

actividades		
PK	id	BIGINT(20)
*	nombre	VARCHAR(64)
	descripcion	TEXT
*	Inicio_actividad	DATETIME
*	fin_actividad	DATETIME
	inscripcion	INT(11)
FK	Id_evento	BIGINT(20)

7.4. AFICHE

7.4.1. Asignar afiche

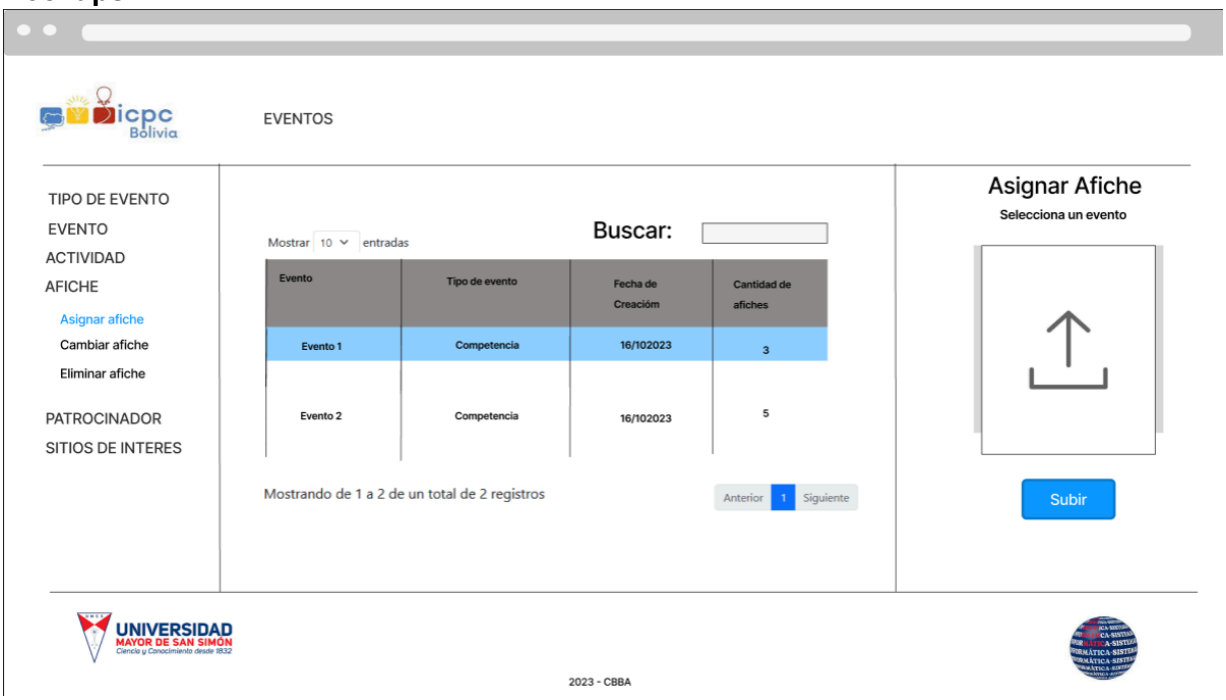
7.4.1.1. Historia de usuario de asignar afiche

Historia de Usuario	
Título: Agregar afiche a un evento	**ID: 6**
Estimación: 8	**Importancia: Media**

Descripción

Yo como usuario con privilegios para asignar afiche a un evento, quiero una opción que me permita agregar un afiche para una mejor visibilidad y publicidad del evento.

Mockups



Criterios de Aceptación

1. Al hacer clic en la sección de “AFICHE” en el menú lateral y seleccionar la opción “Asignar afiche”, se muestra una tabla con los eventos registrados previamente, y se muestra una sección donde se previsualiza la imagen a subir y un botón “Subir”.
2. El botón “Subir” está deshabilitado por defecto.
3. La tabla tiene las siguientes columnas, “Nombre del evento”,”Tipo de evento”,”Fecha de creación”,”Cantidad de afiches”.
4. Al hacer clic en un evento de la tabla, se habilita el botón “Subir”.
5. Al hacer clic en el botón “Subir”, se muestra el administrador de archivos del sistema operativo para seleccionar una imagen.
6. La imagen seleccionada debe tener como peso máximo 5 mb.
7. La imagen seleccionada tiene los formatos permitidos jpg, jpeg, png.
8. Si no ocurre error en la validación, la imagen seleccionada se previsualiza.
9. Si no ocurre error en la validación aparecen dos botones “Reemplazar” y “Asignar”.

10. Al hacer clic “Reemplazar” abre el administrador de archivos del sistema operativo para seleccionar una imagen.
11. Al hacer clic en el botón “Asignar” sale una alerta temporal con el mensaje “Asignado exitosamente”.
12. Si se asigna correctamente un afiche a un evento, el valor en la columna “Cantidad de afiches” del evento seleccionado se incrementa en uno.
13. Si la imagen seleccionada no cumple con los formatos aparece una alerta con el mensaje “Archivo no válido”.
14. Si la imagen seleccionada no cumple con el peso establecido aparece una alerta con el mensaje “Archivo no válido”.

7.4.1.2. Código fuente de asignar afiche

Controlador de asignar afiche

El controlador AficheController en el sistema Laravel contiene funciones relacionadas con la gestión de afiches para eventos, destacando especialmente la asignación y eliminación de afiches. La función storageAfiche se encarga de almacenar la imagen del afiche en el sistema de archivos, verificando si se ha enviado un archivo y devolviendo la URL de la imagen almacenada. Por otro lado, la función asignarAfiche asigna un nuevo afiche a un evento específico, creando una instancia de la clase Afiche, almacenando la ruta de la imagen utilizando la función anterior y guardando la información en la base de datos.

Ruta del archivo: app/Http/Controllers/AficheController.php

```php
/**
 * Almacena la imagen del afiche en el sistema de archivos.
 */
public function storageAfiche(Request $request)
{
    try {
        // Verifica si se ha enviado un archivo en la solicitud.
        if ($request->hasFile('afiche')) {
            // Almacena la imagen en la carpeta 'public/evento' y devuelve la ruta.
            $ruta = $request->file('afiche')->store('public/evento');
            return Storage::url($ruta);
        }
        // Retorna un mensaje de error si no se proporcionó un archivo.
        return "error";
    } catch (\Throwable $th) {
        // Retorna una respuesta JSON indicando un error.
        return response()->json(['error' => true]);
    }
}
/**
 * Asigna un afiche a un evento específico.
 */
public function asignarAfiche(Request $request)
{
    try {
        // Crea una nueva instancia de Afiche.
        $afiche = new Afiche();
        // Asigna la ruta de la imagen utilizando la función storageAfiche.
        $afiche->ruta_imagen = $this->storageAfiche($request);
        // Asigna el ID del evento al afiche.
        $afiche->id_evento = $request->id_evento;
        // Guarda el afiche en la base de datos.
        $afiche->save();
        // Retorna una respuesta JSON indicando el éxito de la operación.
        return response()->json(['mensaje' => 'Asignado exitosamente', 'error' => false]);
    } catch (QueryException $e) {
        // Retorna un mensaje de error en caso de excepción de consulta.
        return $e->getMessage();
    }
}
```

Código JavaScript de asignar afiche

Se encarga de gestionar la interfaz de usuario para la asignación de afiches a eventos. Utiliza la librería DataTable para crear y gestionar una tabla que muestra información resumida de los eventos. La interfaz permite la selección de un evento específico, mostrando de manera dinámica una vista previa del afiche asociado a dicho evento. Además, la tabla se actualiza en tiempo real al asignar afiches. El código también incluye funciones para validar la elección de imágenes antes de su asignación, asegurando que cumplan con ciertos requisitos de tipo y tamaño. Al seleccionar un evento, se activa la función que permite la previsualización de la imagen a subir, y tras la asignación, se actualiza la visualización del afiche en la interfaz.

Ruta del archivo: public/js/Afiche/afiche.js

```
// Declaración de variables para la gestión de la tabla y elementos del DOM
let tablaDeTipos;
let tablaInicializada = false;
const input = document.getElementById("imageUpload");
const imagenPreview = document.getElementById("imagePreview");
const botonSubir = document.getElementById("botonSubirAfiche");
const contenedorAsignar = document.getElementById("contenedorAsignar");
const eventoSeleccionado = document.getElementById("nombreEvento");
// Opciones de configuración para la tabla DataTable
const dataTableOptions = {
    pageLength: 10,
    lengthMenu: [5, 10, 15, 20],
    destroy: true,
    order: [[2, "desc"]],
    language: {
        // Configuración de mensajes y etiquetas en la tabla
        lengthMenu: "Mostrar _MENU_ entradas",
        zeroRecords: "Ningún evento encontrado",
        info: "Mostrando de _START_ a _END_ de un total de _TOTAL_ registros",
        infoEmpty: "Ningún evento encontrado",
        infoFiltered: "(filtrados desde _MAX_ registros totales)",
        search: "Buscar:",
        loadingRecords: "Cargando...",
        paginate: {
            first: "Primero",
            last: "Último",
            next: "Siguiente",
            previous: "Anterior",
        },
    },
};
// Inicialización de la tabla DataTable y configuración inicial al cargar la página
const initDataTable = async () => {
    if (tablaInicializada) {
        tablaDeTipos.destroy();
    }
    DataTable.datetime("DD-MM-YYYY");
    tablaDeTipos = $("#tablaEvento").DataTable(dataTableOptions);
    tablaInicializada = true;
};
```

```
// Evento que se ejecuta al cargar la página
window.addEventListener("load", () => {
    initDataTable();
    // Deshabilitar el botón de subir afiche si no hay un evento seleccionado
    if (!seleccionado) {
        eventoSeleccionado.textContent = "Selecciona un evento";
        botonSubir.disabled = true;
    }
});
// Función para validar el tipo y tamaño de una imagen antes de subirla
const validarImagen = (input, peso, callback) => {
    if (input.files.length > 0) {
        const imagen = input.files[0];
        const maxFileSize = peso * 1024 * 1024;
        let mensaje = { mensaje: "", error: false };

        // Validar el tipo y tamaño de la imagen
        const type = !/image\/(png|jpeg|jpg)/.test(imagen.type);
        if (type || imagen.size > maxFileSize) {
            input.value = "";
            mensaje = { mensaje: "Archivo no válido", error: true };
        }

        // Ejecutar el callback con el mensaje de validación
        if (typeof callback === "function") {
            callback(mensaje);
        }
    }
};
// Variables para gestionar la selección de eventos
let seleccionado;
let idSeleccionado;

// Función que se ejecuta al seleccionar un evento
const seleccionarEvento = (id, nombre) => {
    // Desmarcar evento anterior y habilitar el botón de subir afiche
    if (seleccionado) {
        seleccionado.classList.remove("table-primary");
    }
    botonSubir.disabled = false;
```

```
    // Marcar el evento seleccionado y actualizar información en la interfaz
    seleccionado = document.getElementById(id);
    seleccionado.classList.add("table-primary");
    idSeleccionado = id;
    eventoSeleccionado.textContent = nombre;

    // Limpiar la selección anterior de archivos y mostrar la vista previa por defecto
    input.value = "";
    imagenPreview.src = "/image/uploading.png";
    botonSubir.style.display = "block";
    contenedorAsignar.style.display = "none";
};

// Función para previsualizar una imagen antes de subirla
function previsualizarImagen(event) {
    // Validar la imagen y mostrar la vista previa si es válida
    validarImagen(input, 5, (mensaje) => {
        if (!mensaje.error) {
            const file = event.target.files[0];
            const reader = new FileReader();
            reader.onload = function (e) {
                imagenPreview.src = e.target.result;
            };
            reader.readAsDataURL(file);

            // Ocultar el botón de subir y mostrar el contenedor para asignar
            botonSubir.style.display = "none";
            contenedorAsignar.style.display = "block";
        } else {
            // Mostrar mensaje de error en caso de imagen no válida
            mostrarAlerta("Error", mensaje.mensaje, "danger");
        }
    });
}
```

```
// Función para asignar un afiche al evento seleccionado
const asignarAfiche = async () => {
    if (idSeleccionado) {
        // Crear un formulario con la imagen y el ID del evento
        const form = new FormData();
        form.append("afiche", input.files[0]);
        form.append("id_evento", idSeleccionado);

        // Enviar la solicitud al servidor y mostrar mensaje de resultado
        await axios.post("/api/afiche", form).then((response) => {
            mostrarAlerta(
                "Éxito",
                response.data.mensaje,
                response.error ? "danger" : "success"
            );
            // Actualizar la visualización del afiche después de asignar
            cargarAfiche();
        });
    }
};

// Función para cargar y actualizar la visualización del afiche asociado al evento
const cargarAfiche = async () => {
    await axios.get(`/api/afiche/${idSeleccionado}`).then((response) => {
        // Actualizar el contador de afiches en la tabla y mostrar vista por defecto
        document.getElementById(
            `contadorAfiches${idSeleccionado}`
        ).textContent = response.data.length;
        imagenPreview.src = "/image/uploading.png";
        botonSubir.style.display = "block";
        contenedorAsignar.style.display = "none";
    });
};
```

Vista de asignar afiche

La vista se divide en dos secciones principales.

En la primera sección, se presenta una tabla que muestra información relevante sobre los eventos disponibles, como el nombre del evento, el tipo de evento, la fecha de creación y la cantidad de afiches asociados. Se utiliza la función @foreach para iterar sobre la colección de eventos y generar dinámicamente las filas de la tabla. Además, se asignan funciones JavaScript a los clics en las filas para permitir la selección de un evento específico.

La segunda sección consiste en un contenedor para la asignación de afiches a eventos seleccionados. Permite cargar una imagen como afiche y proporciona una vista previa de la misma. Se incluyen botones para subir, asignar y reemplazar afiches, así como un mensaje informativo. La interactividad de la interfaz se logra mediante funciones JavaScript, como la previsualización de la imagen antes de la carga y la ejecución de acciones al hacer clic en los botones.

Ruta del archivo: resources/views/afiche/asignarAfiche.blade.php

```
@extends('layouts.app')

@section('content')
    <div class="container">
        <div class="row g-5">
            <div class="col-sm-12 col-md-8">
                <!-- Encabezado y tabla de eventos -->
                <h2>Asignar afiche</h2>
                <table class="table table-responsive table-striped text-secondary table-hover cursor" id="tablaEvento">
                    <caption>Eventos</caption>
                    <thead>
                        <tr>
                            <th scope="col" class="col-sm-4 col-md-4">Nombre del evento</th>
                            <th scope="col" class="col-sm-0 col-md-3 text-center">Tipo de evento</th>
                            <th scope="col" class="col-sm-3 col-md-2 text-center">Fecha de creación</th>
                            <th scope="col" class="col-sm-3 col-md-2 text-center font-sm">Cantidad de afiches</th>
                        </tr>
                    </thead>
                    <tbody id="datosTabla">
                        <!-- Iteración sobre los eventos y visualización en la tabla -->
                        @foreach ($afiches as $afiche)
                            @if (strtotime($afiche->fin_evento) >= time())
                                <tr onclick="seleccionarEvento({{ $afiche->id }}, '{{ $afiche->nombre }}', event)"
                                    id="{{ $afiche->id }}">
                                    <td>{{ $afiche->nombre }}</td>
                                    <td class="text-center">{{ $afiche->tipoEvento->nombre }}</td>
                                    <td class="text-center">{{ date('d-m-Y', strtotime($afiche->created_at)) }}</td>
                                    <td class="text-center" id="contadorAfiches{{ $afiche->id }}">
                                        {{ $afiche->afiches->count() }}</td>
                                </tr>
                            @endif
                        @endforeach
                    </tbody>
                </table>
            </div>
            <div class="col-sm-12 col-md-4">
                <!-- Contenedor para asignar afiche -->
                <div class="container d-flex flex-column justify-content-center align-items-center border p-3 container-image">
                    <div class="col-12 d-flex justify-content-center align-items-center">
                        <h4>Asignar afiche</h4>
                    </div>
                    <h5 id="nombreEvento" class="text-center fw-bold"></h5>
                    <div class="d-flex justify-content-center">
                        <!-- Vista previa de la imagen y carga de archivos -->
                        <img src="{{ asset('/image/uploading.png') }}" alt="image" id="imagePreview"
                            class="rounded mx-auto d-block img-thumbnail">
                        <input type="file" id="imageUpload" class="d-none" accept="image/jpeg, image/png, image/jpg"
                            onchange="previsualizarImagen(event)">
                    </div>
                    <div class="d-flex justify-content-center mt-5">
                        <!-- Botón de subir afiche y contenedor para asignar o reemplazar -->
                        <button type="button" class="btn btn-primary btn-lg hover-button" id="botonSubirAfiche"
                            onclick="document.getElementById('imageUpload').click()">Subir</button>
                    </div>
                    <div id="contenedorAsignar" style="display: none">
                        <!-- Botones para asignar o reemplazar afiche y mensaje informativo -->
                        <div class="d-flex justify-content-center">
                            <button type="button" class="btn btn-light btn-lg hover-button"
                                onclick="document.getElementById('imageUpload').click()">Reemplazar</button>
                            <button type="button" class="btn btn-primary btn-lg hover-button"
```

```
                                onclick="asignarAfiche()">Asignar</button>
                            </div>
                            <div>
                                <p class="text-muted mt-3">
                                    Si el afiche seleccionado no es el deseado, puede reemplazarlo
                                    presionando "Reemplazar".
                                </p>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </div>
        <!-- Enlaces a las librerías y scripts necesarios -->
        <link href="https://cdn.datatables.net/1.13.6/css/dataTables.bootstrap5.min.css" rel="stylesheet">
        <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.7.1/jquery.min.js"></script>
        <script src="https://cdn.datatables.net/1.13.6/js/jquery.dataTables.min.js"></script>
        <script src="https://cdn.datatables.net/1.13.6/js/dataTables.bootstrap5.min.js"></script>
        <script src="https://cdnjs.cloudflare.com/ajax/libs/moment.js/2.29.2/moment.min.js"></script>
        <script src="{{ asset('js/Afiche/afiche.js') }}" defer></script>
@endsection
```

7.4.1.3. Interfaz de usuario de asignar afiche

La interfaz de usuario destinada a la asignación de afiches a eventos presenta un diseño claro y funcional, facilitando la tarea de los usuarios. A continuación, se detallan los elementos clave de esta interfaz:

Tabla de Eventos:

La tabla central exhibe una lista de eventos con columnas relevantes, como "Nombre del evento", "Tipo de evento", "Fecha de creación" y "Cantidad de afiches". Al hacer clic en un evento específico, se activa una función que muestra detalles adicionales en la sección lateral.

Previsualizador de Imágenes:

En la sección derecha, se encuentra un previsualizador de imágenes que permite visualizar la imagen seleccionada como afiche. Inicialmente, muestra una imagen de carga. Este espacio es activado al seleccionar un evento en la tabla.

Botón "Subir":

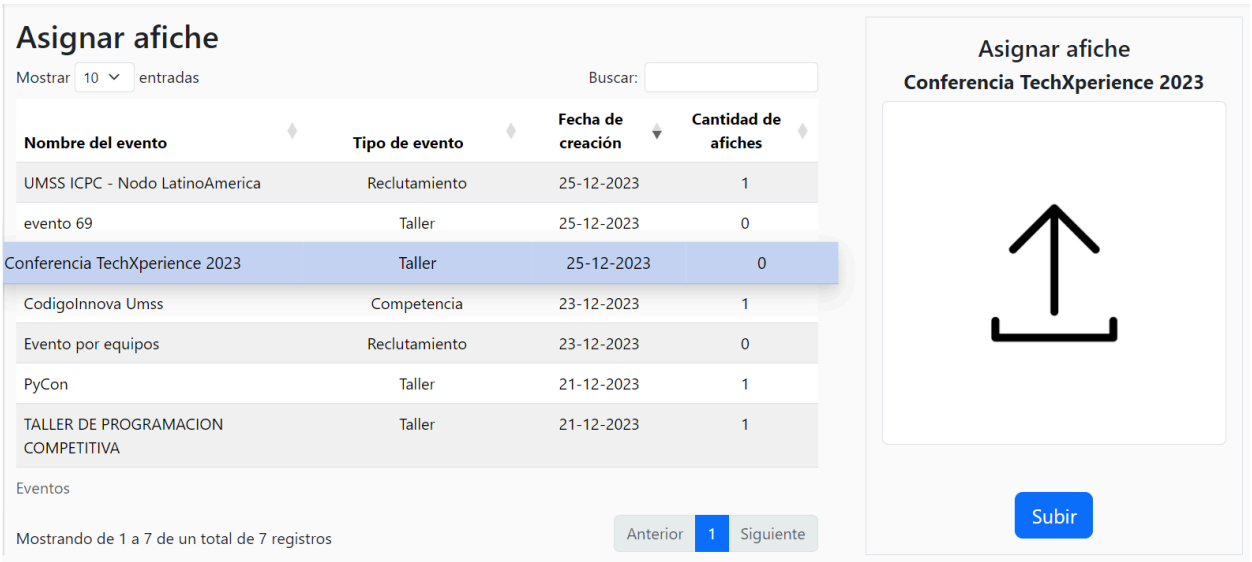
Justo debajo del previsualizador, se ubica el botón "Subir". Este botón solo está activo cuando se ha seleccionado un evento específico de la tabla, asegurando que la asignación de afiches sea coherente.

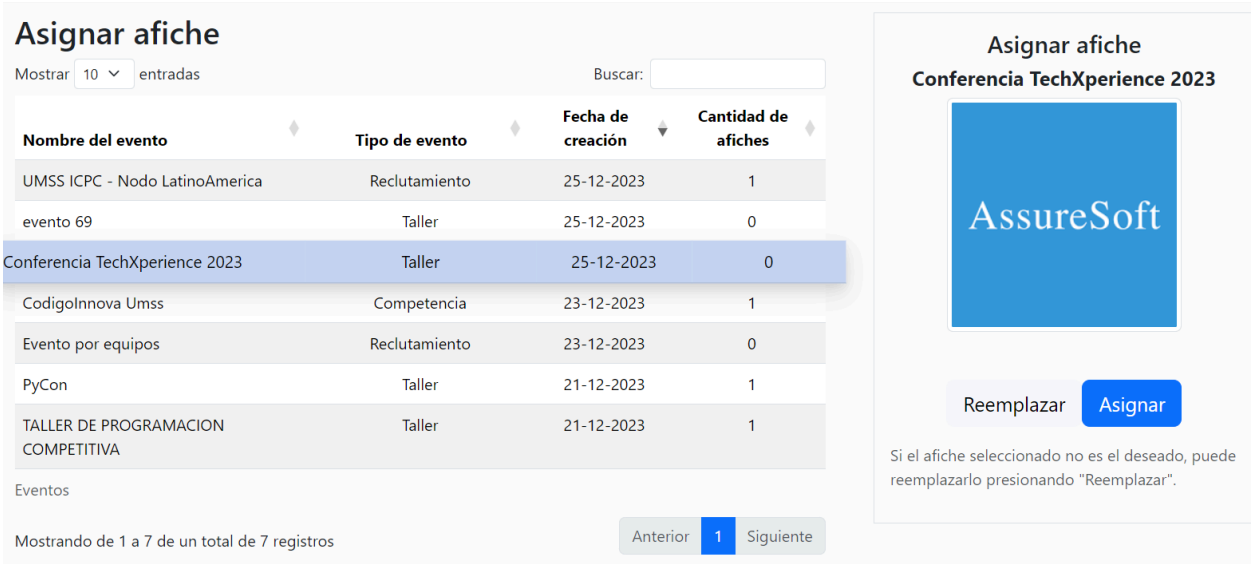
Botones "Reemplazar" y "Asignar":

Después de seleccionar una imagen y verificar su idoneidad, se revelan dos nuevos botones en la parte inferior del previsualizador: "Reemplazar" y "Asignar". Estos permiten al usuario decidir si desea sustituir la imagen actualmente asignada al evento o confirmar y asignar la nueva imagen seleccionada.

Mensaje Informativo:

Se incluye un mensaje informativo que guía al usuario en caso de que el afiche seleccionado no sea el deseado. Proporciona información sobre cómo proceder y ofrece claridad en las acciones disponibles.





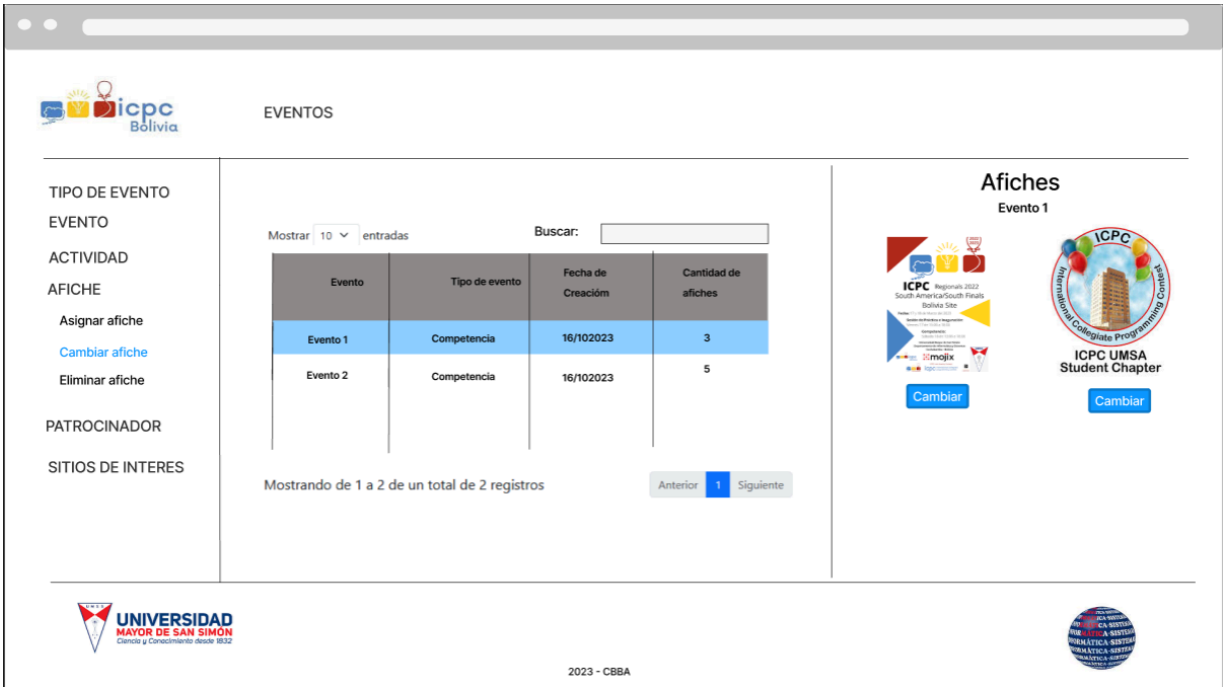
7.4.2. Cambiar afiche

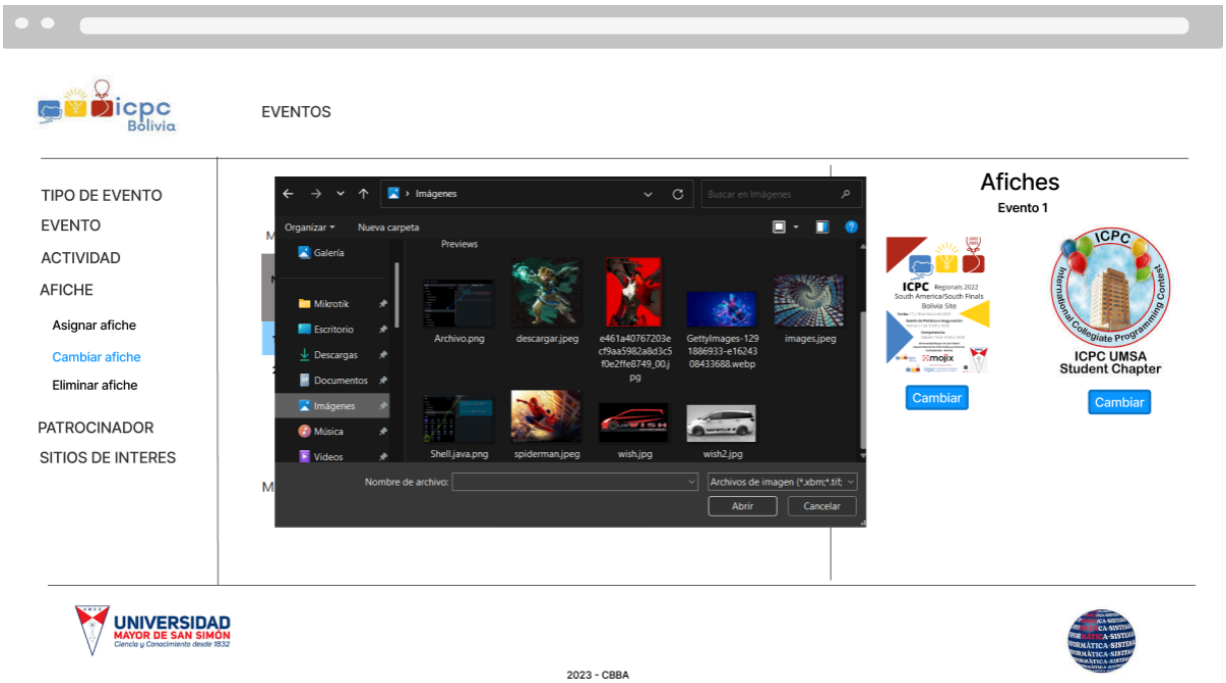
7.4.2.1. Historia de usuario de cambiar afiche

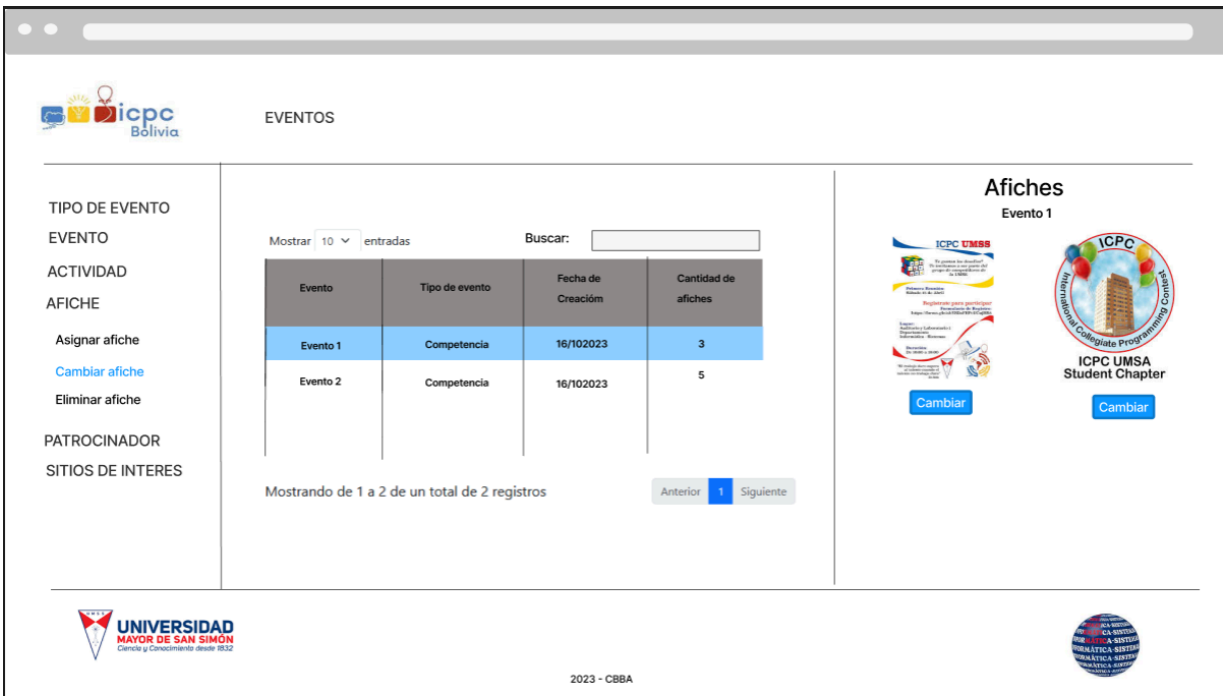
Historia de Usuario	
Título: Cambiar afiche de un evento	**ID: 11**
Estimación: 13	**Importancia: Media**
Descripción	

Como un usuario con privilegios para administrar eventos, quiero modificar los afiches asociados a un evento para garantizar que la información gráfica del evento sea precisa y actualizada.

Mockups







Criterios de aceptación.

1. Al hacer clic en la sección “AFICHE” en el menú lateral y seleccionar la opción “Cambiar afiche” se carga una tabla que muestra la información de los eventos creados previamente.
2. La tabla tiene las columnas “Nombre del evento”, “Tipo de evento”, “Fecha de creación” y “Cantidad de afiches”.
3. La columna “Cantidad de afiches” contiene la cantidad de afiches asociadas a un evento.
4. Cuando se selecciona un evento específico de la tabla, en la parte derecha se muestran los afiches que previamente se asignaron a ese evento.
5. Cada afiche se presenta con su imagen y un botón "Cambiar".
6. Al hacer clic en el botón "Cambiar" de un afiche, se muestra el gestor de archivos que permite al usuario cargar una nueva imagen para reemplazar el afiche actual.
7. Solo se permite formatos de archivo válidos, que incluyen JPG, JPEG y PNG.
8. El peso máximo de la imagen que se puede subir es de 5 MB.
9. Si se sube un archivo con un formato distinto al aceptado, sale una alerta temporal con el mensaje “Archivo no válido”.
10. Cuando el archivo es válido se muestra un modal de confirmación con el mensaje ¿Está seguro de cambiar el afiche?, con los botones “No” y “Sí”
11. Al presionar “Sí” aparece una alerta temporal con el mensaje “Actualizado exitosamente”.
12. Al hacer clic “Sí” se cierra el modal.

13. Después de cargar la nueva imagen y confirmar la acción, el afiche se actualiza con la nueva imagen proporcionada.
14. Al hacer clic "No", no se realiza ningún cambio y se cierra el modal.

7.4.2.2. Código fuente de cambiar afiche

Controlador de cambiar afiche

La función update en el controlador AficheController se encarga de cambiar el afiche asociado a un evento específico. Cuando se realiza una solicitud para actualizar un afiche mediante el método HTTP POST, la función busca el afiche actual relacionado con el evento mediante su identificador único. Posteriormente, elimina la imagen actual del afiche para dar paso a la nueva imagen proporcionada en la solicitud. La función utiliza la subrutina storageAfiche para almacenar la nueva imagen y obtener su ruta. Finalmente, guarda la actualización del afiche en la base de datos y retorna una respuesta JSON indicando el éxito de la operación. En caso de que ocurra una excepción durante el proceso, la función captura la excepción y retorna un mensaje de error.

Ruta del archivo: app/Http/Controllers/AficheController.php

```
/**
 * Cambia el afiche asociado a un evento específico.
 */
public function update(Request $request, $id)
{
    try {
        // Busca el afiche actual asociado al evento por su ID.
        $afiche = Afiche::find($id);

        // Elimina la imagen actual del afiche.
        Storage::delete(storage_path($afiche->ruta_imagen));

        // Almacena la nueva imagen proporcionada en la solicitud y obtiene su ruta.
        $afiche->ruta_imagen = $this->storageAfiche($request);

        // Guarda la actualización del afiche en la base de datos.
        $afiche->save();

        // Retorna una respuesta JSON indicando el éxito de la operación.
        return response()->json(['mensaje' => 'Actualizado exitosamente', 'error' => false]);
    } catch (QueryException $e) {
        // En caso de error, retorna un mensaje de error.
        return $e->getMessage();
    }
}
```

Código JavaScript de cambiar afiche

Se encarga de gestionar la asignación y cambio de afiches en eventos a través de una interfaz interactiva. Utiliza la biblioteca DataTable para presentar una tabla de eventos con información relevante, como el nombre del evento y la cantidad de afiches asociados. Al cargar la página, se inicializa DataTable con opciones de configuración predefinidas. La selección de un evento desencadena la carga dinámica de sus afiches asociados, presentándolos en tarjetas con la opción de cambiar la imagen. El código utiliza

funciones para previsualizar y validar las imágenes antes de realizar cambios. Al seleccionar una imagen, se muestra una previsualización y se permite al usuario cambiarla. Además, se incorporan funciones para asignar un nuevo afiche a un evento y para cancelar la subida de una imagen. El diseño se ajusta dinámicamente utilizando un observador de cambios en la altura del contenedor de la tabla de eventos, asegurando una presentación adecuada en diferentes dispositivos.

Ruta del archivo: public/js/Afiche/editarAfiche.js

```
// Elementos de la interfaz
const eventoSeleccionado = document.getElementById("nombreEvento");
const contenedorAsignar = document.getElementById("contenedorAsignar");
// Variables relacionadas con la tabla
let tablaDeTipos;
let tablaInicializada = false;
// Opciones de configuración para DataTable
const dataTableOptions = {
    pageLength: 10,
    lengthMenu: [5, 10, 15, 20],
    destroy: true,
    order: [[2, "desc"]],
    language: {
        // Configuración del idioma para DataTable
        lengthMenu: "Mostrar _MENU_ entradas",
        zeroRecords: "Ningún tipo de evento encontrado",
        info: "Mostrando de _START_ a _END_ de un total de _TOTAL_ registros",
        infoEmpty: "Ningún usuario encontrado",
        infoFiltered: "(filtrados desde _MAX_ registros totales)",
        search: "Buscar:",
        loadingRecords: "Cargando...",
        paginate: {
            first: "Primero",
            last: "Último",
            next: "Siguiente",
            previous: "Anterior",
        },
    },
};
// Inicialización de DataTable
const initDataTable = async () => {
    if (tablaInicializada) {
        tablaDeTipos.destroy();
    }
    DataTable.datetime("DD-MM-YYYY");
    tablaDeTipos = $("#tablaEvento").DataTable(dataTableOptions);
    tablaInicializada = true;
};
```

```
// Evento al cargar la página
window.addEventListener("load", () => {
    initDataTable();
    if (!seleccionado) {
        eventoSeleccionado.textContent = "Selecciona un evento";
    }
});
// Variables para el manejo de eventos seleccionados
let seleccionado;
let idEvento;
// Función para seleccionar un evento
const seleccionarEvento = (afiche) => {
    if (seleccionado) {
        seleccionado.classList.remove("table-primary");
        contenedorAsignar.innerHTML = "";
    }
    seleccionado = document.getElementById(afiche.id);
    seleccionado.classList.add("table-primary");
    idEvento = afiche.id;
    eventoSeleccionado.textContent = afiche.nombre;
    cambiarEvento();
};
// Función para cambiar el evento seleccionado
const cambiarEvento = async () => {
    const response = await cargarAfiche();
    response.data.map((afiche) => {
        contenedorAsignar.innerHTML += `<div class="col-auto" id="tarjetaAfiche${afiche.id}">
        <div class="card" style="width: 10rem;">
            <img src="${afiche.ruta_imagen}" class="card-img-top" alt="Afiche" id=
            "imagenAfichepreview${afiche.id}">
            <div class="card-body d-flex justify-content-around gap-2">
                <input type="file" id="imageUpload${afiche.id}" style="display: none;
                "onchange="previsualizarImagen(event,${afiche.id})" accept="image/jpeg,image/png,
                image/jpg">
                <button class="btn btn-primary btn-sm" onclick="document.getElementById(
                'imageUpload${afiche.id}').click()">Cambiar</button>
            </div>
        </div>
    </div>`;
    });
```

```
};

// Variables para el manejo de la previsualización y carga de imágenes
let aficheSeleccion;
let rutaSeleccion;

// Función para validar la imagen seleccionada
const validarImagen = (input, peso, callback) => {
    if (input.files.length > 0) {
        const imagen = input.files[0];
        const maxFileSize = peso * 1024 * 1024;
        let mensaje = { mensaje: "", error: false };

        const type = !/image\/(png|jpeg|jpg)/.test(imagen.type);

        if (type || imagen.size > maxFileSize) {
            input.value = "";
            mensaje = { mensaje: "Archivo no válido", error: true };
        }

        if (typeof callback === "function") {
            callback(mensaje);
        }
    }
};

// Función para previsualizar la imagen antes de cambiarla
const previsualizarImagen = (event, idAfiche) => {
    const input = document.getElementById(`imageUpload${idAfiche}`);
    validarImagen(input, 5, (mensaje) => {
        if (!mensaje.error) {
            const file = event.target.files[0];
            const reader = new FileReader();
            reader.onload = (e) => {
                const imagen = document.getElementById(
                    `imagenAfichepreview${idAfiche}`
                );
                rutaSeleccion = imagen.src;
                imagen.src = e.target.result;
            };
```

```
            reader.readAsDataURL(file);
            aficheSeleccion = idAfiche;
            $('#modalCambiarAfiche').modal('show');
        } else {
            mostrarAlerta("Error", mensaje.mensaje, "danger");
        }
    });
};

// Función para cambiar la imagen del afiche
const cambiarAfiche = async () => {
    if (aficheSeleccion) {
        const input = document.getElementById(`imageUpload${aficheSeleccion}`);
        const form = new FormData();
        form.append("afiche", input.files[0]);
        const response = await axios.post(
            `/api/afiche/${aficheSeleccion}`,
            form
        );
        mostrarAlerta(
            "Éxito",
            response.data.mensaje,
            response.error ? "danger" : "success"
        );
    }
};

// Función para asignar un afiche a un evento
const asignarAfiche = async () => {
    if (idEvento) {
        const form = new FormData();
        form.append("afiche", input.files[0]);
        form.append("id_evento", idEvento);
        await axios.post("/api/afiche", form).then((response) => {
            mostrarAlerta(
                "Éxito",
                response.data.mensaje,
                response.error ? "danger" : "success"
            );
        });
```

```
    }
};

// Función para cancelar la subida de un afiche
const cancelarSubidaAfiche = async () => {
    if (aficheSeleccion && rutaSeleccion) {
        const imagen = document.getElementById(
            `imagenAfichepreview${aficheSeleccion}`
        );
        imagen.src = rutaSeleccion;
        aficheSeleccion = null;
        rutaSeleccion = null;
    }
};

// Función para seleccionar un afiche
const seleccionarAfiche = (id) => {
    aficheSeleccion = id;
};

// Función para cargar el afiche actual del evento
const cargarAfiche = async () => {
    const response = await axios.get(`/api/afiche/${idEvento}`);
    return response;
};

// Observador para ajustar la altura máxima del contenedorAsignar
const resize_ob = new ResizeObserver(function (entries) {
    let rect = entries[0].contentRect;
    let height = rect.height;
    let vh = parseInt((height / window.innerHeight) * 89);
    document.getElementById("contenedorAsignar").style.maxHeight = vh + "vh";
});

// Observar cambios en el contenedor de la tabla de eventos
resize_ob.observe(document.getElementById("tablaEventos"));
```

Vista de cambiar afiche

La interfaz consta de dos secciones principales. La columna izquierda presenta una tabla que lista los eventos disponibles con detalles como el nombre, tipo, fecha de creación y la cantidad de afiches asociados. Al hacer clic en un evento específico de la tabla, se invoca la función seleccionarEvento, que destaca la fila seleccionada y carga los afiches asociados al evento en la columna derecha.

La columna derecha muestra el nombre del evento seleccionado y un contenedor dinámico para visualizar las tarjetas de cambio de afiche. Cada tarjeta incluye una imagen previsualizada del afiche actual y un botón para cambiarlo. Al hacer clic en el botón "Cambiar", se activa un modal de confirmación. Además, existe un modal de confirmación general denominado 'modalCambiarAfiche'. El script utiliza DataTable para mejorar la presentación y funcionalidad de la tabla de eventos. En cuanto a la lógica de interacción, se destacan funciones como cambiarAfiche, asignarAfiche, cancelarSubidaAfiche, y previsualizarImagen para gestionar el cambio de afiche, asignación de nuevos afiches, cancelación de cambios y previsualización de imágenes, respectivamente.

Ruta del archivo: resources/views/afiche/editarAfiche.blade.php

```
@extends('layouts.app')

@section('content')
    <div class="container">
        <div class="row g-5">
            <!-- Columna izquierda con la tabla de eventos -->
            <div class="col-sm-12 col-md-8" id="tablaEventos">
                <h2>Cambiar afiche</h2>
                <table class="table table-responsive table-striped text-secondary table-hover cursor" id="tablaEvento">
                    <caption>Eventos</caption>
                    <thead>
                        <tr>
                            <th scope="col" class="col-sm-4 col-md-4">Nombre del evento</th>
                            <th scope="col" class="col-sm-0 col-md-3 text-center">Tipo de evento</th>
                            <th scope="col" class="col-sm-3 col-md-2 text-center">Fecha de creación</th>
                            <th scope="col" class="col-sm-3 col-md-2 text-center font-sm">Cantidad de afiches</th>
                        </tr>
                    </thead>
                    <tbody id="datosTabla">
                        <!-- Iteración sobre los eventos para mostrar en la tabla -->
                        @foreach ($afiches as $afiche)
                            @if (strtotime($afiche->fin_evento) >= time())
                                <tr onclick="seleccionarEvento({{ $afiche }})" id="{{ $afiche->id }}">
                                    <td>{{ $afiche->nombre }}</td>
                                    <td class="text-center">{{ $afiche->tipoEvento->nombre }}</td>
                                    <td class="text-center">{{ date('d-m-Y', strtotime($afiche->created_at)) }}</td>
                                    <td class="text-center" id="contadorAfiches{{ $afiche->id }}">
                                        {{ $afiche->afiches->count() }}</td>
                                </tr>
                            @endif
                        @endforeach
                    </tbody>
                </table>
            </div>
            <!-- Columna derecha con la sección de cambio de afiche -->
            <div class="col-sm-12 col-md-4">
                <div class="container">
                    <div class="col-12 d-flex justify-content-center align-items-center">
                        <h4>Afiches</h4>
                    </div>
                    <!-- Mostrar el nombre del evento seleccionado -->
                    <h5 id="nombreEvento" class="text-center fw-bold"></h5>
                    <!-- Contenedor dinámico para mostrar tarjetas de cambio de afiche -->
                    <div class="row gap-2 mt-2 d-flex justify-content-center pb-2" style="overflow-y: auto; overflow-x: hidden" id="contenedorAsignar"></div>
                    <!-- Modal de confirmación para cambiar el afiche -->
                    @component('components.modal')
                        @slot('modalId', 'modalCambiarAfiche')
                        @slot('modalTitle', 'Confirmación')
                        @slot('modalContent')
                            ¿Está seguro de cambiar el afiche?
                        @endslot
                        @slot('modalButton')
                            <button type="button" class="btn btn-secondary w-25 mx-8" data-bs-dismiss="modal" onclick="cancelarSubidaAfiche()">No</button>
                            <button type="reset" class="btn btn-primary w-25 mx-8" data-bs-dismiss="modal" onclick="cambiarAfiche()">Sí</button>
                        @endslot
                    @endcomponent
                </div>
            </div>
        </div>
    </div>

    <!-- Enlaces a bibliotecas externas para DataTable y scripts -->
    <link href="https://cdn.datatables.net/1.13.6/css/dataTables.bootstrap5.min.css" rel="stylesheet">
    <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.7.1/jquery.min.js"></script>
    <script src="https://cdn.datatables.net/1.13.6/js/jquery.dataTables.min.js"></script>
    <script src="https://cdn.datatables.net/1.13.6/js/dataTables.bootstrap5.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/moment.js/2.29.2/moment.min.js"></script>
    <!-- Script personalizado para esta vista -->
    <script src="{{ asset('js/Afiche/editarAfiche.js') }}" defer></script>
@endsection
```

7.4.3.3. Interfaz de usuario de cambiar afiche

La interfaz de usuario para cambiar el afiche de un evento presenta una estructura intuitiva dividida en dos secciones principales:

Tabla de Eventos

En la columna izquierda, se encuentra una tabla detallada que enumera los eventos disponibles. Cada fila de la tabla proporciona información relevante, como el nombre del evento, el tipo de evento, la fecha de creación y la cantidad actual de afiches asociados. Al hacer clic en un evento específico, se activa la

función seleccionarEvento, que resalta la fila seleccionada y carga los afiches asociados en la columna derecha.

Tarjetas de Cambio de Afiche

En la columna derecha, se presenta el nombre del evento seleccionado y un área dinámica para visualizar las tarjetas de cambio de afiche. Cada tarjeta muestra una previsualización del afiche actual y proporciona un botón "Cambiar". Al hacer clic en este botón, se despliega un modal de confirmación ('modalCambiarAfiche') para asegurar la confirmación del cambio.

Modal de Confirmación para Cambiar Afiche

El modal de confirmación se activa al hacer clic en el botón "Cambiar" en una tarjeta de cambio de afiche. Este modal presenta un mensaje claro preguntando al usuario si está seguro de cambiar el afiche. Se proporcionan botones "No" y "Sí" para permitir al usuario confirmar o cancelar el cambio de afiche.

Proceso de Cambio de Afiche

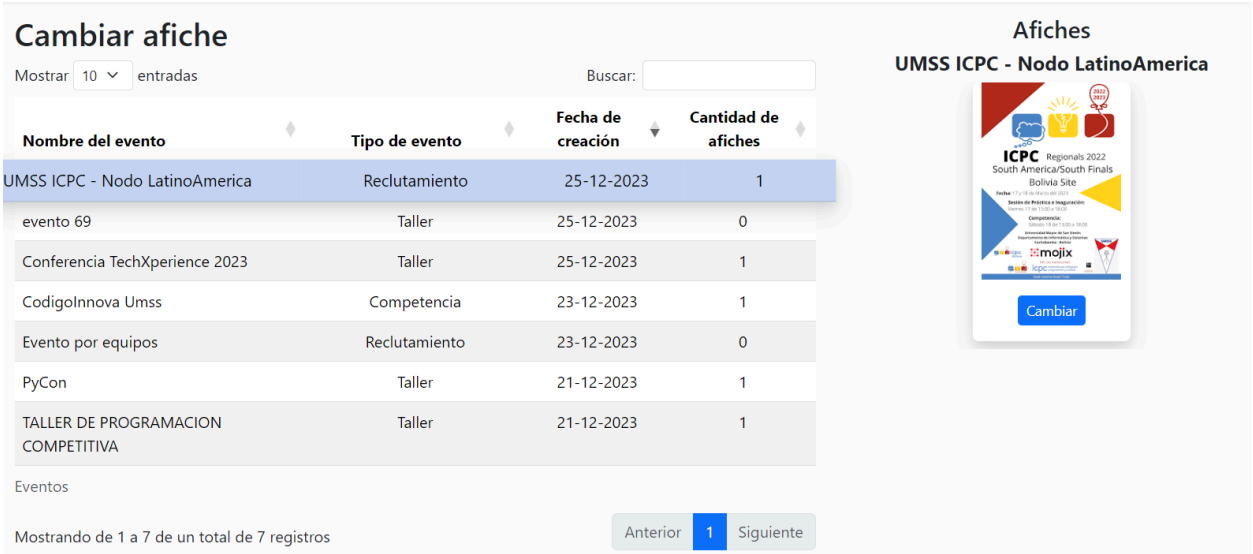
Si el usuario hace clic en "Sí" en el modal de confirmación, se activa la función cambiarAfiche, que realiza la actualización del afiche seleccionado. Se muestra una alerta indicando el éxito o cualquier error durante el proceso de cambio.

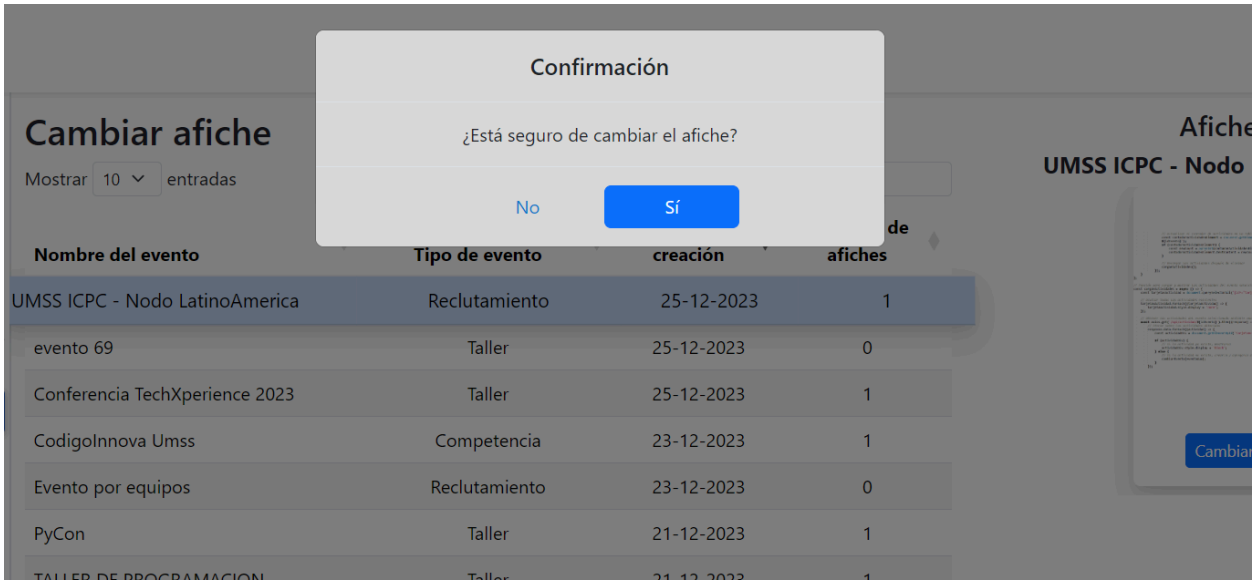
Interacción del Usuario

La tabla de eventos facilita la identificación rápida de la información clave.

Las tarjetas de cambio de afiche ofrecen detalles específicos, y el botón "Cambiar" proporciona una opción directa para cambiar el afiche.

La inclusión de un modal de confirmación garantiza que el usuario tome decisiones informadas antes de proceder con el cambio de afiche.

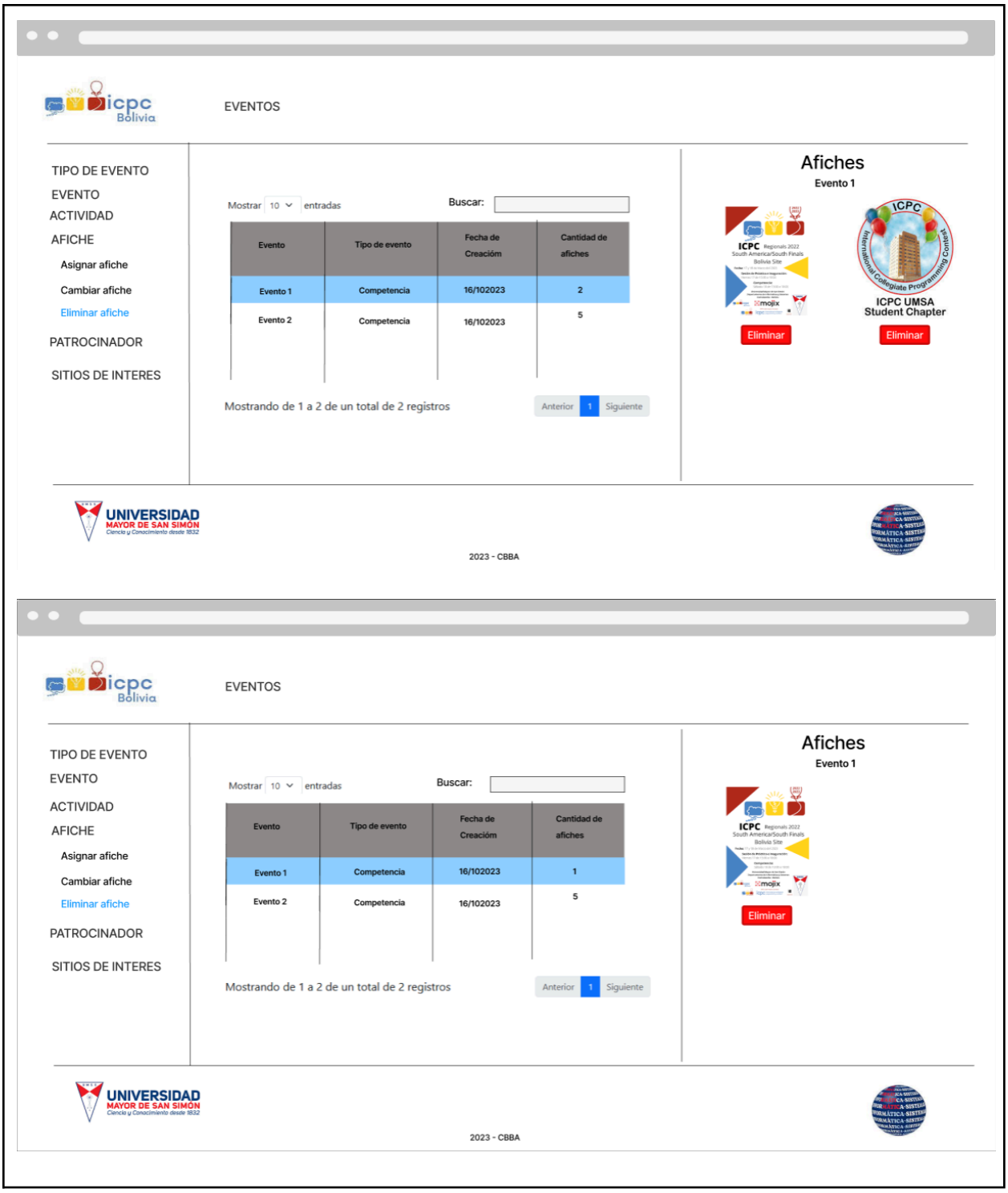


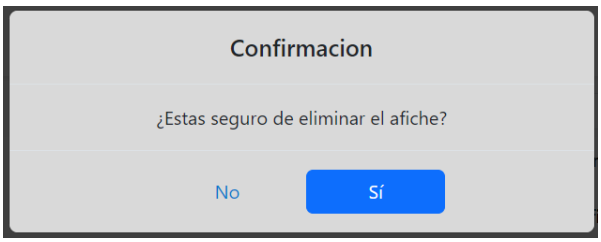


7.4.3. Eliminar afiche

7.4.3.1. Historia de usuario de eliminar afiche

Historia de Usuario	
Título: Eliminar afiche de un evento	**ID: 20**
Estimación: 3	**Importancia: Media**
Descripción: Como un usuario con privilegio para administrar eventos, necesito la capacidad de eliminar el/los afiche(s) asociado(s) a un evento para garantizar que la información gráfica del evento que ya no se requiera se pueda quitar.	
Mockups	


icpc Bolivia
EVENTOS
TIPO DE EVENTO
EVENTO
ACTIVIDAD
AFICHE
Asignar afiche
Cambiar afiche
Eliminar afiche
PATROCINADOR
SITIOS DE INTERES
Mostrar 10 entradas
Buscar:
Evento
Tipo de evento
Fecha de Creación
Cantidad de afiches
Evento 1
Competencia
16/102023
2
Evento 2
Competencia
16/102023
5
Mostrando de 1 a 2 de un total de 2 registros
Anterior 1 Siguiente
Afiches
Evento 1
Eliminar
ICPC UMSA Student Chapter
Eliminar
UNIVERSIDAD MAYOR DE SAN SIMÓN
2023 - CBBA
icpc Bolivia
EVENTOS
TIPO DE EVENTO
EVENTO
ACTIVIDAD
AFICHE
Asignar afiche
Cambiar afiche
Eliminar afiche
PATROCINADOR
SITIOS DE INTERES
Mostrar 10 entradas
Buscar:
Evento
Tipo de evento
Fecha de Creación
Cantidad de afiches
Evento 1
Competencia
16/102023
1
Evento 2
Competencia
16/102023
5
Mostrando de 1 a 2 de un total de 2 registros
Anterior 1 Siguiente
Afiches
Evento 1
Eliminar
UNIVERSIDAD MAYOR DE SAN SIMÓN
2023 - CBBA



Criterios de aceptación.

1. Al hacer clic en la sección “AFICHE” en el menú lateral y seleccionar la opción “Eliminar afiche” se muestra una tabla con los eventos creados previamente.
2. La tabla tiene las columnas “Nombre del evento”, “Tipo de evento”, “Fecha de creación” y “Cantidad de afiches”.
3. La columna “Cantidad de afiches” contiene la cantidad de afiches asociados a un evento.
4. Cuando se selecciona un evento específico de la tabla, en la parte derecha se muestran los afiches que previamente se asignaron a ese evento.
5. Cada afiche se presenta con su imagen y un botón "Eliminar".
6. Al hacer clic en el botón “Eliminar” de algún afiche se muestra un modal con el mensaje “¿Está seguro de eliminar el afiche?”.
7. El modal presenta dos botones: "No" y "Sí".
8. Al hacer clic en el botón "No", no se elimina el afiche.
9. Al hacer clic en el botón "No", se cierra el modal.
10. Al hacer clic en el botón "Sí" se elimina el afiche.
11. Al hacer clic en el botón "Sí" se cierra el modal.
12. Al eliminar un afiche, se muestra una alerta temporal con el mensaje “Eliminado exitosamente”.
13. Al quitar un afiche, el valor en la columna “Cantidad de afiches” se reduce en 1.

7.4.3.2. Código fuente de eliminar afiche

Controlador de eliminar afiche

La función eliminarAfiche en el controlador AficheController de Laravel se encarga de eliminar un afiche y su archivo asociado del sistema de archivos y de la base de datos. Primero, busca el afiche correspondiente utilizando su ID. Luego, borra el archivo del afiche del sistema de archivos utilizando el servicio de almacenamiento de Laravel (Storage::delete). Posteriormente, elimina el registro del afiche de la base de datos. La función devuelve una respuesta en formato JSON, indicando el resultado de la operación (éxito o error) y un mensaje descriptivo. En caso de éxito, el mensaje informa que la eliminación se realizó correctamente, mientras que en caso de error, el mensaje proporciona detalles sobre la excepción capturada durante el proceso.

Ruta del archivo: app/Http/Controllers/AficheController.php

```php
/**
 * Elimina un afiche y su archivo asociado.
 */
public function eliminarAfiche($id)
{
    try {
        // Busca el afiche por su ID
        $afiche = Afiche::find($id);

        // Elimina el archivo del afiche en el sistema de archivos
        Storage::delete(storage_path($afiche->ruta_imagen));

        // Elimina el afiche de la base de datos
        $afiche->delete();

        // Retorna una respuesta JSON indicando éxito y un mensaje informativo
        return response()->json(['mensaje' => 'Eliminado exitosamente', 'error' => false]);
    } catch (QueryException $e) {
        // En caso de error, retorna un mensaje de error en formato JSON
        return response()->json(['mensaje' => $e->getMessage(), 'error' => true]);
    }
}
```

Código JavaScript de eliminar afiche

El código proporcionado implementa la interfaz de usuario para la gestión de afiches en eventos, con funcionalidades de asignación y eliminación. Utiliza DataTable para presentar una tabla de eventos con detalles, como nombre, tipo, fecha de creación y cantidad de afiches. Al seleccionar un evento, se activa una función que carga y muestra tarjetas detalladas de cada afiche vinculado al evento. Cada tarjeta incluye una imagen del afiche y un botón "Eliminar". Además, se emplea un modal de confirmación para asegurar la decisión del usuario antes de eliminar un afiche. El código utiliza Axios para realizar solicitudes a la API y actualizar dinámicamente la interfaz de usuario. También se incluyen funciones para cargar afiches, cambiar la visualización de afiches en el contenedor, y gestionar eventos de eliminación. El diseño responsivo se aborda mediante el uso de ResizeObserver para ajustar la altura del contenedor según el tamaño de la ventana.

Ruta del archivo: public/js/Afiche/eliminarAfiche.js

```
// Obtener referencias a elementos del DOM
const eventoSeleccionado = document.getElementById("nombreEvento");
const contenedorAsignar = document.getElementById("contenedorAsignar");
// Variables para gestionar la tabla DataTable
let tablaDeTipos;
let tablaInicializada = false;
// Opciones de configuración para DataTable
const dataTableOptions = {
    pageLength: 10,
    lengthMenu: [5, 10, 15, 20],
    destroy: true,
    order: [[2, "desc"]],
    language: {
        // Configuración de idioma para DataTable
        lengthMenu: "Mostrar _MENU_ entradas",
        zeroRecords: "Ningún tipo de evento encontrado",
        info: "Mostrando de _START_ a _END_ de un total de _TOTAL_ registros",
        infoEmpty: "Ningún usuario encontrado",
        infoFiltered: "(filtrados desde _MAX_ registros totales)",
        search: "Buscar:",
        loadingRecords: "Cargando...",
        paginate: {
            first: "Primero",
            last: "Último",
            next: "Siguiente",
            previous: "Anterior",
        },
    },
};

// Función para inicializar DataTable
const initDataTable = async () => {
    if (tablaInicializada) {
        tablaDeTipos.destroy();
    }
    DataTable.datetime("DD-MM-YYYY");
    tablaDeTipos = $("#tablaEvento").DataTable(dataTableOptions);
    tablaInicializada = true;
};
```

```
// Evento al cargar la página
window.addEventListener("load", () => {
    initDataTable();
    if (!seleccionado) {
        eventoSeleccionado.textContent = "Selecciona un evento";
    }
});
// Variables para gestionar el evento seleccionado y su ID
let seleccionado;
let idEvento;
// Función para seleccionar un evento y cargar sus afiches
const seleccionarEvento = (afiche) => {
    if (seleccionado) {
        seleccionado.classList.remove("table-primary");
        contenedorAsignar.innerHTML = "";
    }
    seleccionado = document.getElementById(afiche.id);
    seleccionado.classList.add("table-primary");
    idEvento = afiche.id;
    eventoSeleccionado.textContent = afiche.nombre;
    cargarAfiche();
};

// Función para obtener los afiches del evento mediante una solicitud a la API
const getAfichesDelEvento = async () => {
    let afiches = await axios.get("/api/afiche/" + idEvento).then(response => {
        return response.data;
    });
    return afiches;
};
// Función para cambiar la visualización de los afiches en el contenedor
const cambiarEvento = (afiches) => {
    // Actualizar la cantidad de afiches en la interfaz
    document.getElementById(`contadorAfiches${idEvento}`).textContent = afiches.length;
    let content = "";
    // Construir tarjetas de afiches en el contenedor
    afiches.map((afiche) => {
        content += `<div class="col-auto" id="tarjetaAfiche${afiche.id}">
```

```
            <div class="card" style="width: 10rem;">
                <img src="${afiche.ruta_imagen}" class="card-img-top" alt="Afiche" id=
                "imagenAfichepreview${afiche.id}">
                <div class="card-body d-flex justify-content-around gap-2">
                    <input type="file" id="imageUpload${afiche.id}" style="display: none;"
                    onchange="previsualizarImagen(event, ${afiche.id})" accept="image/jpeg, image/png,
                    image/jpg">
                    <a href="#" class="btn btn-danger btn-sm" data-bs-toggle="modal"
                    data-bs-target="#modalEliminarAfiche" onclick="seleccionarAfiche(${afiche.id})">
                    Eliminar</a>
                </div>
            </div>
        </div>`;
        });
        // Actualizar el contenido del contenedor con las tarjetas de afiches
        contenedorAsignar.innerHTML = content;
};

// Variable para almacenar el ID del afiche seleccionado para eliminar
let aficheSeleccion;

// Función para seleccionar un afiche antes de eliminarlo
const seleccionarAfiche = (id) => {
    aficheSeleccion = id;
};

// Función para eliminar un afiche
const eliminarAfiche = async () => {
    if (aficheSeleccion) {
        await axios
            .delete(`/api/afiche/${aficheSeleccion}`)
            .then((response) => {
                // Mostrar una alerta con el resultado de la operación y recargar los afiches
                mostrarAlerta(
                    "Éxito",
                    response.data.mensaje,
                    response.error ? "danger" : "success"
                );
                cargarAfiche();
            });
    }
};

// Función para cargar los afiches del evento seleccionado
const cargarAfiche = async () => {
    let afiches = await getAfichesDelEvento();
    // Actualizar la visualización de los afiches en el contenedor
    cambiarEvento(afiches);
};

// Observador de cambios en el tamaño de la ventana para ajustar la altura del contenedor
const resize_ob = new ResizeObserver(function (entries) {
    let rect = entries[0].contentRect;
    let height = rect.height;
    vh = parseInt((height / window.innerHeight) * 89);
    document.getElementById("contenedorAsignar").style.maxHeight = vh + "vh";
});

// Observar cambios en el tamaño de la tabla de eventos
resize_ob.observe(document.getElementById("tablaEventos"));
```

Vista de eliminar afiche

La vista presenta una tabla de eventos, donde cada fila representa un evento con detalles como nombre, tipo, fecha de creación y cantidad de afiches. Solo se muestran eventos cuya fecha de finalización es posterior a la fecha actual.

La columna de "Eliminar Afiche" permite al usuario seleccionar un evento y abrir un modal de confirmación para eliminar el afiche asociado. La barra lateral derecha muestra información adicional sobre los afiches seleccionados, incluido el nombre del evento. Se utiliza DataTables para mejorar la funcionalidad de la tabla, y se incluye un script personalizado (eliminarAfiche.js) para manejar la lógica de eliminación.

Ruta del archivo: resources/views/afiche/eliminarAfiche.blade.php

```
@extends('layouts.app')

@section('content')
    <div class="container">
        <div class="row g-5">
            <!-- Columna principal (Tabla de Eventos) -->
            <div class="col-sm-12 col-md-8" id="tablaEventos">
                <h2>Eliminar afiche</h2>
                <!-- Tabla que muestra información sobre eventos -->
                <table class="table table-responsive table-striped text-secondary table-hover cursor" id="tablaEvento">
                    <caption>Eventos</caption>
                    <thead>
                        <tr>
                            <th scope="col" class="col-sm-4 col-md-4">Nombre del evento</th>
                            <th scope="col" class="col-sm-0 col-md-3 text-center">Tipo de evento</th>
                            <th scope="col" class="col-sm-3 col-md-2 text-center">Fecha de creación</th>
                            <th scope="col" class="col-sm-3 col-md-2 text-center font-sm">Cantidad de afiches</th>
                        </tr>
                    </thead>
                    <tbody id="datosTabla">
                        <!-- Bucle que recorre los eventos (afiches) -->
                        @foreach ($afiches as $afiche)
                            <!-- Solo muestra eventos cuya fecha de finalización es posterior a la fecha actual -->
                            @if (strtotime($afiche->fin_evento) >= time())
                                <!-- Fila de la tabla con información del evento -->
                                <tr onclick="seleccionarEvento({{ $afiche }})" id="{{ $afiche->id }}">
                                    <td>{{ $afiche->nombre }}</td>
                                    <td class="text-center">{{ $afiche->tipoEvento->nombre }}</td>
                                    <td class="text-center">{{ date('d-m-Y', strtotime($afiche->created_at)) }}</td>
                                    <td class="text-center" id="contadorAfiches{{ $afiche->id }}">
```

```
                                {{ $afiche->afiches->count() }}</td>
                            </tr>
                        @endif
                    @endforeach
                </tbody>
            </table>
        </div>

        <!-- Columna lateral derecha (Detalles y Confirmación de Eliminación) -->
        <div class="col-sm-12 col-md-4">
            <div class="container ">
                <!-- Título y nombre del evento seleccionado -->
                <div class="col-12 d-flex justify-content-center align-items-center">
                    <h4>Afiches</h4>
                </div>
                <h5 id="nombreEvento" class="text-center fw-bold"></h5>

                <!-- Contenedor para mostrar detalles de afiches seleccionados -->
                <div class="row gap-2 mt-2 d-flex justify-content-center pb-2" style="overflow-y: auto; overflow-x: hidden"
                    id="contenedorAsignar">
                    <!-- Detalles de afiches se cargarán aquí -->
                </div>

                <!-- Componente Modal para confirmar la eliminación de un afiche -->
                @component('components.modal')
                    @slot('modalId', 'modalEliminarAfiche')
                    @slot('modalTitle', 'Confirmación')
                    @slot('modalContent')
                        ¿Está seguro de eliminar el afiche?
                    @endslot
                    -
                    @slot('modalButton')
                        <button type="button" class="btn btn-secondary w-25 mx-8" data-bs-dismiss="modal">No</button>
                        <button type="reset" class="btn btn-danger w-25 mx-8" data-bs-dismiss="modal"
                            onclick="eliminarAfiche()">Sí</button>
                    @endslot
                @endcomponent
            </div>
        </div>
    </div>
</div>

<!-- Enlaces a estilos y scripts externos -->
<link href="https://cdn.datatables.net/1.13.6/css/dataTables.bootstrap5.min.css" rel="stylesheet">
<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.7.1/jquery.min.js"></script>
<script src="https://cdn.datatables.net/1.13.6/js/jquery.dataTables.min.js"></script>
<script src="https://cdn.datatables.net/1.13.6/js/dataTables.bootstrap5.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/moment.js/2.29.2/moment.min.js"></script>
<script src="{{ asset('js/Afiche/eliminarAfiche.js') }}" defer></script>
@endsection
```

7.4.3.3. Interfaz de usuario de eliminar afiche

La interfaz de usuario destinada a la eliminación de afiches de eventos proporciona una experiencia clara y accesible para la gestión de afiches asociados a eventos existentes. Aquí se describen los elementos clave de esta interfaz:

Tabla de Eventos:

Nombre del Evento: Identifica el nombre del evento.

Tipo de Evento: Muestra el tipo de evento asociado al evento.

Fecha de Creación: Presenta la fecha de creación del evento en formato "dd-mm-aa".

Cantidad de Afiches: Indica la cantidad de afiches asociados al evento.

Barra Lateral Derecha:

Título "Afiches": Indica la sección que mostrará detalles sobre los afiches seleccionados.

Nombre del Evento Seleccionado: Muestra el nombre del evento seleccionado.

Detalles de Afiches: Espacio que se llenará dinámicamente con las imagenes de los afiches seleccionados.

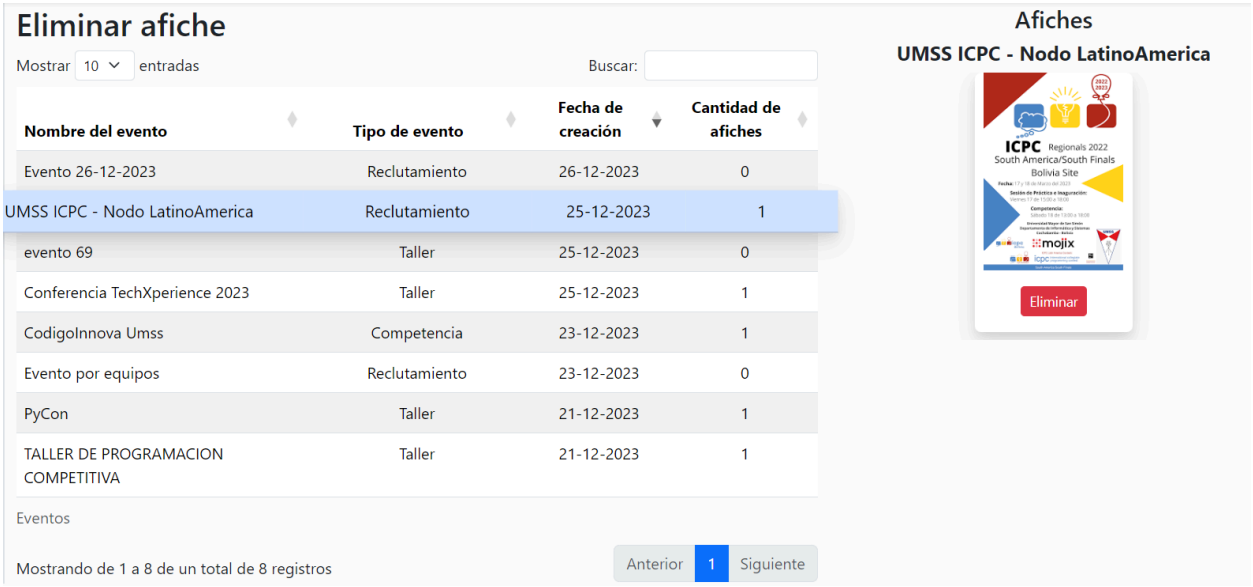
Modal de Confirmación: Se utiliza un modal con el título "Confirmación" al intentar eliminar un afiche, proporcionando botones "Sí" y "No" para confirmar la acción.

Interacción del Usuario:

Selección de Eventos: La tabla facilita la identificación rápida de eventos disponibles, y al hacer clic en una fila, se activa la opción para eliminar el afiche asociado.

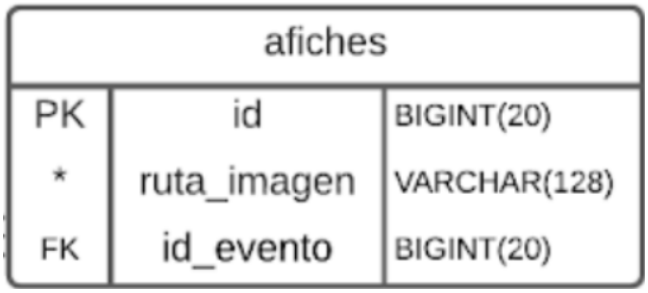
Usabilidad Mejorada: DataTables mejora la usabilidad al permitir ordenar y buscar eventos de manera eficiente.

Componente Modal: El modal de confirmación asegura que el usuario esté consciente de la acción a realizar, evitando eliminaciones accidentales.



7.4.4. Tabla de base de datos de afiche

La tabla 'afiches' en nuestra base de datos gestiona información visual esencial para cada evento. Cada afiche está identificado de manera única por su 'id' y se asocia con un evento específico a través de 'id_evento'. La 'ruta_imagen' señala la ubicación del archivo de imagen correspondiente, permitiendo una presentación gráfica y atractiva para cada evento. Esta estructura facilita el acceso eficiente a los afiches asociados a través de la relación directa con la información detallada de cada evento.



7.5. PATROCINADOR

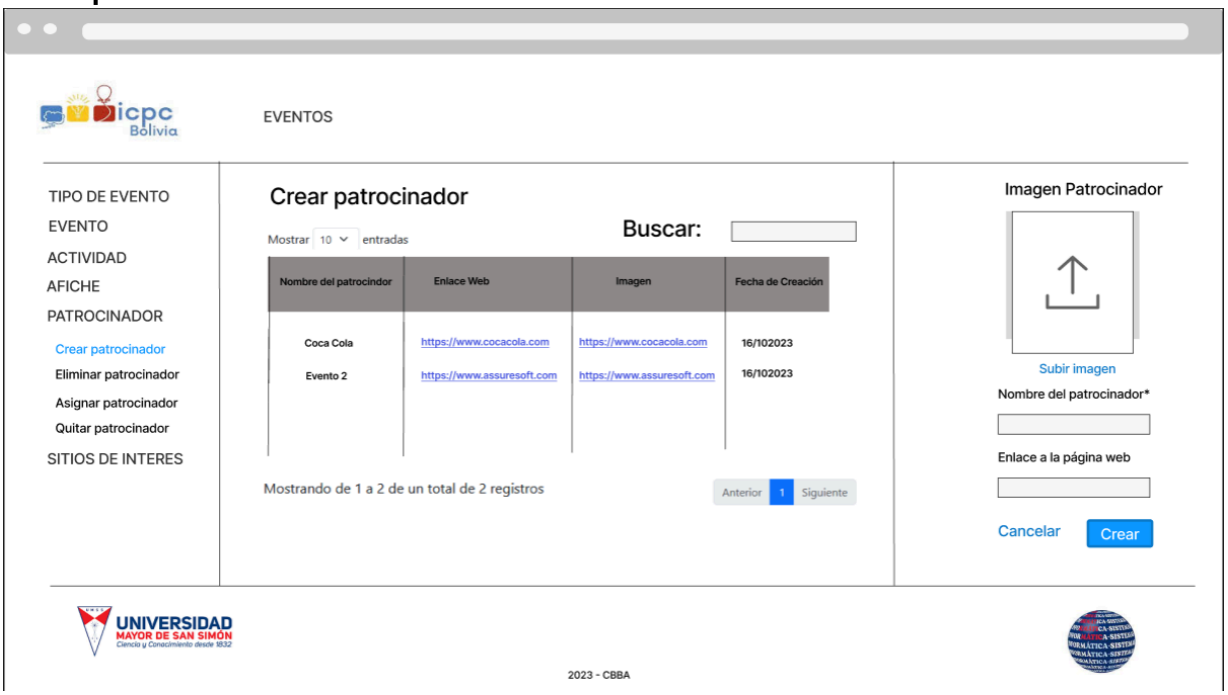
7.5.1. Crear patrocinador

7.5.1. Historia de usuario de crear patrocinador

Historia de Usuario	
Título: Crear patrocinador	**ID: 17**
Estimación: 3	**Importancia: Media**

Descripción
Como usuario con privilegios para la creación de patrocinador, quiero agregar nuevos patrocinadores para construir un registro de los patrocinadores, teniendo así una gestión más efectiva y proporcionándome la posibilidad de asignar estos patrocinadores a eventos específicos posteriormente.

Mockups



Criterios de Aceptación

1. Al hacer clic en la sección “PATROCINADOR” en el menú lateral y seleccionar la opción “Crear patrocinador” se muestra una tabla con los patrocinadores creados previamente y un formulario para crear un nuevo patrocinador.
2. La tabla tiene las columnas “Nombre del patrocinador”, ”Enlace web”, ”Imagen” y “Fecha de creación”.
3. En la columna "Imagen" de la tabla, se proporciona un enlace web que redirige a una ventana nueva donde se puede visualizar la imagen del patrocinador elegido.
4. El formulario para crear un patrocinador tiene los siguientes campos “Nombre del patrocinador *”, “Enlace a la página web del patrocinador”, y un botón para “Subir imagen”.
5. El campo “Nombre de patrocinador *” es obligatorio.
6. El campo “Enlace a la página web del patrocinador”, solo admite enlaces con las preposiciones “http://” o “https://”.
7. La imagen del patrocinador es obligatoria.
8. Si el peso de la imagen excede los 2 Mb, sale una alerta con el siguiente mensaje: “Archivo no válido”
9. Si el formato del archivo subido es diferente a png, jpeg o jpg, sale una alerta con el

siguiente mensaje: “Archivo no válido”.
10. Si la imagen se sube correctamente, esta se previsualiza en el recuadro ubicado encima del botón “Subir imagen”.
11. Al hacer clic en el botón “Cancelar”, la imagen subida y todos los campos retornan a sus valores por defecto.
12. Al hacer clic en el botón “Crear”, si un campo del formulario no cumple con alguna restricción, este campo se resalta en rojo.
13. Al crear un patrocinador, se muestra una alerta temporal con el mensaje “Creado exitosamente”.
14. Si el patrocinador se crea exitosamente, los campos del formulario regresan a su valor por defecto.
15. Si el nombre de patrocinador ingresado al momento de crear un patrocinador coincide con el nombre de un patrocinador ya existente, sale una alerta temporal con el siguiente mensaje: "El patrocinador ya existe".
16. Si el nombre de patrocinador ingresado al momento de crear un patrocinador coincide con el nombre de un patrocinador eliminado previamente, se muestra el modal "Patrocinador ya existente".
17. En el modal “Patrocinador ya existente” se muestra el siguiente mensaje “**El patrocinador que intentó crear ya existe,** ¿Desea habilitarlo nuevamente?” y los botones de "No" y "Sí".
18. Al hacer clic en el botón “No” el formulario regresa a sus valores por defecto.
19. Al hacer clic en el botón “No” el modal se cierra.
20. Al hacer clic en el botón “Si”, se muestra un modal con el título “Actualizar datos del patrocinador” con el mensaje “¿Desea actualizar los datos del patrocinador?” y los botones de "No" y "Sí"
21. Al hacer clic en el botón "No", el modal se cierra.
22. Al hacer clic en el botón "No", se muestra una alerta con el mensaje "Restaurado exitosamente".
23. Al hacer clic en el botón "Sí", el modal se cierra.
24. Al hacer clic en el botón "Sí", se muestra una alerta con el mensaje "Restaurado exitosamente".
25. Al hacer clic en el botón "Sí", el patrocinador se restaura con los datos del formulario.

7.5.2. Código fuente de crear patrocinador

Controlador de crear patrocinador

La función store en el controlador PatrocinadorController de Laravel se encarga de crear un nuevo patrocinador con la información proporcionada en la solicitud HTTP. En primer lugar, instancia un objeto de la clase Patrocinador y asigna los valores del nombre y enlace web del patrocinador desde la solicitud. Además, almacena la imagen (logo) asociada al patrocinador en el sistema de archivos mediante la función auxiliar storeImage, obteniendo la URL correspondiente. Luego, guarda el nuevo patrocinador en la base de datos. La función maneja posibles excepciones, como la detección de

patrocinadores duplicados, y responde con mensajes JSON indicando el resultado de la operación. Si el patrocinador se crea con éxito, se devuelve un mensaje de creación exitosa, indicando que no hubo errores ni borrados lógicos. En caso de que el patrocinador ya exista, se captura la excepción correspondiente y se devuelve un mensaje indicando que el patrocinador ya está registrado. Para otras excepciones, se proporciona un mensaje de error general junto con un código de estado 500 para alertar sobre problemas inesperados.

Ruta del archivo: app/Http/Controllers/PatrocinadorController.php

```php
/**
 * Crea un nuevo patrocinador con la información proporcionada en la solicitud HTTP.
 */
public function store(Request $request)
{
    try {
        // Instancia un nuevo objeto Patrocinador
        $patrocinador = new Patrocinador();

        // Asigna el nombre del patrocinador desde la solicitud
        $patrocinador->nombre = $request->nombre;

        // Almacena la imagen (logo) del patrocinador en el sistema de archivos y obtiene la URL
        $patrocinador->ruta_imagen = $this->storeImage($request);

        // Asigna el enlace web del patrocinador desde la solicitud
        $patrocinador->enlace_web = $request->enlace_web;

        // Guarda el nuevo patrocinador en la base de datos
        $patrocinador->save();

        // Devuelve una respuesta JSON indicando el éxito de la creación
        return response()->json(['mensaje' => 'Creado exitosamente', 'error' => false, 'borrado' =>
        false]);
    } catch (QueryException $e) {
        // Maneja excepciones, por ejemplo, si el patrocinador ya existe
        if ($e->errorInfo[1] == 1062) {
            return response()->json(['mensaje' => 'El patrocinador ya existe', 'error' => true, 'borrado'
            => false]);
        } else {
            // Devuelve una respuesta JSON con el mensaje de error y código 500 en caso de otra excepción
            return response()->json(['error' => $e->getMessage()], 500);
        }
    }
}
```

Código JavaScript de crear patrocinador

Se encarga de manejar la interfaz de usuario para la creación y actualización de patrocinadores. En primer lugar, inicializa una tabla interactiva utilizando la biblioteca DataTable para mostrar y organizar la información de los patrocinadores. Se incorpora la funcionalidad de validar y previsualizar imágenes antes de ser cargadas, mejorando la experiencia del usuario al subir logotipos de patrocinadores. El script gestiona las respuestas del servidor después de intentar crear un nuevo patrocinador. En caso de que el patrocinador ya exista pero esté marcado como borrado, se muestra un modal para confirmar su restauración. Además, se incluyen funciones para restaurar patrocinadores existentes, obtener y mostrar datos del patrocinador restaurado, y actualizar la información del patrocinador. Las funciones updateTablaPatrocinadores y resetInputs proporcionan acciones posteriores a operaciones exitosas. La primera redirige al usuario a

la página de administración de patrocinadores, mientras que la segunda restablece los campos del formulario después de completar una acción.

```
// Declaración de variables globales
let tablaDePatrocinadores;
let tablaInicializada = false;

// Obtención de elementos del DOM
const input = document.getElementById("imageUpload");
const imagenPreview = document.getElementById("imagePreview");
const botonSubir = document.getElementById("botonSubirLogoPatrocinador");
const nombrePatrocinador = document.getElementById("nombrePatricinador");
const urlPatrocinador = document.getElementById("urlPatrocinador");
const cancelar = document.getElementById("crearPatrocinadorCancelar");
const asignar = document.getElementById("crearPatrocinadorCrear");

// Opciones de configuración para DataTable
const dataTableOptions = {
    pageLength: 10,
    lengthMenu: [5, 10, 15, 20],
    destroy: true,
    order: [[3, "desc"]],
    language: {
        // Configuración del idioma para DataTable
        lengthMenu: "Mostrar _MENU_ patrocinadores",
        zeroRecords: "Ningún patrocinador encontrado",
        info: "Mostrando de _START_ a _END_ de un total de _TOTAL_ patrocinadores",
        infoEmpty: "Ningún patrocinador encontrado",
        infoFiltered: "(filtrados desde _MAX_ patrocinadores en total)",
        search: "Buscar:",
        loadingRecords: "Cargando...",
        paginate: {
            first: "Primero",
            last: "Último",
            next: "Siguiente",
            previous: "Anterior",
        },
    },
};
```

```
// Inicialización de DataTable al cargar la página
const initDataTable = async () => {
    if (tablaInicializada) {
        tablaDePatrocinadores.destroy();
    }
    DataTable.datetime("DD-MM-YYYY");
    tablaDePatrocinadores = $("#tablaEvento").DataTable(dataTableOptions);
    tablaInicializada = true;
};

// Manejador de eventos al cargar la página
window.addEventListener("load", () => {
    initDataTable();
});

// Función para validar la imagen del patrocinador antes de subirla
const validarImagen = (input, peso, callback) => {
    if (input.files.length > 0) {
        const imagen = input.files[0];
        const maxFileSize = peso * 1024 * 1024;
        let mensaje = { mensaje: "", error: false };

        const type = !/image\/(png|jpeg|jpg)/.test(imagen.type);

        if (type || imagen.size > maxFileSize) {
            input.value = "";
            mensaje = { mensaje: "Archivo no válido", error: true };
        }

        if (typeof callback === "function") {
            callback(mensaje);
        }
    }
};
```

```
// Función para previsualizar la imagen seleccionada por el usuario
function previsualizarImagen(event) {
    validarImagen(input, 2, (mensaje) => {
        if (!mensaje.error) {
            const file = event.target.files[0];
            const reader = new FileReader();
            reader.onload = function (e) {
                imagenPreview.src = e.target.result;
                botonSubir.classList.remove("btn-outline-danger");
                botonSubir.classList.add("btn-light", "text-primary");
            };
            reader.readAsDataURL(file);
        } else {
            mostrarAlerta("Error", mensaje.mensaje, "danger");
        }
    });
}
// Función para validar los datos antes de crear un patrocinador
const validarDatos = () => {
    let form = document.getElementById("formularioAgregarPatrocinador");
    if (form.checkValidity() && input.files[0] != undefined) {
        form.classList.remove("was-validated");
        crearPatrocinador();
        return;
    }
    if (input.files[0] === undefined) {
        botonSubir.classList.remove("btn-light", "text-primary");
        botonSubir.classList.add("btn-outline-danger");
    }
    form.classList.add("was-validated");
};
// Función para obtener los datos del formulario como FormData
const getFormData = () => {
    const formData = new FormData();
    formData.append("nombre", nombrePatrocinador.value);
    formData.append("enlace_web", urlPatrocinador.value);
    formData.append("logo", input.files[0]);
    return formData;
};
```

```
// Variables para manejar la respuesta de la restauración de un patrocinador existente
let idPatrocinador;
let restoreResponseData;

// Función para crear un patrocinador
const crearPatrocinador = async () => {
    let res = await axios.post("/api/patrocinador/esta-borrado", getFormData()).then((response) => {
        return response.data;
    });

    if (res.borrado) {
        idPatrocinador = res.id;
        $('#modalPatrocinadroExistente').modal('show');
    } else {
        mostrarAlerta(
            res.error ? "Peligro" : "Éxito",
            res.mensaje,
            res.error ? "danger" : "success"
        );
        resetInputs();
        if (!res.error) {
            updateTablaPatrocinadores();
        }
    }
};

// Función para restaurar un patrocinador existente
const restaurarPatrocinador = async () => {
    $('#modalPatrocinadroExistente').modal('hide');
    let res = await axios.post("/api/patrocinador/restaurar/" + idPatrocinador).then((response) => {
        return response.data;
    });
    restoreResponseData = res;
    document.getElementById("nombrePatrocinador").innerText = nombrePatrocinador.value;
    await getDatosPatrocinadorRestaurado();
    getDatosNuevosPatrocinador();
    $('#modalActualizarPatrocinador').modal('show');
};

// Función para obtener los datos de un patrocinador restaurado
const getDatosPatrocinadorRestaurado = async () => {
    let patrocinador = await axios.get("/api/patrocinador/show/" + idPatrocinador).then((response) => {
        return response.data;
    });
    let img = document.getElementById("imagenPatrocinadorRestaurada");
    img.src = patrocinador.ruta_imagen;
    img.alt = patrocinador.nombre;
    let datosDiv = document.getElementById("datosPatrocinadorRestaurado");
    let enlace = patrocinador.enlace_web === null ? "" : patrocinador.enlace_web;
    datosDiv.innerHTML = `<p class="text-truncate" title="${enlace}"><a href="${enlace}" target="_blank">
    ${enlace}</a></p>`;
};
// Función para obtener los nuevos datos de un patrocinador creado
const getDatosNuevosPatrocinador = () => {
    let img = document.getElementById("nuevaImagenPatrocinador");
    img.src = imagenPreview.src;
    img.alt = nombrePatrocinador.value;
    let datosDiv = document.getElementById("nuevosDatosPatrocinador");
    let enlace = urlPatrocinador.value === null ? "" : urlPatrocinador.value;
    datosDiv.innerHTML = `<p class="text-truncate" title="${enlace}"><a href="${enlace}" target="_blank">
    ${enlace}</a></p>`;
};
```

```
// Función para actualizar los datos del patrocinador y mostrar una alerta
const actualizarDatosPatrocinador = async (actualizar) => {
    $('#modalActualizarPatrocinador').modal('hide');
    if (actualizar) {
        let res = await axios.post("/api/patrocinador/editar/" + idPatrocinador, getFormData()).then(
          (response) => {
            return response.data;
        });
    }
    mostrarAlerta(
        restoreResponseData.error ? "Peligro" : "Éxito",
        restoreResponseData.mensaje,
        restoreResponseData.error ? "danger" : "success"
    );
    resetInputs();
    updateTablaPatrocinadores();
};

// Función para actualizar la tabla de patrocinadores
const updateTablaPatrocinadores = () => {
    setTimeout(() => {
        window.location.href = "/admin/patrocinador";
    }, 1750);
};

// Función para resetear los inputs del formulario
const resetInputs = () => {
    document.getElementById("formularioAgregarPatrocinador").classList.remove("was-validated");
    botonSubir.classList.remove("btn-outline-danger");
    botonSubir.classList.add("btn-light", "text-primary");
    nombrePatrocinador.value = "";
    urlPatrocinador.value = "";
    input.value = "";
    imagenPreview.src = "/image/uploading.png";
};
```

Vista de crear patrocinador

La vista se divide en dos secciones principales: una tabla que muestra los patrocinadores existentes y un formulario para agregar nuevos patrocinadores. La tabla utiliza DataTables para proporcionar una interfaz de usuario interactiva y organizada. En la sección de patrocinadores existentes, se utiliza una tabla para mostrar información relevante, como el nombre, enlace web, imagen y fecha de creación de cada patrocinador. El formulario de creación de patrocinadores incluye campos para el nombre, enlace web y una función para subir una imagen del patrocinador. El código también incorpora modales para manejar situaciones específicas, como la existencia previa de un patrocinador. Cuando se intenta crear un patrocinador que ya existe, se despliega un modal que ofrece la opción de restaurar el patrocinador existente. Otro modal se utiliza para confirmar la actualización de datos al intentar modificar un patrocinador existente.

Ruta del archivo: resouces/views/patrocinador/crearPatrocinador.blade.php

```
@extends('layouts.app')

@section('content')
    <!-- Contenedor principal -->
    <div class="container">
        <!-- Encabezado de la página -->
        <div class="row mb-2">
            <h2>Crear patrocinador</h2>
        </div>

        <!-- Sección de patrocinadores existentes -->
        <div class="row g-5">
            <div class="col-sm-12 col-md-8">
                <!-- Tabla para mostrar patrocinadores -->
                <table class="table table-responsive table-striped text-secondary" id="tablaEvento">
                    <caption>Patrocinadores</caption>
                    <thead>
                        <tr>
                            <th scope="col" class="col-sm-3 col-md-3">Nombre del patrocinador</th>
                            <th scope="col" class="col-sm-4 col-md-4">Enlace web</th>
                            <th scope="col" class="col-sm-3 col-md-3">Imagen</th>
                            <th scope="col" class="col-sm-2 col-md-2">Fecha de creación</th>
                        </tr>
                    </thead>
                    <tbody id="datosTabla">
                        <!-- Iteración sobre la lista de patrocinadores para llenar la tabla -->
                        @foreach ($patrocinadores as $patrocinador)
                            <tr id="{{ $patrocinador->id }}">
                                <td>{{ $patrocinador->nombre }}</td>
                                <td>
                                    <!-- Enlace web truncado con enlace externo -->
                                    <a class="d-inline-block text-truncate" href="{{ $patrocinador->enlace_web }}"
                                        target="_blank" style="max-width: 250px;" title="{{ $patrocinador->enlace_web }}">
                                        {{ $patrocinador->enlace_web }}
                                    </a>
                                </td>
                                <td>
                                    <!-- Enlace a la imagen del patrocinador -->
                                    <a href="{{ $patrocinador->ruta_imagen }}" target="_blank">
                                        {{ $patrocinador->nombre }}
                                    </a>
                                </td>
                                <td>{{ date('d-m-Y', strtotime($patrocinador->created_at)) }}</td>
                            </tr>
                        @endforeach
                    </tbody>
                </table>
            </div>

            <!-- Sección de creación de nuevos patrocinadores -->
            <div class="col-sm-12 col-md-4">
                <!-- Contenedor del formulario de creación de patrocinadores -->
                <div class="container d-flex flex-column align-items-center border p-3 container-image">
                    <!-- Título para la imagen del patrocinador -->
                    <h5 class="text-start">Imagen patrocinador</h5>

                    <!-- Previsualización de la imagen y botón para subir una nueva imagen -->
                    <div class="d-flex justify-content-center">
                        <img src="{{ asset('/image/uploading.png') }}" alt="image" id="imagePreview"
                            class="rounded mx-auto d-block img-thumbnail" style="max-height: 270px; max-width: 270px">
```

```
    <input type="file" id="imageUpload" class="d-none" accept="image/jpeg, image/png, image/jpg"
        onchange="previsualizarImagen(event)">
</div>

<!-- Botón para subir la imagen del patrocinador -->
<div class="d-flex justify-content-center mt-3">
    <button type="button" class="btn btn-light text-primary" id="botonSubirLogoPatrocinador"
        onclick="document.getElementById('imageUpload').click()">Subir imagen</button>
</div>

<!-- Formulario de creación de patrocinadores -->
<form class="needs-validation" novalidate id="formularioAgregarPatrocinador">
    <!-- Campo para ingresar el nombre del patrocinador -->
    <div class="col-md-12 mt-2">
        <label for="nombrePatricinador" class="form-label">Nombre del patrocinador *</label>
        <input name="nombre" type="text" class="form-control custom-input" id="nombrePatricinador"
            value="" placeholder="Ingrese un nombre" required>
        <div class="invalid-feedback">
            El nombre no puede estar vacío.
        </div>
    </div>

    <!-- Campo para ingresar el enlace web del patrocinador -->
    <div class="col-md-12 mt-2">
        <label for="urlPatrocinador" class="form-label">Enlace a la página web del patrocinador</label>
        <input name="enlace_web" type="url" class="form-control custom-input" id="urlPatrocinador"
            value="" pattern="https?://.+" placeholder="https://www.ejemplo.com">
    </div>
</form>
```

```
<!-- Botones para cancelar o crear el patrocinador -->
<div class="d-flex justify-content-center mt-3">
    <button type="button" class="btn btn-light" onclick="resetInputs()"
        id="crearPatrocinadorCancelar">Cancelar</button>
    <button type="button" class="btn btn-primary" onclick="validarDatos()"
        id="crearPatrocinadorCrear">Crear</button>

    <!-- Modal para patrocinador existente -->
    @component('components.modal')
        @slot('modalId', 'modalPatrocinadroExistente')
        @slot('modalTitle', 'Patrocinador ya existente')
        @slot('modalContent')
            <p>
                <b>El patrocinador que intentó crear ya existe,</b>
                ¿Desea habilitarlo nuevamente?
            </p>
        @endslot
        @slot('modalButton')
            <button type="button" class="btn btn-secondary" data-bs-dismiss="modal"
                onclick="resetInputs()">No</button>
            <button type="button" class="btn btn-danger" onclick="restaurarPatrocinador()">Sí</button>
        @endslot
    @endcomponent

    <!-- Modal para actualizar patrocinador -->
    <div class="modal fade" id="modalActualizarPatrocinador" tabindex="-1"
        aria-labelledby="modalAnularLabel" aria-hidden="true">
        <!-- Contenido del modal -->
        <div class="modal-dialog">
            <div class="modal-content" style="width: 600px">
```

```
                <div class="modal-header">
                    <!-- Título del modal -->
                    <h5 class="modal-title">
                        ¿Desea actualizar los datos del patrocinador?
                    </h5>
                </div>

                <!-- Cuerpo del modal -->
                <div class="modal-body">
                    <div class="row">
                        <div class="col-md-12 text-center">
                            <!-- Nombre del patrocinador -->
                            <h5 id="nombrePatrocinador"></h5>
                        </div>
                    </div>

                    <!-- Sección de comparación de datos anterior y nuevo -->
                    <div class="row g-4">
                        <!-- Datos anteriores -->
                        <div class="col-md-6 border-end border-2">
                            <div class="col-md-12 text-center">
                                Anterior
                            </div>
                            <div class="row justify-content-center mt-2" style="height: 150px">
                                <!-- Imagen del patrocinador antes de la actualización -->
                                <img src="" class="img-fluid rounded object-fit-scale"
                                    alt="logoPatrocinador" style="max-height: 100%; width: auto"
                                    id="imagenPatrocinadorRestaurada">
                            </div>
                            <!-- Enlace web del patrocinador antes de la actualización -->
                            <h6 class="mt-2" id="datosPatrocinadorRestaurado"></h6>
                        </div>

                        <!-- Datos nuevos -->
                        <div class="col-md-6">
                            <div class="col-md-12 text-center">
                                Nuevo
                            </div>
                            <div class="row justify-content-center mt-2" style="height: 150px">
                                <!-- Imagen del patrocinador después de la actualización -->
                                <img src="" class="img-fluid rounded object-fit-scale"
                                    alt="logoPatrocinador" style="max-height: 100%; width: auto"
                                    id="nuevaImagenPatrocinador">
                            </div>
                            <!-- Enlace web del patrocinador después de la actualización -->
                            <h6 class="mt-2" id="nuevosDatosPatrocinador"></h6>
                        </div>
                    </div>
                </div>

                <!-- Pie del modal con botones de confirmación -->
                <div class="modal-footer d-flex justify-content-center">
                    <button type="button" class="btn btn-secondary" data-bs-dismiss="modal"
                        onclick="actualizarDatosPatrocinador(false)">
                        No</button>
                    <button type="button" class="btn btn-danger"
                        onclick="actualizarDatosPatrocinador(true)">Sí</button>
                </div>
            </div>
```

```
                                    </div>
                                </div>
                            </div>
                        </div>
                    </div>
                </div>
            </div>

            <!-- Enlaces a bibliotecas externas y scripts locales -->
            <link href="https://cdn.datatables.net/1.13.6/css/dataTables.bootstrap5.min.css" rel="stylesheet">
            <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.7.1/jquery.min.js"></script>
            <script src="https://cdn.datatables.net/1.13.6/js/jquery.dataTables.min.js"></script>
            <script src="https://cdn.datatables.net/1.13.6/js/dataTables.bootstrap5.min.js"></script>
            <script src="https://cdnjs.cloudflare.com/ajax/libs/moment.js/2.29.2/moment.min.js"></script>
            <script src="{{ asset('js/Patrocinador/crearPatrocinador.js') }}" defer></script>
@endsection
```

Follow link (ctrl + click)

7.5.3. Interfaz de usuario de crear patrocinador

La interfaz de usuario para la creación y gestión de patrocinadores proporciona una experiencia clara y eficiente. Aquí se detallan los elementos clave de la interfaz:

Tabla de Patrocinadores:

Nombre del Patrocinador: Identifica de manera clara el nombre del patrocinador.

Enlace Web: Muestra el enlace web asociado al patrocinador, facilitando la navegación.

Imagen: Presenta la imagen del patrocinador con la opción de previsualizar antes de la creación.

Fecha de Creación: Indica la fecha en que se creó el patrocinador para una referencia rápida.

Formulario de Creación de Patrocinador:

Nombre del Patrocinador: Campo esencial para ingresar el nombre del nuevo patrocinador.

Enlace Web: Campo opcional que permite agregar el enlace web del patrocinador.

Imagen: Permite subir una imagen representativa del patrocinador.

Validación de Datos:

Se realiza una validación en tiempo real de los campos del formulario, destacando los errores y proporcionando mensajes claros para garantizar la integridad de los datos ingresados.

Previsualización de Imagen:

Se incorpora una sección que permite previsualizar la imagen seleccionada antes de confirmar la creación del patrocinador.

Modales:

Patrocinador Existente: En caso de intentar crear un patrocinador que ya existe, se despliega un modal que ofrece la opción de restaurar el patrocinador existente, evitando duplicados innecesarios.

Actualización de Datos: Al modificar un patrocinador existente, se utiliza un modal para confirmar la actualización de los datos, mostrando una comparación visual entre la información anterior y la nueva.

Navegación Eficiente:

La tabla de patrocinadores permite una identificación rápida y eficiente de la información existente, facilitando la toma de decisiones al agregar nuevos patrocinadores.

Diseño y Usabilidad:

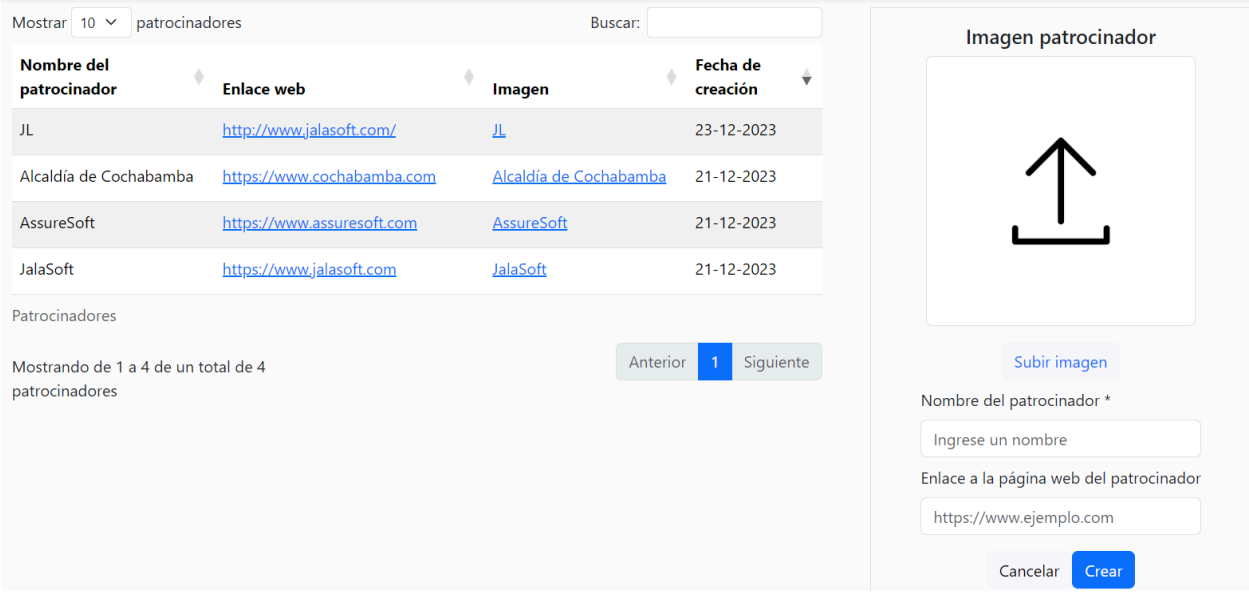
El diseño claro y organizado del formulario simplifica la entrada de datos, proporcionando una experiencia intuitiva para los usuarios.

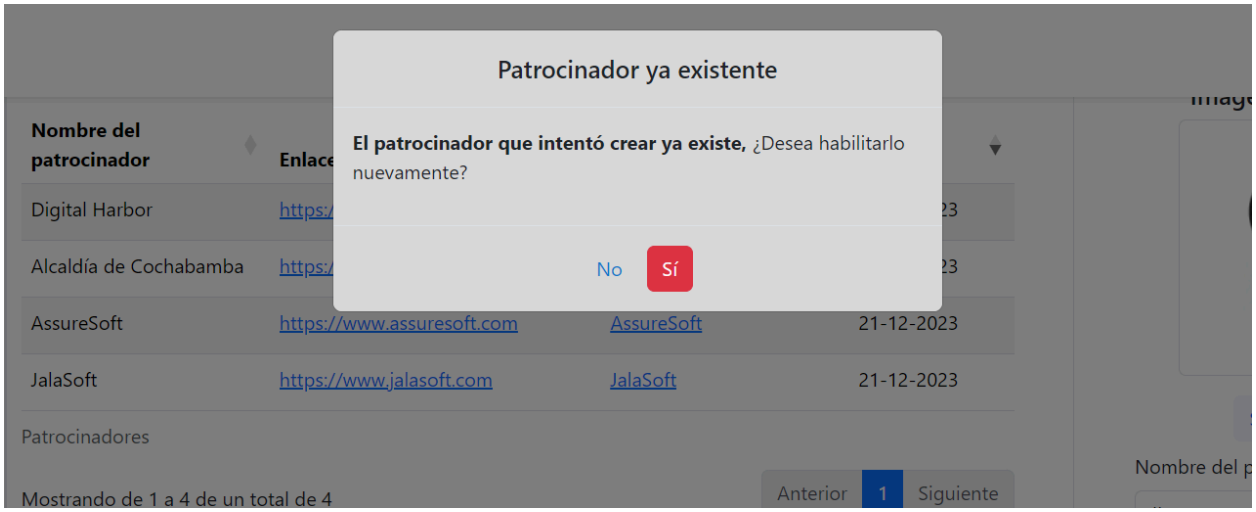
Los elementos visuales, como las imágenes previsualizadas, mejoran la comprensión y facilitan la interacción del usuario.

Mensajes de Alerta:

Se implementan mensajes de alerta para informar sobre acciones exitosas o posibles errores durante el proceso de creación y gestión de patrocinadores.

Esta interfaz busca optimizar la creación de patrocinadores, ofreciendo herramientas visuales y funcionales que aseguran una experiencia efectiva y libre de complicaciones para el usuario.





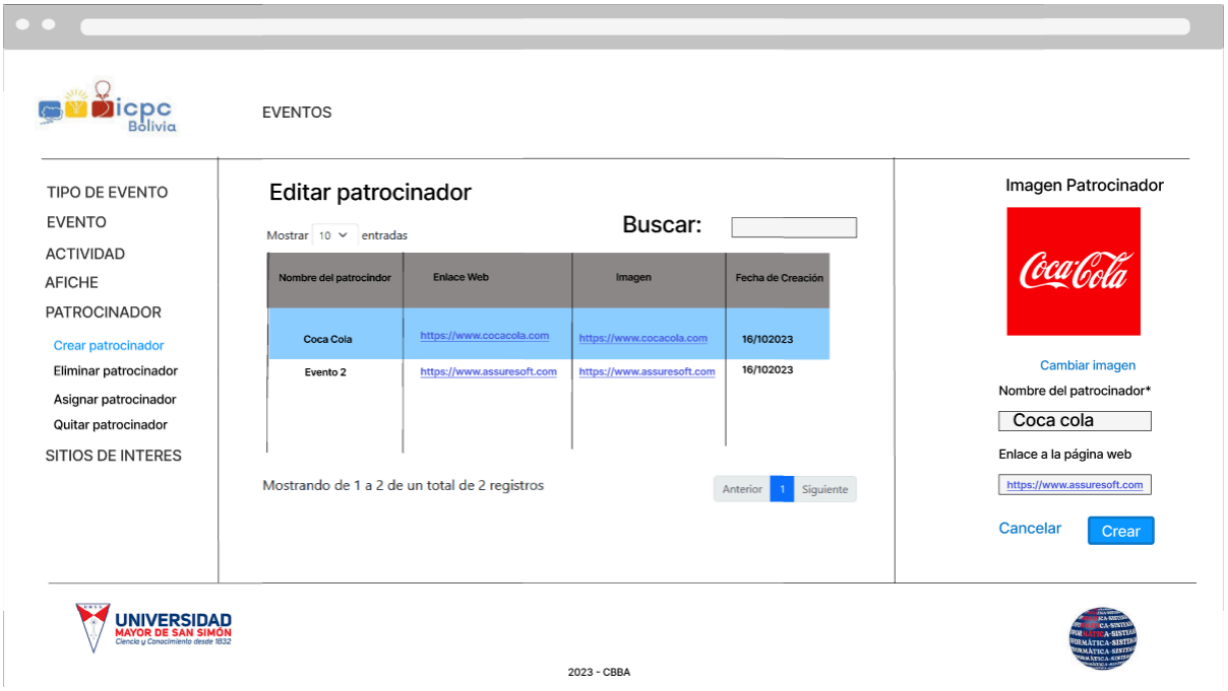
7.5.2. Editar patrocinador

7.5.2.1. Historia de usuario de editar patrocinador

Historia de Usuario	
Título: Editar patrocinador	**ID: 48**
Estimación: 8	**Importancia:** Media
Descripción:	

Como usuario con privilegios de editar patrocinador, quiero editar un patrocinador para mantener la información de los patrocinadores existentes actualizada y correcta.

MockUps



Criterios de aceptación

1. Al presionar en la sección “PATROCINADOR” en el menú lateral y seleccionar la opción “Editar patrocinador”, se muestra una tabla con los patrocinadores existentes y un formulario para editar un patrocinador.
2. La tabla tiene las columnas “Nombre del patrocinador”, “Enlace web”, “Imagen”, “Última actualización”.
3. En la columna 'Imagen' de la tabla, se proporciona un enlace que redirige a una ventana nueva, donde se puede visualizar la imagen del patrocinador elegido.
4. En la columna “Última actualización” de la tabla, se muestra la fecha en la que se hizo la última actualización de la información de cada patrocinador.
5. Una vez elegido el patrocinador se recupera la información de ese patrocinador como la imagen, nombre del patrocinador y el enlace web en el formulario.
6. No se puede modificar el campo “Nombre del patrocinador *”.
7. Se puede modificar el campo “Enlace a la página web del patrocinador”.
8. El campo “Enlace a la página web del patrocinador”, requiere que se tenga el prefijo “http:” o “https:”.
9. Al presionar el botón “Cambiar imagen” se muestra el gestor de archivos del sistema operativo para seleccionar una nueva imagen.
10. La imagen seleccionada debe tener como peso máximo 2 MB.
11. La imagen seleccionada tiene los formatos permitidos jpg, jpeg, png.

12. Cuando la imagen no cumple con los formatos mencionados se muestra una alerta con el mensaje “Archivo no válido”.
13. Si no ocurre error en la validación, la imagen se previsualiza.
14. Al presionar el botón “Editar” se abre un modal de confirmación con el mensaje “¿Está seguro de editar este patrocinador?”.
15. Al presionar el botón “Sí” en el modal, aparece una alerta temporal con el mensaje “Actualizado exitosamente”.
16. Al presionar el botón “Sí” en el modal, se cierra el modal.
17. Al presionar el botón “No” en el modal, se cierra el modal.
18. Al presionar el botón “Cancelar” el formulario se recarga y regresa a su estado inicial.

7.5.2.2. Código fuente de editar patrocinador

Controlador de editar patrocinador

La función update en el controlador PatrocinadorController de Laravel se encarga de modificar la información de un patrocinador existente en la base de datos. Al recibir una solicitud HTTP que contiene los datos actualizados, identifica el patrocinador mediante su identificador único ($id). Posteriormente, actualiza los campos del patrocinador con la nueva información proporcionada en la solicitud, permitiendo la edición del nombre, logo y enlace web del patrocinador. En caso de que se adjunte un nuevo archivo de imagen (logo), la función elimina la imagen anterior asociada al patrocinador y almacena la nueva imagen en el sistema de archivos. Finalmente, guarda los cambios realizados en la base de datos.

Ruta del archivo: app/Http/Controllers/PatrocinadorController.php

```php
/**
 * Actualiza la información de un patrocinador existente.
 */
public function update(Request $request, $id)
{
    try {
        // Obtiene el patrocinador por su identificador.
        $patrocinador = Patrocinador::find($id);

        // Actualiza los campos del patrocinador con los datos proporcionados en la solicitud.
        $patrocinador->nombre = $request->nombre;
        if ($request->hasFile('logo')) {
            // Elimina la imagen anterior y almacena la nueva.
            Storage::delete($patrocinador->ruta_imagen);
            $patrocinador->ruta_imagen = $this->storeImage($request);
        }
        $patrocinador->enlace_web = $request->enlace_web;

        // Guarda los cambios en la base de datos.
        $patrocinador->save();

        // Retorna una respuesta JSON indicando el éxito de la operación.
        return response()->json(['mensaje' => 'Actualizado exitosamente', 'error' => false]);
    } catch (QueryException $e) {
        // En caso de error, retorna un mensaje de error.
        return response()->json(['error' => $e->getMessage()], 500);
    }
}
```

Codigo JavaScript de editar patrocinador

Se encarga de inicializar y gestionar la tabla de patrocinadores utilizando DataTables con opciones específicas de paginación, orden y búsqueda. Además, se implementan mecanismos para la validación de imágenes antes de ser cargadas, y se proporciona una funcionalidad de previsualización de imágenes seleccionadas. El código también controla la habilitación y deshabilitación de elementos del formulario según la interacción del usuario y maneja eventos relacionados con la edición de patrocinadores. La lógica de selección de patrocinadores, la creación de objetos FormData para la actualización de patrocinadores mediante solicitudes HTTP, y la gestión de alertas para informar al usuario sobre el resultado de las operaciones, también están presentes en el código.

Ruta del archivo: public/js/Patrocinador/editarPatrocinador.js

```
// Declaración de variables globales
let tablaDePatrocinadores;
let tablaInicializada = false;
const input = document.getElementById("imageUpload");
const imagenPreview = document.getElementById("imagePreview");
const botonSubir = document.getElementById("botonSubirLogoPatrocinador");
const nombrePatrocinador = document.getElementById("nombrePatricinador");
const urlPatricinador = document.getElementById("urlPatrocinador");
const cancelar = document.getElementById("cancelarEditarPatrocinador");
const asignar = document.getElementById("editarPatrocinador");
const patrociandorSeleccionado = document.getElementById("nombrePatrocinador");

// Opciones para DataTables
const dataTableOptions = {
    pageLength: 10,
    lengthMenu: [5, 10, 15, 20],
    destroy: true,
    order: [[3, "desc"]],
    language: {
        lengthMenu: "Mostrar _MENU_ patrocinadores",
        zeroRecords: "Ningún patrociandor encontrado",
        info: "Mostrando de _START_ a _END_ de un total de _TOTAL_ patrocinadores",
        infoEmpty: "Ningún patrocinador encontrado",
        infoFiltered: "(filtrados desde _MAX_ patrocinadores en total)",
        search: "Buscar:",
        loadingRecords: "Cargando...",
        paginate: {
            first: "Primero",
            last: "Último",
            next: "Siguiente",
            previous: "Anterior",
        },
    },
};
```

```
// Inicialización de DataTable
const initDataTable = async () => {
    if (tablaInicializada) {
        tablaDePatrocinadores.destroy();
    }
    DataTable.datetime("DD-MM-YYYY");
    tablaDePatrocinadores = $("#tablaPatrocinadores").DataTable(
        dataTableOptions
    );
    tablaInicializada = true;
};

// Evento al cargar la página
window.addEventListener("load", () => {
    initDataTable();
    if (!seleccionado) {
        patrociandorSeleccionado.textContent = "Seleccione un patrocinador";
        inhabilitarFormulario();
    }
});

// Función para validar características de la imagen
const validarImagen = (input, peso, callback) => {
    if (input.files.length > 0) {
        const imagen = input.files[0];
        const maxFileSize = peso * 1024 * 1024;
        let mensaje = { mensaje: "", error: false };

        const type = !/image\/(png|jpeg|jpg)/.test(imagen.type);

        if (type || imagen.size > maxFileSize) {
            input.value = "";
            mensaje = { mensaje: "Archivo no válido", error: true };
        }

        if (typeof callback === "function") {
            callback(mensaje);
        }
    }
};
```

```
// Función para previsualizar la imagen seleccionada
function previsualizarImagen(event) {
    validarImagen(input, 2, (mensaje) => {
        if (!mensaje.error) {
            const file = event.target.files[0];
            const reader = new FileReader();
            reader.onload = function (e) {
                imagenPreview.src = e.target.result;
                botonSubir.classList.remove("btn-outline-danger");
                botonSubir.classList.add("btn-light", "text-primary");
            };
            reader.readAsDataURL(file);
        } else {
            mostrarAlerta("Error", mensaje.mensaje, "danger");
        }
    });
}

// Función para validar datos del formulario
const validarDatos = () => {
    let form = document.getElementById("formularioEditarPatrocinador");
    if (form.checkValidity()) {
        form.classList.remove("was-validated");
        crearFormData();
        return;
    }
    form.classList.add("was-validated");
};

// Función para crear un objeto FormData con la información del formulario
const crearFormData = () => {
    const formData = new FormData();
    formData.append("nombre", nombrePatrocinador.value);
    formData.append("enlace_web", urlPatricinador.value);
    formData.append("logo", input.files[0]);
    editarPatrocinador(formData);
};
```

```
// Función para actualizar la tabla de patrocinadores
const updateTablaPatrocinadores = () => {
    setTimeout(() => {
        window.location.href = "editar";
    }, 1700);
};
// Función para resetear los campos del formulario
const resetInputs = (control) => {
    document
        .getElementById("formularioEditarPatrocinador")
        .classList.remove("was-validated");
    nombrePatrocinador.value = "";
    urlPatricinador.value = "";
    input.value = "";
    imagenPreview.src = "/image/uploading.png";
    if (control == 0) {
        window.location.reload();
    }
};
// Función para inhabilitar el formulario
const inhabilitarFormulario = () => {
    botonSubir.disabled = true;
    nombrePatrocinador.disabled = true;
    urlPatricinador.disabled = true;
    input.disabled = true;
    asignar.disabled = true;
    cancelar.disabled = true;
    document.getElementById("formularioEditarPatrocinador").disabled = true;
};
// Función para habilitar el formulario
const habilitarFormulario = () => {
    resetInputs();
    botonSubir.disabled = false;
    urlPatricinador.disabled = false;
    urlPatricinador
    input.disabled = false;
    asignar.disabled = false;
    cancelar.disabled = false;
    document.getElementById("formularioEditarPatrocinador").disabled = false;
};
```

```
// Variables para gestionar el estado de la selección
let seleccionado;
let idSeleccionado;
let asignando = false;
let patrocinadoresNoAsigandos;

// Función para seleccionar un patrocinador
const seleccionarPatrocinador = async (id, nombre) => {
    if (seleccionado) {
        seleccionado.classList.remove("table-primary");
    }
    habilitarFormulario();
    seleccionado = document.getElementById(id);
    seleccionado.classList.add("table-primary");
    idSeleccionado = id;
    patrociandorSeleccionado.textContent = nombre;
    await llenarFormulario(id);
};

// Función para editar un patrocinador
const editarPatrocinador = async (formData) => {
    await axios.post(`/api/patrocinador/editar/${idSeleccionado}`, formData).then((response) => {
        mostrarAlerta(
            "Éxito",
            response.data.mensaje,
            response.error ? "danger" : "success"
        );
        resetInputs(1);
        updateTablaPatrocinadores();
    });
};

// Función para obtener la información de un patrocinador
const getPatrocinador = async (id) => {
    const response = await axios.get(`/api/patrocinador/show/${id}`);
    return response.data;
};

// Función para llenar el formulario con la información del patrocinador
const llenarFormulario = async (id) => {
    const patrocinador = await getPatrocinador(id);
    nombrePatrocinador.value = patrocinador.nombre;
    urlPatricinador.value = patrocinador.enlace_web;
    imagenPreview.src = patrocinador.ruta_imagen;
};
```

Vista de editar patrocinador

La interfaz se compone de dos secciones principales: la tabla de patrocinadores en la parte izquierda y los detalles del patrocinador seleccionado junto con un formulario de edición en la parte derecha. La tabla muestra información clave de los patrocinadores, como el nombre, enlace web, imagen y última actualización. Al hacer clic en un patrocinador específico de la tabla, se activa la función seleccionarPatrocinador, que resalta la fila seleccionada y carga los detalles del patrocinador en el formulario de edición de la sección derecha. El formulario de edición permite cambiar el nombre del patrocinador, el enlace web y la imagen asociada. Se proporciona una previsualización de la imagen y un botón para cargar una nueva. Además, se incorpora un sistema de validación para garantizar la integridad de los datos ingresados. El código también

incluye un modal de confirmación (modalEdicionPatrocinador) que se activa al intentar realizar la edición.

Ruta del archivo: resources/views/patrocinador/editarPatrocinador.blade.php

```
@extends('layouts.app')

@section('content')
    <div class="container">
        <div class="row mb-2">
            <h2>Editar patrocinador</h2>
        </div>
        <div class="row g-5">
            <!-- Sección izquierda: Tabla de patrocinadores -->
            <div class="col-sm-12 col-md-8">
                <table class="table table-responsive table-striped text-secondary" id="tablaPatrocinadores">
                    <!-- Encabezados de la tabla -->
                    <thead>
                        <tr>
                            <th scope="col" class="col-sm-3 col-md-3">Nombre del patrocinador</th>
                            <th scope="col" class="col-sm-4 col-md-4">Enlace web</th>
                            <th scope="col" class="col-sm-3 col-md-3">Imagen</th>
                            <th scope="col" class="col-sm-2 col-md-2">Última actualización</th>
                        </tr>
                    </thead>
                    <!-- Cuerpo de la tabla, generada dinámicamente con datos de patrocinadores -->
                    <tbody id="datosTabla">
                        @foreach ($patrocinadores as $patrocinador)
                            <tr id="{{ $patrocinador->id }}"
                                onclick="seleccionarPatrocinador({{ $patrocinador->id }}, '{{ $patrocinador->nombre }}')">
                                <td>{{ $patrocinador->nombre }}</td>
                                <td>
                                    <a class="d-inline-block text-truncate" href="{{ $patrocinador->enlace_web }}"
                                        target="_blank" style="max-width: 250px;" title="{{ $patrocinador->enlace_web }}">
                                        {{ $patrocinador->enlace_web }}
                                    </a>
                                </td>
                                <td><a href="{{ $patrocinador->ruta_imagen }}"
                                        target="_blank">{{ $patrocinador->nombre }}</a></td>
                                <td>{{ date('d-m-Y', strtotime($patrocinador->updated_at)) }}</td>
                            </tr>
                        @endforeach
                    </tbody>
                </table>
            </div>

            <!-- Sección derecha: Detalles y formulario de edición del patrocinador seleccionado -->
            <div class="col-sm-12 col-md-4">
                <div class="container d-flex flex-column align-items-center border p-3 container-image">
                    <!-- Nombre del patrocinador seleccionado -->
                    <h5 id="nombrePatrocinador" class="text-center fw-bold"></h5>
                    <h5 class="text-start">Imagen patrocinador</h5>
                    <div class="d-flex justify-content-center">
                        <!-- Previsualización de la imagen del patrocinador y carga de nueva imagen -->
                        <img src="{{ asset('/image/uploading.png') }}" alt="image" id="imagePreview"
                            class="rounded mx-auto d-block img-thumbnail" style="max-height: 270px; max-width: 270px">
                        <input type="file" id="imageUpload" class="d-none" accept="image/jpeg, image/png, image/jpg"
                            onchange="previsualizarImagen(event)">
                    </div>

                    <div class="d-flex justify-content-center mt-3">
                        <!-- Botón para cambiar la imagen del patrocinador -->
                        <button type="button" class="btn btn-light text-primary" id="botonSubirLogoPatrocinador"
                            onclick="document.getElementById('imageUpload').click()">Cambiar imagen</button>
                    </div>
```

```
                <!-- Formulario para editar información del patrocinador -->
                <form class="needs-validation" novalidate id="formularioEditarPatrocinador">
                    <!-- Campo para el nombre del patrocinador -->
                    <div class="col-md-12 mt-2">
                        <label for="nombrePatricinador" class="form-label">Nombre del patrocinador *</label>
                        <input name="nombre" type="text" class="form-control custom-input" id="nombrePatricinador"
                            value="" placeholder="Ingrese un nombre" required
                            title="El nombre del patrocinador no se puede editar">
                        <div class="invalid-feedback">
                            El nombre no puede estar vacío.
                        </div>
                    </div>

                    <!-- Campo para el enlace web del patrocinador -->
                    <div class="col-md-12 mt-2">
                        <label for="urlPatrocinador" class="form-label">Enlace a la página web del patrocinador</label>
                        <input name="enlace_web" type="url" class="form-control custom-input" id="urlPatrocinador"
                            value="" pattern="https?://.+" placeholder="https://www.ejemplo.com">
                    </div>
                </form>
                <div class="d-flex justify-content-center mt-3">
                    <!-- Botones para cancelar la edición o abrir el modal de confirmación -->
                    <button type="button" class="btn btn-light" onclick="resetInputs(0)"
                        id="cancelarEditarPatrocinador">Cancelar</button>
                    <button type="button" class="btn btn-primary" data-bs-toggle="modal"
                        data-bs-target="#modalEdicionPatrocinador" id="editarPatrocinador">Editar</button>
                </div>
                <!-- Modal de confirmación para la edición del patrocinador -->
                @component('components.modal')
                    @slot('modalId', 'modalEdicionPatrocinador')
                    @slot('modalTitle', 'Confirmación')
                    @slot('modalContent')
                        ¿Está seguro de editar este patrocinador?
                    @endslot
                    @slot('modalButton')
                        <button type="button" class="btn btn-secondary w-25 mx-8" data-bs-dismiss="modal">No</button>
                        <button type="reset" class="btn btn-primary w-25 mx-8" data-bs-dismiss="modal"
                            onclick="validarDatos()">Sí</button>
                    @endslot
                @endcomponent
            </div>
        </div>
    </div>
</div>

<!-- Enlaces a bibliotecas y scripts necesarios -->
<link href="https://cdn.datatables.net/1.13.6/css/dataTables.bootstrap5.min.css" rel="stylesheet">
<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.7.1/jquery.min.js"></script>
<script src="https://cdn.datatables.net/1.13.6/js/jquery.dataTables.min.js"></script>
<script src="https://cdn.datatables.net/1.13.6/js/dataTables.bootstrap5.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/moment.js/2.29.2/moment.min.js"></script>
<script src="{{ asset('js/Patrocinador/editarPatrocinador.js') }}" defer></script>
@endsection
```

7.5.2.3. Interfaz de usuario de editar patrocinador

La interfaz de usuario destinada a la edición de patrocinadores proporciona una experiencia intuitiva y eficiente. Aquí se describen los elementos clave de la interfaz:

Tabla de Patrocinadores:

Nombre del Patrocinador: Identifica claramente el nombre del patrocinador.

Enlace Web: Muestra el enlace web asociado al patrocinador, facilitando la navegación.

Imagen: Presenta la imagen del patrocinador con la opción de previsualizar antes de la edición.

Fecha de Última Actualización: Indica la fecha de la última actualización del patrocinador para una referencia rápida.

Formulario de Edición de Patrocinador:

Nombre del Patrocinador: Campo esencial para editar el nombre del patrocinador existente.

Enlace Web: Campo opcional que permite modificar el enlace web del patrocinador.

Imagen: Permite subir una nueva imagen representativa del patrocinador.

Validación de Datos:

Se realiza una validación en tiempo real de los campos del formulario, resaltando errores y proporcionando mensajes claros para garantizar la integridad de los datos ingresados.

Previsualización de Imagen:

Se incorpora una sección que permite previsualizar la nueva imagen seleccionada antes de confirmar la edición del patrocinador.

Modales:

Confirmación de Edición: Al modificar un patrocinador existente, se utiliza un modal para confirmar la actualización de los datos, mostrando una comparación visual entre la información anterior y la nueva.

Navegación Eficiente:

La tabla de patrocinadores permite una identificación rápida y eficiente de la información existente, facilitando la toma de decisiones al editar patrocinadores.

Diseño y Usabilidad:

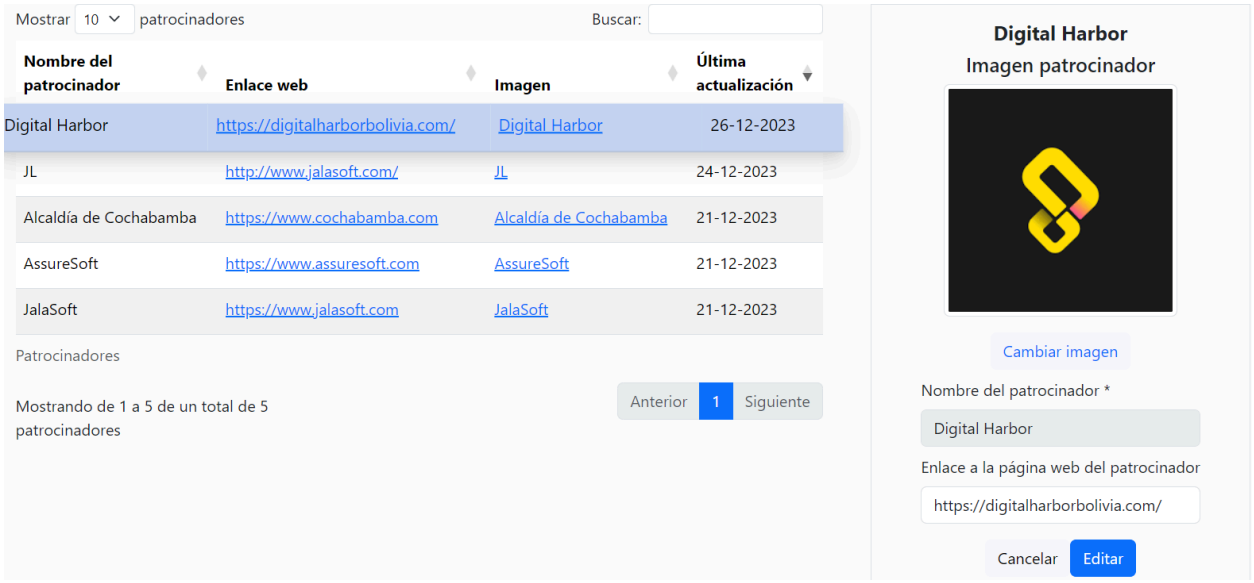
El diseño claro y organizado del formulario simplifica la edición de datos, proporcionando una experiencia intuitiva para los usuarios.

Los elementos visuales, como las imágenes previsualizadas, mejoran la comprensión y facilitan la interacción del usuario.

Mensajes de Alerta:

Se implementan mensajes de alerta para informar sobre acciones exitosas o posibles errores durante el proceso de edición de patrocinadores.

Esta interfaz busca optimizar la edición de patrocinadores, ofreciendo herramientas visuales y funcionales que aseguran una experiencia efectiva y libre de complicaciones para el usuario.

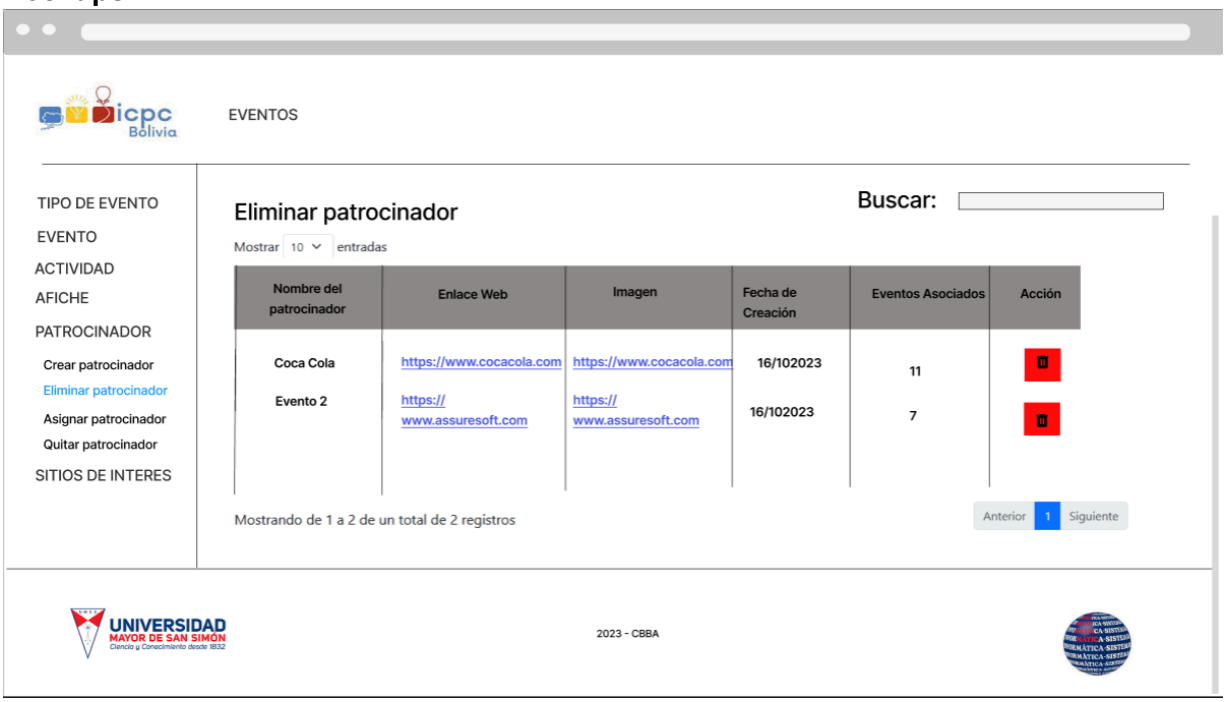


7.5.3. Eliminar patrocinador

7.5.3.1. Historia de usuario de eliminar patrocinador

Historia de Usuario	
Título: Eliminar patrocinador	**ID: 12**
Estimación: 3	**Importancia: Media**
Descripción Como usuario con privilegios para eliminar patrocinador, deseo eliminar un patrocinador para que la información sobre quienes patrocinan los eventos esté siempre correcta y actualizada.	

Mockups



Criterios de Aceptación

1. Al hacer clic en la sección “PATROCINADOR” en el menú lateral y seleccionar la opción “Eliminar patrocinador” se carga una tabla que muestra la información de los patrocinadores creados previamente.
2. En la columna “Nombre del patrocinador” en la tabla, se muestran los nombres de los patrocinadores existentes.
3. En la columna “Enlace web” de la tabla, se muestran los enlaces web de cada uno de los patrocinadores existentes.
4. Los elementos de la columna “Enlace web” son enlaces que redirigen a una página web asociada al patrocinador.
5. Los elementos de la columna “Enlace web” están vacíos, si el patrocinador no tiene un enlace registrado.
6. En la columna “Imagen” de la tabla, se muestra un enlace que redirecciona a una nueva ventana en la cual se visualiza de manera gráfica la imagen del patrocinador.
7. En la columna de “Fecha de creación” de la tabla, la fecha se muestra en el formato DD-MM-AAAA.
8. En la columna “Eventos asociados” se muestra la cantidad de eventos asociados a cada patrocinador.
9. En la columna “Acción” de la tabla, se muestra el botón “ ” para cada patrocinador.
10. Si el patrocinador tiene cero eventos asociados. Al hacer clic sobre el icono “ ”, se muestra un modal con el mensaje “¿Está seguro que quiere eliminar este patrocinador?”, con los botones de “No" y "Sí".
11. Al presionar “No” el modal se cierra y no se realiza ningún cambio.
12. Al presionar “Sí” se muestra una alerta temporal con el mensaje “Eliminado

exitosamente".

13. Si el patrocinador tiene eventos asociados, Al hacer clic sobre el icono "  ", se muestra un modal de confirmación con el mensaje "Este patrocinador tiene eventos asociados, ¿Está seguro que quiere eliminar este patrocinador?", con los botones "No" y "Sí".
14. Al presionar "No" se cierra el modal y no ocurre ningún cambio.
15. Al presionar "Sí" se muestra una alerta temporal con el mensaje "Eliminado exitosamente".
16. Al presionar "Sí" se cierra el modal.
17. Al presionar "Sí" el patrocinador ya no es visible en la tabla.

7.5.3.2. Código fuente de eliminar patrocinador

Controlador de eliminar patrocinador

La función destroy se encarga de eliminar un patrocinador específico y su imagen asociada, si existe. En primer lugar, se intenta localizar el patrocinador mediante su ID en la base de datos. En caso de éxito, se procede a eliminar la imagen asociada a ese patrocinador utilizando el servicio de almacenamiento de Laravel (Storage). Luego, se elimina el registro del patrocinador de la base de datos. Finalmente, se devuelve una respuesta JSON indicando si la operación fue exitosa, junto con un mensaje correspondiente.

Ruta del archivo: app/Http/Controllers/PatrocinadorController.php

```php
/**
     * Elimina un patrocinador y su imagen asociada.
 */
    public function destroy($id)
    {
        try {
            $patrocinador = Patrocinador::find($id);

            // Elimina la imagen asociada al patrocinador.
            Storage::delete($patrocinador->ruta_imagen);

            // Elimina el patrocinador de la base de datos.
            $patrocinador->delete();

            return response()->json(['mensaje' => 'Eliminado exitosamente', 'error' => false]);
        } catch (QueryException $e) {
            return response()->json(['error' => $e->getMessage()]);
        }
    }
```

Código JavaScript de eliminar patrocinador

El código proporcionado en JavaScript se encarga de gestionar la interfaz de usuario relacionada con la eliminación de patrocinadores en un entorno administrativo. En primer lugar, se definen varias variables que representan elementos del formulario y de la interfaz, como campos de entrada, botones y áreas de previsualización de imágenes. Además, se establecen opciones de configuración para una tabla de patrocinadores, como

el número de registros por página y el orden inicial. La función initDataTable se encarga de inicializar la tabla de patrocinadores, destruyendo la instancia existente si ya está inicializada. Luego, se utiliza el plugin DataTable para formatear las fechas en el formato "DD-MM-YYYY". El evento load de la ventana dispara la inicialización de la tabla cuando la página se carga completamente. La función eliminarPatrocinador realiza una solicitud DELETE a la API para eliminar un patrocinador específico, oculta un modal de confirmación y muestra una alerta con el resultado de la operación. Además, se utiliza la función updateTablaPatrocinadores para redirigir a la página de eliminación de patrocinadores después de un breve período de tiempo, proporcionando así una actualización visual después de la eliminación.

Ruta del archivo: public/js/Patrocinador/eliminarPatrocinador.js

```
// Variables para gestionar la tabla de patrocinadores y su inicialización
let tablaDePatrocinadores;
let tablaInicializada = false;

// Elementos del formulario y la interfaz de usuario
const input = document.getElementById("imageUpload");
const imagenPreview = document.getElementById("imagePreview");
const botonSubir = document.getElementById("botonSubirLogoPatrocinador");
const nombrePatrocinador = document.getElementById("nombrePatricinador");
const urlPatricinador = document.getElementById("urlPatrocinador");
const cancelar = document.getElementById("crearPatrocinadorCancelar");
const asignar = document.getElementById("crearPatrocinadorCrear");

// Opciones para la configuración de la tabla de patrocinadores
const dataTableOptions = {
    pageLength: 10,
    lengthMenu: [5, 10, 15, 20],
    destroy: true,
    order: [[3, "desc"]],
    language: {
        lengthMenu: "Mostrar _MENU_ patrocinadores",
        zeroRecords: "Ningún patrociandor encontrado",
        info: "Mostrando de _START_ a _END_ de un total de _TOTAL_ patrocinadores",
        infoEmpty: "Ningún patrocinador encontrado",
        infoFiltered: "(filtrados desde _MAX_ patrocinadores en total)",
        search: "Buscar:",
        loadingRecords: "Cargando...",
        paginate: {
            first: "Primero",
            last: "Último",
            next: "Siguiente",
            previous: "Anterior",
        },
    },
};
```

```
// Función para inicializar la tabla de patrocinadores
const initDataTable = async () => {
    // Verificar si la tabla ya está inicializada y destruirla si es necesario
    if (tablaInicializada) {
        tablaDePatrocinadores.destroy();
    }
    // Configurar el formato de fecha en la tabla y crear una nueva instancia
    DataTable.datetime("DD-MM-YYYY");
    tablaDePatrocinadores = $("#tablaEvento").DataTable(dataTableOptions);
    tablaInicializada = true;
};

// Evento que se dispara cuando la página se carga completamente
window.addEventListener("load", () => {
    initDataTable();
});

// Función para eliminar un patrocinador mediante una solicitud DELETE a la API
const eliminarPatrocinador = async (id) => {
    await axios.delete("/api/patrocinador/" + id).then((response) => {
        // Ocultar el modal de eliminación
        $('#modalEliminarPatrocinador' + id).modal('hide');
        // Mostrar una alerta con el resultado de la operación
        mostrarAlerta(
            "Éxito",
            response.data.mensaje,
            response.error ? "danger" : "success"
        );
        // Actualizar la tabla de patrocinadores después de un tiempo de espera
        updateTablaPatrocinadores();
    });
};

// Función para redirigir a la página de eliminación de patrocinadores después de una actualización
const updateTablaPatrocinadores = () => {
    setTimeout(() => {
        window.location.href = "/admin/patrocinador/eliminar";
    }, 1700);
};
```

Vista de eliminar patrocinador

La vista presenta una tabla que muestra información detallada de cada patrocinador, incluyendo nombre, enlace web, imagen, fecha de creación y el número de eventos asociados. Además, se incluyen botones de acción, como el botón de eliminación, que abre un modal de confirmación. La tabla se alimenta con datos provenientes de la variable $patrocinadores, que se itera mediante una estructura @foreach para mostrar los detalles de cada patrocinador. Para mejorar la interfaz, se utiliza Bootstrap Icons para representar iconos visuales, como el ícono de papelera para la acción de eliminación. Cada patrocinador en la tabla tiene asociado un modal de eliminación (modalEliminarPatrocinador) que se abre al hacer clic en el botón de eliminación. Este modal incluye un mensaje de confirmación estándar, pero en caso de que el patrocinador tenga eventos asociados, se muestra un mensaje adicional destacando esta relación.

Ruta del archivo: resources/views/patrocinador/eliminarPatrocinador.blade.php

```
@extends('layouts.app')

@section('content')
    <div class="container">
        <div class="row mb-2">
            <h2>Eliminar patrocinador</h2>
        </div>
        <div class="row g-5">
            <div class="col-sm-12 col-md-12">
                <!-- Tabla que muestra la información de los patrocinadores -->
                <table class="table table-responsive table-striped text-secondary" id="tablaEvento">
                    <caption>Patrocinadores</caption>
                    <thead>
                        <tr>
                            <th scope="col" class="col-sm-3 col-md-3">Nombre del patrocinador</th>
                            <th scope="col" class="col-sm-3 col-md-3">Enlace web</th>
                            <th scope="col" class="col-sm-3 col-md-3">Imagen</th>
                            <th scope="col" class="col-sm-1 col-md-1">Fecha de creación</th>
                            <th scope="col" class="col-sm-1 col-md-1">Eventos asociados</th>
                            <th scope="col" class="col-sm-1 col-md-1">Acción</th>
                        </tr>
                    </thead>
                    <tbody id="datosTabla">
                        <!-- Iteración sobre los patrocinadores para mostrar sus detalles -->
                        @foreach ($patrocinadores as $patrocinador)
                            <tr id="{{ $patrocinador->id }}">
                                <td>{{ $patrocinador->nombre }}</td>
                                <td>
                                    <a class="d-inline-block text-truncate" href="{{ $patrocinador->enlace_web }}"
                                        target="_blank" style="max-width: 250px;" title="{{ $patrocinador->enlace_web }}">
                                        {{ $patrocinador->enlace_web }}
                                    </a>
                                </td>
                                <td><a href="{{ $patrocinador->ruta_imagen }}"
                                        target="_blank">{{ $patrocinador->nombre }}</a></td>
                                <td>{{ date('d-m-Y', strtotime($patrocinador->created_at)) }}</td>
                                <td class="text-center">{{ $patrocinador->eventoPatrocinador->count() }}</td>
                                <td class="text-center">
                                    <!-- Botón para abrir el modal de eliminación -->
                                    <button data-bs-toggle="modal"
                                        data-bs-target="#modalEliminarPatrocinador{{ $patrocinador->id }}"
                                        title="Eliminar patrocinador" type="button" class="btn btn-danger btn-sm">
                                        <i class="bi bi-trash"></i>
                                    </button>
                                    <!-- Modal de confirmación para la eliminación -->
                                    @component('components.modal')
                                        @slot('modalId', 'modalEliminarPatrocinador' . $patrocinador->id)
                                        @slot('modalTitle', 'Eliminar patrocinador')
                                        @slot('modalContent')
                                            @if ($patrocinador->eventoPatrocinador->count() > 0)
                                                <p>
                                                    <b>Este patrocinador tiene eventos asociados,</b>
                                                    ¿Está seguro que quiere eliminar este patrocinador?
                                                </p>
                                            @else
                                                ¿Está seguro que quiere eliminar este patrocinador?
                                            @endif
                                        @endslot
                                        @slot('modalButton')
```

```
                                    <!-- Botones de confirmación o cancelación en el modal -->
                                    <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">No</button>
                                    <button type="button" class="btn btn-danger"
                                        onclick="eliminarPatrocinador({{ $patrocinador->id }})">Sí</button>
                                @endslot
                            @endcomponent
                        </td>
                    </tr>
                @endforeach
            </tbody>
        </table>
    </div>
</div>
</div>

<!-- Inclusión de estilos y scripts externos para DataTables y Bootstrap Icons -->
<link href="https://cdn.datatables.net/1.13.6/css/dataTables.bootstrap5.min.css" rel="stylesheet">
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.11.1/font/bootstrap-icons.css">
<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.7.1/jquery.min.js"></script>
<script src="https://cdn.datatables.net/1.13.6/js/jquery.dataTables.min.js"></script>
<script src="https://cdn.datatables.net/1.13.6/js/dataTables.bootstrap5.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/moment.js/2.29.2/moment.min.js"></script>
<!-- Inclusión del script personalizado eliminarPatrocinador.js -->
<script src="{{ asset('js/Patrocinador/eliminarPatrocinador.js') }}" defer></script>
@endsection
```

7.5.3.3. Interfaz de usuario de eliminar patrocinador

La interfaz de usuario destinada a la eliminación de patrocinadores ofrece una experiencia clara y eficiente para la gestión de estos asociados a eventos. A continuación, se detallan los elementos y características clave de la interfaz:

Tabla de Patrocinadores:

Nombre del Patrocinador: Identifica claramente el nombre de cada patrocinador presente en la tabla.

Enlace Web: Muestra el enlace web asociado al patrocinador para facilitar la navegación.

Imagen: Presenta la imagen del patrocinador para una identificación visual rápida.

Fecha de Creación: Indica la fecha en que se creó el patrocinador, proporcionando contexto temporal.

Eventos Asociados: Muestra el número de eventos vinculados a cada patrocinador para comprender su alcance.

Acciones:

Botón de Eliminación: Cada patrocinador presenta un botón de eliminación que desencadena un modal de confirmación al hacer clic.

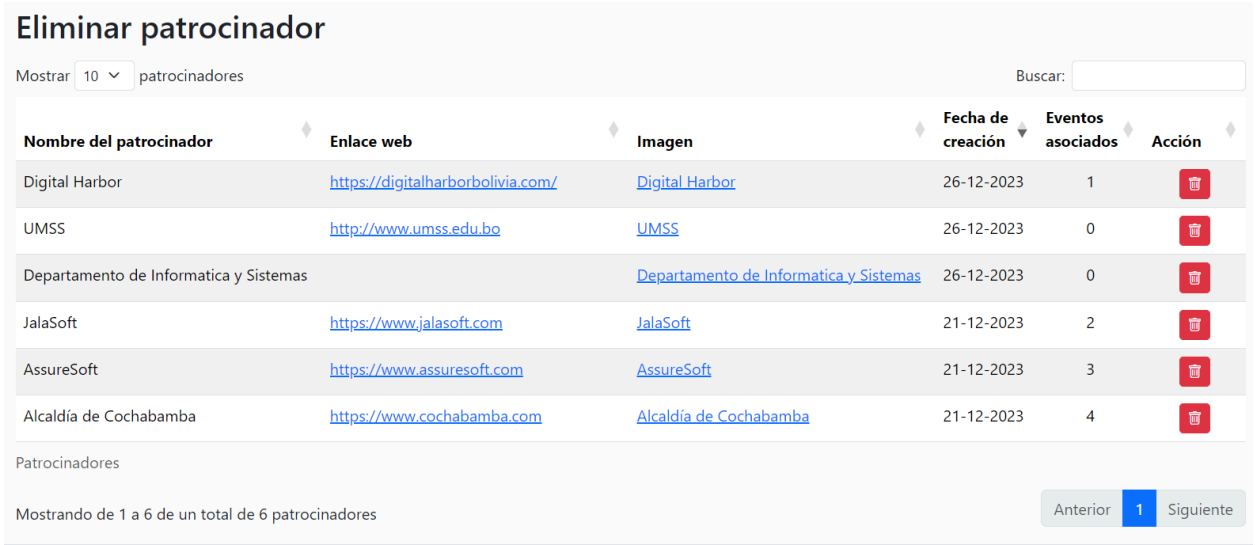
Modal de Confirmación:

Mensaje de Confirmación: El modal incluye un mensaje estándar de confirmación de eliminación.

Mensaje Adicional (si aplica): En caso de que el patrocinador tenga eventos asociados, se agrega un mensaje adicional que destaca esta relación.

Diseño y Usabilidad:

Interfaz Clara y Organizada: El diseño se presenta de manera clara, proporcionando una experiencia intuitiva.
Confirmación Visual: El uso de modales y mensajes de confirmación visual ofrece claridad al usuario durante el proceso de eliminación.



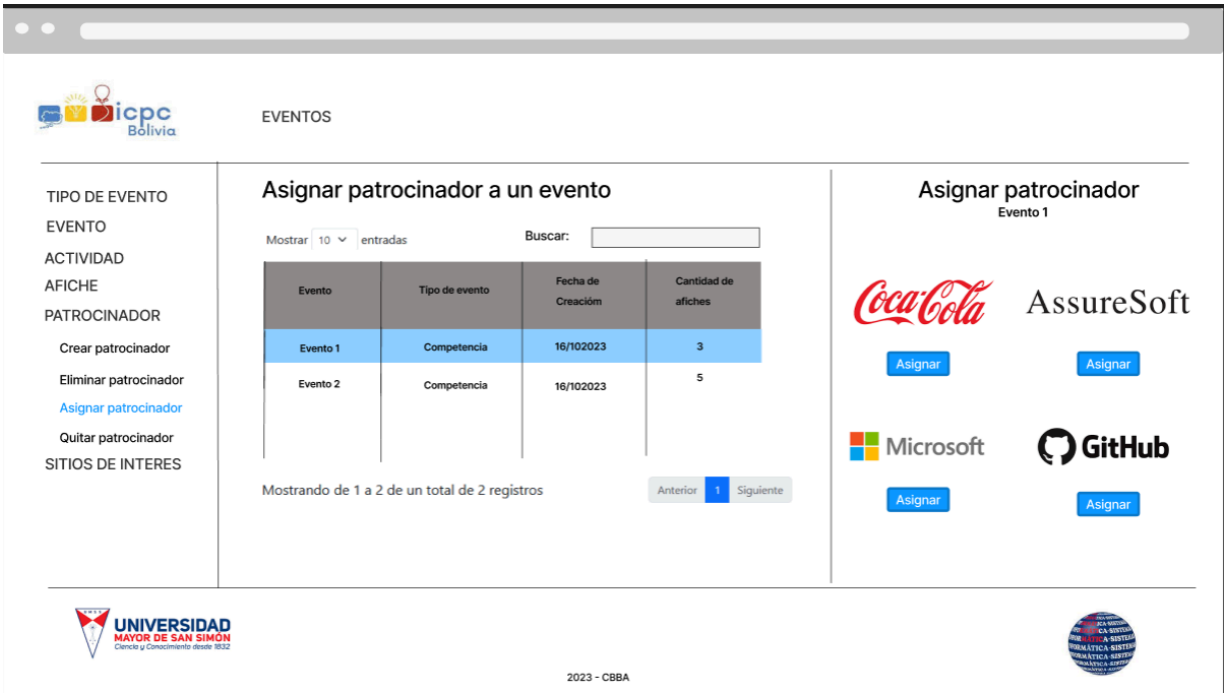
7.5.4. Asignar patrocinador

7.5.4.1. Historia de usuario de asignar patrocinador

Historia de Usuario	
Título: Asignar patrocinador a un evento	**ID: 8**
Estimación: 5	**Importancia: Media**
Descripción	

Yo como usuario con privilegios para asignar patrocinadores a un evento, quiero asignar patrocinadores a los eventos que tengo para promover la colaboración y patrocinio del evento.

Mockups



Criterios de Aceptación

1. Al hacer clic en la sección de “PATROCINADOR” en el menú lateral y seleccionar la opción “Asignar patrocinador”, se muestra una tabla con los eventos creados previamente.
2. La tabla tiene las columnas “Nombre del evento”, “Tipo de evento”, “Fecha de creación” y “Cantidad de patrocinadores”.
3. Al seleccionar un evento cualquiera en la tabla de eventos, se muestran en la parte derecha los patrocinadores aún no asignados al evento.
4. Los patrocinadores se ven en tarjetas, cada tarjeta contiene su imagen, su nombre y un botón “Asignar”.
5. Al hacer clic en el botón “Asignar”, se asigna el patrocinador seleccionado.
6. Si se asigna correctamente un patrocinador a un evento, se muestra una alerta temporal con el mensaje: “Asignado exitosamente”.
7. Si se asigna correctamente un patrocinador a un evento, el valor en la columna “Cantidad de patrocinadores” del evento seleccionado se incrementa en uno.

7.5.4.2. Código fuente de asignar patrocinador

Controlador de asignar patrocinador

La función asignarPatrocinador en el controlador PatrocinadorController facilita la asignación de un patrocinador a un evento. Al recibir una solicitud HTTP que incluye los parámetros necesarios, como 'id_evento' y 'id_patrocinador', la función crea una nueva instancia del modelo EventoPatrocinador. Luego, asigna los valores de los parámetros de la solicitud a las propiedades correspondientes del modelo, estableciendo la asociación entre el evento y el patrocinador. La función guarda esta asociación en la base de datos y responde con un mensaje indicando el éxito de la asignación y la ausencia de errores, o, en caso de error durante el proceso, proporciona detalles del error con un código HTTP 500. La documentación detallada mejora la comprensión y mantenimiento del código al describir de manera clara los pasos realizados y los posibles problemas que pueden surgir.

Ruta del archivo: app/Http/Controllers/PatrocinadorController.php

```php
/**
 * Asigna un patrocinador a un evento.
 */
public function asignarPatrocinador(Request $request)
{
    try {
        // Se crea una nueva instancia del modelo EventoPatrocinador.
        $patrocinador = new EventoPatrocinador();
        // Se asignan los valores provenientes de la solicitud a las propiedades
        // correspondientes de la instancia del modelo.
        $patrocinador->id_evento = $request->id_evento;
        $patrocinador->id_patrocinador = $request->id_patrocinador;
        // Se guarda la asociación entre el evento y el patrocinador en la base de datos.
        $patrocinador->save();
        // Se devuelve una respuesta JSON indicando que la asignación se realizó
        // exitosamente y sin errores.
        return response()->json(['mensaje' => 'Asignado exitosamente', 'error' => false]);
    } catch (QueryException $e) {
        // En caso de que ocurra un error durante la asignación, se devuelve una
        // respuesta JSON con detalles del error y un código HTTP 500.
        return response()->json(['error' => $e->getMessage()], 500);
    }
}
```

Codigo JavaScript de asignar patrocinador

Utiliza la biblioteca DataTables para presentar una tabla de eventos, permitiendo la selección de un evento específico. Al seleccionar un evento, se carga dinámicamente una lista de patrocinadores no asignados a ese evento y se muestran en tarjetas con la opción de asignarlos. La interfaz incluye una tabla que presenta eventos, con información como el nombre y el número de patrocinadores asignados. Al seleccionar un evento, se resalta visualmente y se actualiza el área de patrocinadores disponibles. Cada patrocinador se representa con una tarjeta que muestra su nombre y logo, y se proporciona un botón para asignarlo al evento seleccionado. El código realiza solicitudes a una API para obtener y manipular datos relacionados con eventos y patrocinadores. Además, utiliza técnicas asincrónicas para gestionar la interfaz de usuario de manera eficiente, como la

actualización dinámica de la tabla y la gestión de eventos de asignación. Se implementa un mecanismo de control para evitar asignaciones simultáneas, y se utiliza la biblioteca axios para realizar solicitudes HTTP de manera asíncrona.

Ruta del archivo: public/js/Patrocinador/asignarPatrocinador.js

```
// Declaración de variables globales
let tablaDeTipos; // Variable para almacenar la referencia de la tabla DataTable
let tablaInicializada = false; // Bandera que indica si la tabla ha sido inicializada
const eventoSeleccionado = document.getElementById("nombreEvento"); // Elemento HTML que muestra el evento
let patrocinadores; // Almacena la lista de patrocinadores disponibles

// Opciones para la configuración de la tabla DataTables
const dataTableOptions = {
    pageLength: 10, // Número de filas por página
    lengthMenu: [5, 10, 15, 20], // Opciones de longitud de página
    destroy: true, // Destruir la tabla antes de volver a inicializar
    order: [2, 'desc'], // Ordenar por la tercera columna en orden descendente al iniciar
    ordering: true, // Permitir ordenar las columnas
    language: {
        // Configuración del idioma para DataTables
        lengthMenu: "Mostrar _MENU_ eventos",
        zeroRecords: "Ningún evento encontrado",
        info: "Mostrando de _START_ a _END_ de un total de _TOTAL_ eventos",
        infoEmpty: "Ningún evento encontrado",
        infoFiltered: "(filtrados desde _MAX_ eventos totales)",
        search: "Buscar:",
        loadingRecords: "Cargando...",
        paginate: {
            first: "Primero",
            last: "Último",
            next: "Siguiente",
            previous: "Anterior",
        },
    },
};
// Inicialización de la tabla DataTable
const initDataTable = async () => {
    if (tablaInicializada) {
        tablaDeTipos.destroy();
    }
    DataTable.datetime("DD-MM-YYYY"); // Configuración del formato de fecha para DataTables
    tablaDeTipos = $("#tablaEvento").DataTable(dataTableOptions);
    tablaInicializada = true;
};
```

```
// Evento de carga de la página
window.addEventListener("load", () => {
    initDataTable(); // Inicializar la tabla
    if (!seleccionado) {
        eventoSeleccionado.textContent = "Seleccione un evento";
    }
    getPatrocinadores(); // Obtener la lista de patrocinadores disponibles
});

// Función para obtener la lista de patrocinadores mediante una solicitud a la API
const getPatrocinadores = async () => {
    let datos = await axios.get("/api/patrocinador").then((response) => {
        return response.data;
    });
    patrocinadores = await datos;
};

// Declaración de variables para la gestión de eventos y patrocinadores
let seleccionado; // Elemento HTML de la fila seleccionada en la tabla
let idSeleccionado; // ID del evento seleccionado
let asignando = false; // Bandera que indica si se está realizando una asignación
let patrocinadoresNoAsigandos; // Lista de patrocinadores no asignados al evento seleccionado

// Función para seleccionar un evento
const seleccionarEvento = async (id, nombre) => {
    if (seleccionado) {
        seleccionado.classList.remove("table-primary");
    }
    seleccionado = document.getElementById(id);
    seleccionado.classList.add("table-primary");
    idSeleccionado = id;
    eventoSeleccionado.textContent = nombre;
    await cargarPatrocinadoresNoAsignados();
    mostrarPatrocinadores();
};
```

```
// Función para cargar la lista de patrocinadores no asignados al evento seleccionado
const cargarPatrocinadoresNoAsignados = async () => {
    let patrocinadoresEvento = await getPatrocinadoresEvento(idSeleccionado);
    patrocinadoresNoAsigandos = getNoAsignados(patrocinadoresEvento);
};

// Función para obtener los patrocinadores asignados a un evento mediante una solicitud a la API
const getPatrocinadoresEvento = async (id) => {
    let data = await axios.get("/api/patrocinador/" + id).then((response) => {
        return response.data;
    });
    return data;
};

// Función para mostrar los patrocinadores no asignados en la interfaz
const mostrarPatrocinadores = () => {
    let div = document.getElementById('divPatrocinadores');
    let content = "";
    patrocinadoresNoAsigandos.map((patrocinador) => {
        content += `
            <div class="col text-center">
                <div class="card" style="height: 13rem">
                    <div class="row justify-content-center" style="height: 125px">
                        <img src="${patrocinador.ruta_imagen}" class="img-fluid object-fit-scale"
                            alt="logoPatrocinador" style="max-height: 100%; width: auto">
                    </div>
                    <div class="card-body">
                        <h6 class="card-title text-truncate" title="${patrocinador.nombre}">
                            ${patrocinador.nombre}
                        </h6>
                        <button class="btn btn-primary btn-sm" onclick="asignarPatrocinador(
                        ${patrocinador.id})" ${asignando ? "disabled" : ""}>Asignar</button>
                    </div>
                </div>
            </div>
        `;
    });
    div.innerHTML = content;
};

// Función para obtener los patrocinadores no asignados
const getNoAsignados = (patrocinadoresEvento) => {
    let patrocinadoresNoAsignados = [];
    let indice = 0;
    patrocinadores.map((patrocinador) => {
        if (indice < patrocinadoresEvento.length && patrocinadoresEvento[indice].id_patrocinador ===
        patrocinador.id) {
            indice++;
        } else {
            patrocinadoresNoAsignados.push(patrocinador);
        }
    });
    return patrocinadoresNoAsignados;
};

// Función para asignar un patrocinador a un evento
const asignarPatrocinador = async (idPatrocinador) => {
    asignando = true;
    mostrarPatrocinadores();
    let res = await realizarPeticion(idPatrocinador);
    if (res === "Asignado exitosamente") {
        updateNroPatrocinadores();
        await cargarPatrocinadoresNoAsignados();
    }
    setTimeout(() => {
        asignando = false;
        mostrarPatrocinadores();
    }, 2000);
};
```

```
// Función para realizar la solicitud de asignación de patrocinador a un evento
const realizarPeticion = async (idPatrocinador) => {
    let formData = new FormData();
    formData.append('id_evento', idSeleccionado);
    formData.append('id_patrocinador', idPatrocinador);
    let data = await axios.post("/api/patrocinador/asignar", formData).then((response) => {
        mostrarAlerta(
            "Éxito",
            response.data.mensaje,
            response.error ? "danger" : "success"
        );
        return response.data.mensaje;
    });
    return data;
};

// Función para actualizar el contador de patrocinadores en la tabla
const updateNroPatrocinadores = () => {
    let casilla = document.getElementById(
        `contadorPatrocinadores${idSeleccionado}`
    );
    let valor = parseInt(casilla.textContent);
    casilla.textContent = valor + 1;
};

// Configuración de ResizeObserver para ajustar la altura del contenedor de patrocinadores
const resize_ob = new ResizeObserver(function (entries) {
    let rect = entries[0].contentRect;
    let height = rect.height;
    vh = parseInt((height / window.innerHeight) * 78) + 1;
    document.getElementById("divPatrocinadores").style.maxHeight = vh + "vh";
});

// Observa cambios en la altura del contenedor y ajusta dinámicamente la altura máxima
resize_ob.observe(document.getElementById("tablaEventos"));
```

Vista de asignar patrocinador

La vista presenta una tabla que muestra eventos futuros con detalles como nombre, tipo, fecha de creación y cantidad actual de patrocinadores. Al hacer clic en un evento, se resalta y se actualiza un panel lateral que indica el evento seleccionado. En dicho panel, se muestra una lista de patrocinadores disponibles, cada uno con su imagen y nombre, junto con un botón de "Asignar" deshabilitado. La interfaz utiliza DataTables para la paginación y presentación ordenada de eventos.

Ruta del archivo: resources/views/patrocinador/asignarPatrocinador.blade.php

```
@extends('layouts.app')

@section('content')
    <div class="container">
        <div class="row mb-2">
            <h2>Asignar patrocinador a un evento</h2>
        </div>
        <div class="row g-5">
            <!-- Tabla de Eventos -->
            <div class="col-sm-12 col-md-8" id="tablaEventos">
                <table class="table table-responsive table-striped text-secondary table-hover cursor" id="tablaEvento">
                    <caption>Eventos</caption>
                    <thead>
                        <tr>
                            <th scope="col" class="col-sm-4 col-md-4">Nombre del evento</th>
                            <th scope="col" class="col-sm-3 col-md-3">Tipo de evento</th>
                            <th scope="col" class="col-sm-3 col-md-3 text-center">Fecha de creación</th>
                            <th scope="col" class="col-sm-2 col-md-2 text-center font-sm">Cantidad de patrocinadores</th>
                        </tr>
                    </thead>
                    <tbody id="datosTabla">
                        <!-- Iteración sobre eventos -->
                        @foreach ($eventos as $evento)
                            <!-- Mostrar solo eventos futuros -->
                            @if (strtotime($evento->fin_evento) >= time())
                                <tr onclick="seleccionarEvento({{ $evento->id }}, '{{ $evento->nombre }}', event)"
                                    id="{{ $evento->id }}">
                                    <td>{{ $evento->nombre }}</td>
                                    <td>{{ $evento->tipoEvento->nombre }}</td>
                                    <td class="text-center">{{ date('d-m-Y', strtotime($evento->created_at)) }}</td>
                                    <td class="text-center" id="contadorPatrocinadores{{ $evento->id }}">
                                        {{ $evento->eventoPatrocinador->count() }}</td>
                                </tr>
                            @endif
                        @endforeach
                    </tbody>
                </table>
            </div>

            <!-- Panel de Asignación -->
            <div class="col-sm-12 col-md-4">
                <div class="container d-flex flex-column border p-3">
                    <div class="col-12 d-flex justify-content-center align-items-center">
                        <h4>Asignar patrocinador a:</h4>
                    </div>
                    <h5 id="nombreEvento" class="text-center fw-bold"></h5>

                    <!-- Lista de Patrocinadores -->
                    <div class="col-md-12">
                        <div class="row row-cols-2 g-3 mt-2 pb-2" style="overflow-y: auto; overflow-x: hidden;"
                            id="divPatrocinadores">
                            <!-- Iteración sobre patrocinadores -->
                            @foreach ($patrocinadores as $patrocinador)
                                <div class="col text-center">
                                    <div class="card" style="height: 13rem">
                                        <div class="row justify-content-center" style="height: 125px">
                                            <img src="{{ $patrocinador->ruta_imagen }}" class="img-fluid object-fit-scale"
                                                alt="logoPatrocinador" style="max-height: 100%; width: auto">
                                        </div>
```

```
                                <div class="card-body">
                                    <h6 class="card-title text-truncate" title="{{ $patrocinador->nombre }}">
                                        {{ $patrocinador->nombre }}
                                    </h6>
                                    <!-- Botón de asignar (actualmente deshabilitado) -->
                                    <button class="btn btn-primary btn-sm" disabled>Asignar</button>
                                </div>
                            </div>
                        </div>
                    @endforeach
                </div>
            </div>
        </div>
    </div>

    <!-- Enlaces y scripts -->
    <link href="https://cdn.datatables.net/1.13.6/css/dataTables.bootstrap5.min.css" rel="stylesheet">
    <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.7.1/jquery.min.js"></script>
    <script src="https://cdn.datatables.net/1.13.6/js/jquery.dataTables.min.js"></script>
    <script src="https://cdn.datatables.net/1.13.6/js/dataTables.bootstrap5.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/moment.js/2.29.2/moment.min.js"></script>
    <script src="{{ asset('js/Patrocinador/asignarPatrocinador.js') }}" defer></script>
@endsection
```

7.5.4.3. Interfaz de usuario de asignar patrocinador

La interfaz de usuario destinada a la asignación de patrocinadores a eventos brinda una experiencia eficaz y fácil de usar para gestionar estas asociaciones. Aquí se detallan los elementos y características clave de la interfaz:

Tabla de Eventos:

Nombre del evento: Identifica de manera distintiva el nombre de cada evento presente en la tabla.

Tipo de evento: Muestra el tipo de evento al que se puede asignar un patrocinador para una referencia rápida.

Fecha de creación: Indica la fecha en que se creó el evento, proporcionando contexto temporal.

Cantidad de patrocinadores: Muestra el número actual de patrocinadores asociados a cada evento para comprender su nivel de participación**.**

Acciones:

Selección de evento: Al hacer clic en un evento específico, este se resalta, y se actualiza el panel lateral con los detalles del evento seleccionado.

Botones de asignación: Cada patrocinador disponible para asignación tiene un botón "Asignar" deshabilitado que, al hacer clic, inicia el proceso de asignación al evento seleccionado. Este botón se desactiva temporalmente durante el proceso para evitar operaciones duplicadas.

Panel Lateral:

Nombre del evento seleccionado: Muestra el nombre del evento seleccionado, brindando claridad sobre la acción que se está realizando.

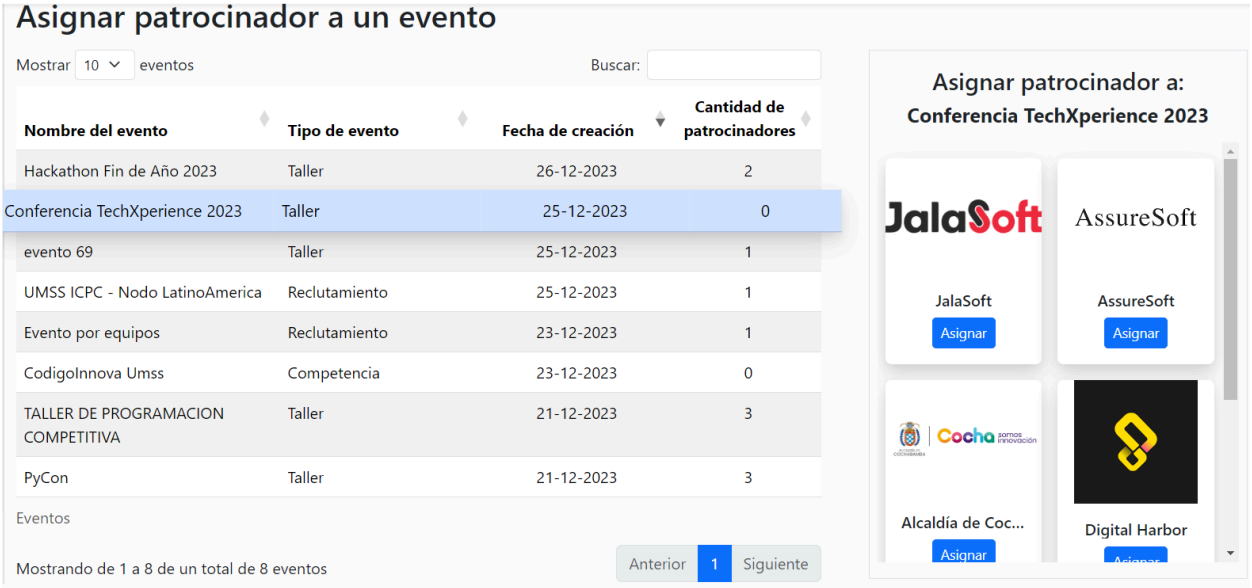
Lista de patrocinadores disponibles: Presenta una cuadrícula con imágenes y nombres de patrocinadores no asignados al evento, permitiendo una asignación rápida y sencilla.

Diseño y Usabilidad:

Interfaz intuitiva: La disposición clara de la información facilita la asignación de patrocinadores a eventos.

Confirmación visual: El sistema utiliza mensajes y desactivación temporal de botones para proporcionar una confirmación visual del estado de la operación.

Eficiencia operativa: La interfaz está diseñada para facilitar la asignación eficiente de patrocinadores, mejorando la experiencia del usuario.

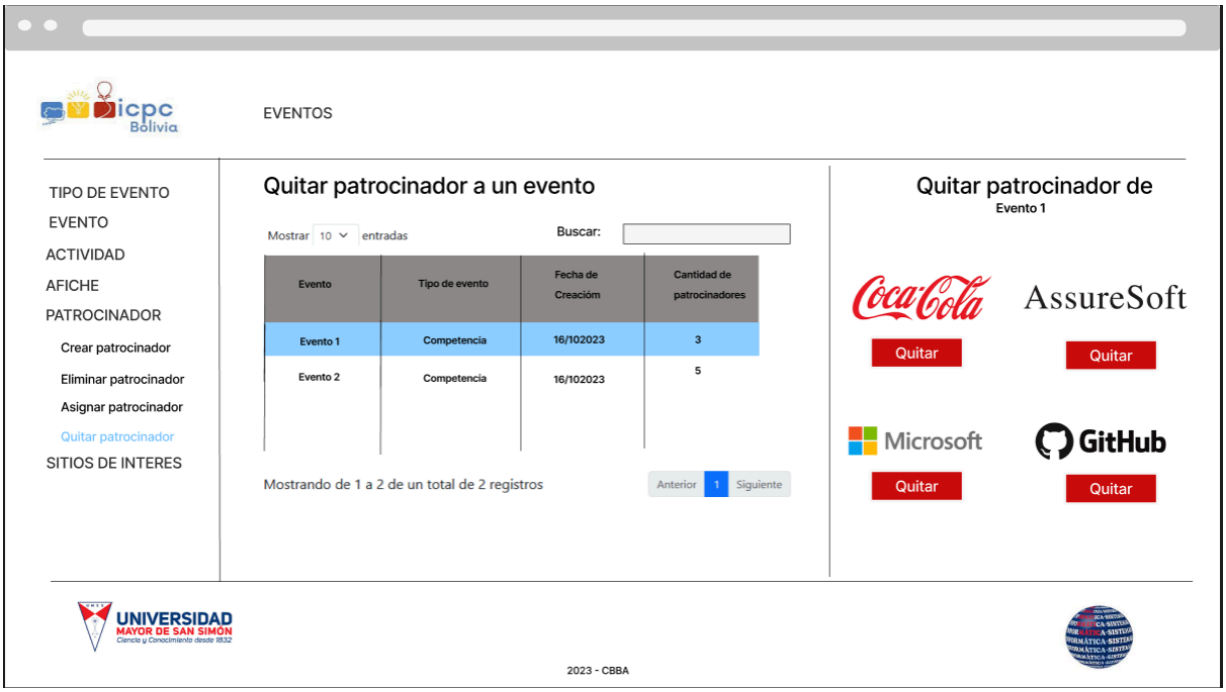


7.5.5. Quitar patrocinador

7.5.5.1. Historia de usuario de quitar patrocinador

Historia de Usuario	
Título: Quitar patrocinador de un evento	**ID: 18**
Estimación: 5	**Importancia: Media**
Descripción Como usuario con privilegio para quitar patrocinador de un evento, deseo tener la capacidad de quitar patrocinadores asociados a un evento específico, para mantener actualizada y precisa la información de los patrocinadores asignados a un evento.	

Mockups



Criterios de Aceptación

1. Al hacer clic en la sección “PATROCINADOR” en el menú lateral y seleccionar la opción “Quitar patrocinador” se muestra una tabla con los eventos existentes.
2. La tabla contiene las columnas “Nombre del evento”, “Tipo de evento”, “Fecha de creación”, “Cantidad de patrocinadores”.
3. En la columna "Cantidad de patrocinadores" de la tabla, se muestran la cantidad de patrocinadores que se asignaron a los eventos existentes.
4. Al hacer clic sobre un evento específico de la tabla, se muestran los patrocinadores asignados a ese evento.
5. Los patrocinadores se muestran con su imagen, nombre y el botón “Quitar”.
6. Si el nombre del patrocinador es muy largo se trunca el texto con "...".
7. Si el nombre del patrocinador esta truncado al pasar el puntero del raton sobre el nombre, se muestra un texto emergente que muestra el nombre completo del patrocinador.
8. Al hacer clic sobre el botón “Quitar” se muestra un modal de confirmación con el mensaje “¿Está seguro de quitar este patrocinador?” y los botones de “No” y “Sí”.
9. Al hacer clic en el botón “No” se cierra el modal.
10. Al hacer clic en el botón “Si”, después de un breve periodo de tiempo se quita el patrocinador del evento.
11. Al hacer clic en el botón “Si” se muestra una alerta temporal con el mensaje “Quitado exitosamente”

12. Al quitar un patrocinador de un evento, el valor en la columna “Cantidad de patrocinadores” se reduce en 1.

7.5.5.2. Código fuente de quitar patrocinador

Controlador de quitar patrocinador

Esta función se encarga de desvincular un patrocinador de un evento, eliminando la relación correspondiente en la base de datos. Primero, busca el evento-patrocinador específico mediante su identificador, y luego procede a eliminarlo. En caso de éxito, devuelve una respuesta JSON indicando que la operación se ha llevado a cabo sin problemas. Por otro lado, si ocurre algún error durante el proceso, la función devuelve un mensaje de error también en formato JSON, proporcionando información detallada sobre la excepción capturada.

Ruta del archivo: app/Http/Controllers/PatrocinadorController.php

```
/**
     * Quita un patrocinador de un evento.
*/
    public function quitarPatrocinador($id)
    {
        try {
            // Buscar y eliminar el EventoPatrocinador por su ID.
            $patrocinador = EventoPatrocinador::find($id);
            $patrocinador->delete();

            // Respuesta exitosa.
            return response()->json(['mensaje' => 'Quitado exitosamente', 'error' => false]);
        } catch (QueryException $e) {
            // En caso de error, devolver el mensaje de error.
            return response()->json(['error' => $e->getMessage()]);
        }
    }
```

Codigo JavaScript de quitar patrocinador

Utiliza la biblioteca DataTable para mostrar una tabla de eventos con diversas funcionalidades, como búsqueda, paginación y ordenamiento. La función initDataTable inicializa y configura la DataTable al cargar la página. La selección de eventos y la carga dinámica de patrocinadores se realizan mediante las funciones seleccionarEvento y cargarPatrocinadores, respectivamente. Los patrocinadores se presentan en tarjetas con información visual, y cada tarjeta cuenta con un botón "Quitar" para eliminar la asociación de un patrocinador con un evento específico. El código también gestiona eventos como la apertura de un modal de confirmación al hacer clic en el botón "Quitar", así como la desactivación temporal del botón durante la ejecución de operaciones para evitar acciones duplicadas. Además, se utiliza un objeto ResizeObserver para ajustar dinámicamente la altura máxima de la sección de patrocinadores en función del tamaño de la tabla de eventos, mejorando la usabilidad de la interfaz.

Ruta del archivo: public/js/Patrocinador/quitarPatrocinador.js

```
// Variables globales para la tabla de eventos y su inicialización
let tablaDeTipos;
let tablaInicializada = false;

// Elemento del DOM para mostrar el nombre del evento seleccionado
const eventoSeleccionado = document.getElementById("nombreEvento");

// Opciones de configuración para la DataTable
const dataTableOptions = {
    pageLength: 10,
    lengthMenu: [5, 10, 15, 20],
    destroy: true,
    order: [2, "desc"],
    ordering: true,
    language: {
        // Configuración de mensajes y etiquetas para la DataTable
        lengthMenu: "Mostrar _MENU_ eventos",
        zeroRecords: "Ningún evento encontrado",
        info: "Mostrando de _START_ a _END_ de un total de _TOTAL_ eventos",
        infoEmpty: "Ningún evento encontrado",
        infoFiltered: "(filtrados desde _MAX_ eventos totales)",
        search: "Buscar:",
        loadingRecords: "Cargando...",
        paginate: {
            first: "Primero",
            last: "Último",
            next: "Siguiente",
            previous: "Anterior",
        },
    },
};
```

```
// Función para inicializar la DataTable al cargar la página
const initDataTable = async () => {
    if (tablaInicializada) {
        tablaDeTipos.destroy();
    }
    DataTable.datetime("DD-MM-YYYY");
    tablaDeTipos = $("#tablaEvento").DataTable(dataTableOptions);
    tablaInicializada = true;
};

// Evento que se ejecuta al cargar la página para inicializar la DataTable
window.addEventListener("load", () => {
    initDataTable();
    if (!seleccionado) {
        eventoSeleccionado.textContent = "Seleccione un evento";
    }
});

// Variables para gestionar la selección de eventos y patrocinadores
let seleccionado;
let idSeleccionado;
let patrocinadores;
let quitando = false;

// Función para seleccionar un evento y cargar sus patrocinadores
const seleccionarEvento = async (id, nombre) => {
    if (seleccionado) {
        seleccionado.classList.remove("table-primary");
    }
    seleccionado = document.getElementById(id);
    seleccionado.classList.add("table-primary");
    idSeleccionado = id;
    eventoSeleccionado.textContent = nombre;
    await cargarPatrocinadores();
    mostrarPatrocinadores();
};
```

```
// Función para cargar los patrocinadores de un evento
const cargarPatrocinadores = async () => {
    let data = await getPatrocinadoresEvento(idSeleccionado);
    patrocinadores = await data;
};
// Función para obtener los patrocinadores de un evento mediante una petición API
const getPatrocinadoresEvento = async (id) => {
    let data = await axios
        .get("/api/patrocinador/evento/" + id)
        .then((response) => {
            return response.data;
        });
    return data;
};
// Función para mostrar los patrocinadores en el DOM
const mostrarPatrocinadores = () => {
    let div = document.getElementById("divPatrocinadores");
    let content = "";
    patrocinadores.map((evento) => {
        content += `
            <div class="col text-center">
                <div class="card" style="height: 13rem">
                    <div class="row justify-content-center" style="height: 125px">
                        <img src="${evento.patrocinadores.ruta_imagen}" class=
                        "img-fluid object-fit-scale"
                            alt="logoPatrocinador" style="max-height: 100%; width: auto">
                    </div>
                    <div class="card-body">
                        <h6 class="card-title text-truncate" title="${
                            evento.patrocinadores.nombre
                        }">
                            ${evento.patrocinadores.nombre}
                        </h6>
                        <button class="btn btn-danger btn-sm"
                            data-bs-toggle="modal" data-bs-target="#modalQuitarPatrocinador"
                            onclick="seleccionPatrocinador(${evento.id})" ${
            quitando ? "disabled" : ""
        }>Quitar</button>
                    </div>
                </div>
```

```
                </div>
            `;
        });
        div.innerHTML = content;
};

// Variable para almacenar el ID del patrocinador seleccionado para quitar
let patrocinadorSeleccionado;

// Función para registrar el patrocinador seleccionado para quitar
const seleccionPatrocinador = (id) => {
    patrocinadorSeleccionado = id;
};

// Función para quitar un patrocinador de un evento
const quitarPatrocinador = async () => {
    if (patrocinadorSeleccionado) {
        quitando = true;
        mostrarPatrocinadores();
        let resultado = await realizarPeticion(patrocinadorSeleccionado);
        if (resultado === "Quitado exitosamente") {
            updateNroPatrocinadores();
            await cargarPatrocinadores();
        }
        setTimeout(() => {
            quitando = false;
            mostrarPatrocinadores();
        }, 2000);
    }
};

// Función para realizar la petición API de quitar un patrocinador
const realizarPeticion = async (idEventoPatrocinador) => {
    let data = await axios
        .delete("/api/patrocinador/quitar/" + idEventoPatrocinador)
        .then((response) => {
            mostrarAlerta(
                "Éxito",
                response.data.mensaje,
                response.error ? "danger" : "success"
```

```
                );
                return response.data.mensaje;
            });
        return data;
};

// Función para actualizar el número de patrocinadores en la interfaz
const updateNroPatrocinadores = () => {
    let casilla = document.getElementById(
        `contadorPatrocinadores${idSeleccionado}`
    );
    let valor = parseInt(casilla.textContent);
    casilla.textContent = valor - 1;
};

// Configuración del objeto ResizeObserver para ajustar la altura máxima
// de la sección de patrocinadores según el tamaño de la tabla de eventos
const resize_ob = new ResizeObserver(function (entries) {
    let rect = entries[0].contentRect;
    let height = rect.height;
    vh = parseInt((height / window.innerHeight) * 78) + 1;
    document.getElementById("divPatrocinadores").style.maxHeight = vh + "vh";
});

// Observación del elemento de la tabla de eventos para ajustar la altura
resize_ob.observe(document.getElementById("tablaEventos"));
```

Vista de quitar patrocinador

La vista presenta una estructura dividida en dos secciones principales: la primera muestra una tabla de eventos con información relevante, como el nombre del evento, el tipo de evento, la fecha de creación y la cantidad de patrocinadores asociados. Estos eventos son generados dinámicamente a partir de datos provenientes del servidor. La segunda sección, ubicada en la parte derecha de la página, proporciona detalles específicos sobre el evento seleccionado, incluyendo su nombre. También presenta una lista de patrocinadores asociados a dicho evento, que se actualiza dinámicamente mediante el uso de JavaScript. Adicionalmente, se implementa un componente de modal que se activa al intentar quitar un patrocinador.

Ruta del archivo: resources/views/patrocinador/quitarPatrocinador.blade.php

```
@extends('layouts.app')

@section('content')
    <div class="container">
        <div class="row mb-2">
            <h2>Quitar patrocinador de un evento</h2>
        </div>
        <div class="row g-5">
            <div class="col-sm-12 col-md-8" id="tablaEventos">
                <!-- Tabla de eventos -->
                <table class="table table-responsive table-striped text-secondary table-hover cursor" id="tablaEvento">
                    <caption>Eventos</caption>
                    <thead>
                        <tr>
                            <!-- Columnas de la tabla de eventos -->
                            <th scope="col" class="col-sm-4 col-md-4">Nombre del evento</th>
                            <th scope="col" class="col-sm-3 col-md-3">Tipo de evento</th>
                            <th scope="col" class="col-sm-3 col-md-3 text-center">Fecha de creación</th>
                            <th scope="col" class="col-sm-2 col-md-2 text-center font-sm">Cantidad de patrocinadores</th>
                        </tr>
                    </thead>
                    <tbody id="datosTabla">
                        <!-- Filas de la tabla de eventos, generadas dinámicamente desde el servidor -->
                        @foreach ($eventos as $evento)
                            @if (strtotime($evento->fin_evento) >= time())
                                <tr onclick="seleccionarEvento({{ $evento->id }}, '{{ $evento->nombre }}', event)"
                                    id="{{ $evento->id }}">
                                    <td>{{ $evento->nombre }}</td>
                                    <td>{{ $evento->tipoEvento->nombre }}</td>
                                    <td class="text-center">{{ date('d-m-Y', strtotime($evento->created_at)) }}</td>
                                    <td class="text-center" id="contadorPatrocinadores{{ $evento->id }}">
                                        {{ $evento->eventoPatrocinador->count() }}</td>
                                </tr>
                            @endif
                        @endforeach
                    </tbody>
                </table>
            </div>
            <div class="col-sm-12 col-md-4">
                <!-- Contenedor de la información sobre el evento seleccionado y patrocinadores -->
                <div class="container d-flex flex-column border p-3">
                    <div class="col-12 d-flex justify-content-center align-items-center">
                        <h4>Quitar patrocinador de:</h4>
                    </div>
                    <h5 id="nombreEvento" class="text-center fw-bold"></h5>
                    <div class="col-md-12">
                        <div class="row row-cols-2 g-3 mt-2 pb-2" style="overflow-y: auto; overflow-x: hidden"
                            id="divPatrocinadores">
                            <!-- Aca se muestran los patrocinadores de un evento desde el js -->
                        </div>
                    </div>
                </div>
                <!-- Componente de modal para confirmar la eliminación de un patrocinador -->
                @component('components.modal')
                    @slot('modalId', 'modalQuitarPatrocinador')
                    @slot('modalTitle', 'Confirmación')
                    @slot('modalContent')
                        ¿Está seguro de quitar este patrocinador?
                    @endslot
                    @slot('modalButton')
```

```
                    <!-- Botones dentro del modal para confirmar o cancelar la eliminación -->
                    <button type="button" class="btn btn-secondary w-25 mx-8" data-bs-dismiss="modal">No</button>
                    <button type="reset" class="btn btn-danger w-25 mx-8" data-bs-dismiss="modal"
                        onclick="quitarPatrocinador()">Sí</button>
                @endslot
            @endcomponent
        </div>
    </div>

    <!-- Enlaces a las hojas de estilo y scripts necesarios para DataTable y funcionalidades de la página -->
    <link href="https://cdn.datatables.net/1.13.6/css/dataTables.bootstrap5.min.css" rel="stylesheet">
    <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.7.1/jquery.min.js"></script>
    <script src="https://cdn.datatables.net/1.13.6/js/jquery.dataTables.min.js"></script>
    <script src="https://cdn.datatables.net/1.13.6/js/dataTables.bootstrap5.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/moment.js/2.29.2/moment.min.js"></script>
    <script src="{{ asset('js/Patrocinador/quitarPatrocinador.js') }}" defer></script>
@endsection
```

7.5.5.3. Interfaz de usuario de quitar patrocinador

La interfaz de usuario diseñada para la funcionalidad "Quitar Patrocinador de un Evento" ofrece una experiencia eficiente y clara para gestionar estas desvinculaciones. A continuación, se detallan los elementos y características clave de la interfaz:

Tabla de Eventos:

Nombre del Evento: Identifica de manera destacada el nombre de cada evento en la tabla.

Tipo de Evento: Muestra el tipo de evento, proporcionando una referencia rápida sobre la categoría del evento.

Fecha de Creación: Indica la fecha de creación del evento para contextualizar temporalmente la información.

Cantidad de Patrocinadores: Presenta el número actual de patrocinadores asociados a cada evento, ofreciendo una perspectiva del nivel de participación.

Acciones:

Selección de Evento: Al hacer clic en un evento específico, este se resalta, y se actualiza el panel lateral con los detalles del evento seleccionado.

Botones de Quitado: Cada patrocinador asignado tiene un botón "Quitar" que, al hacer clic, muestra un modal de confirmacion con los botones de “No” y “Si” al hacer click en “Si” inicia el proceso de desvinculación del evento seleccionado. Este botón se desactiva temporalmente durante el proceso para evitar operaciones duplicadas.

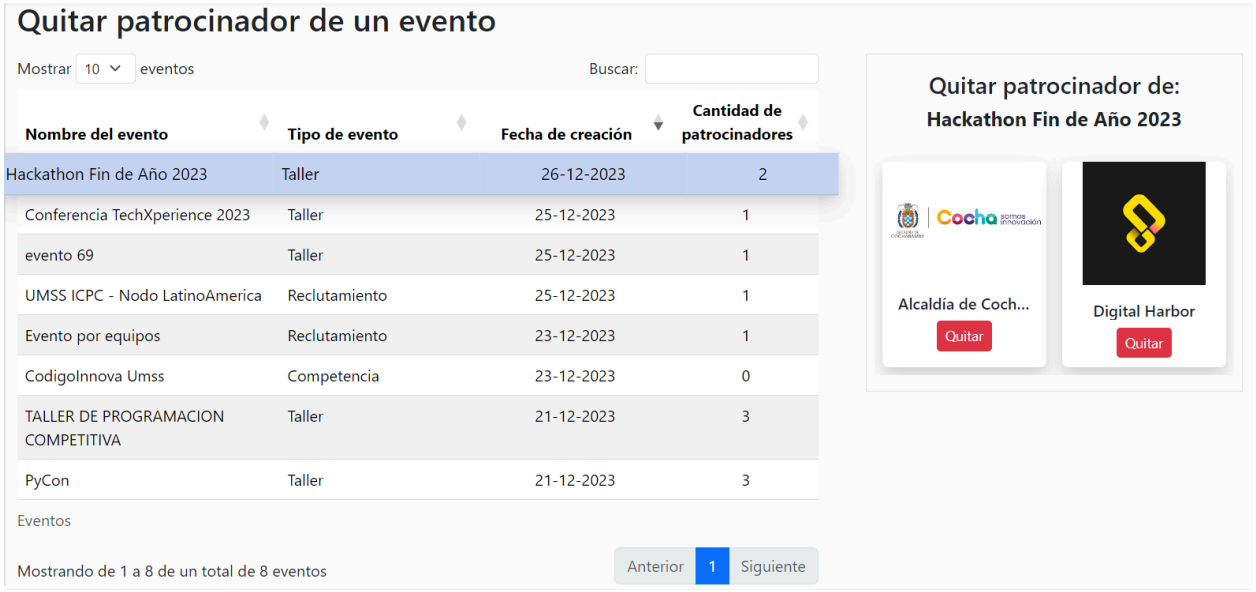
Panel Lateral:

Nombre del Evento Seleccionado: Muestra el nombre del evento seleccionado, proporcionando claridad sobre la acción que se está realizando.

Lista de Patrocinadores Asignados: Presenta una cuadrícula con imágenes y nombres de patrocinadores asignados al evento, permitiendo una desvinculación rápida y sencilla.

Diseño y Usabilidad:

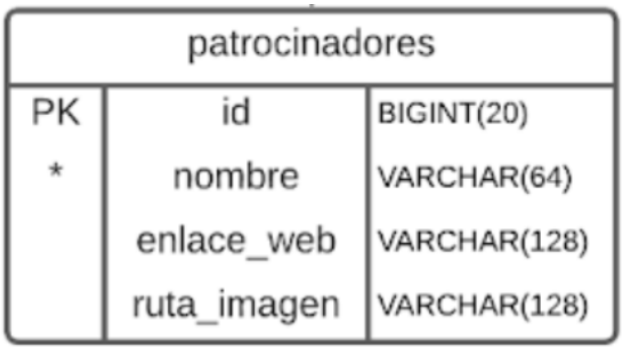
Interfaz Intuitiva: La disposición clara de la información facilita la desvinculación eficiente de patrocinadores de eventos.
Confirmación Visual: El sistema utiliza mensajes y desactivación temporal de botones para proporcionar una confirmación visual del estado de la operación.
Eficiencia Operativa: La interfaz está diseñada para facilitar la desvinculación eficiente de patrocinadores, mejorando la experiencia del usuario durante la gestión de asociaciones.



7.5.6. Tabla de base de datos de patrocinador

La tabla 'patrocinadores' en nuestra base de datos almacena información clave sobre los colaboradores y patrocinadores asociados a nuestros eventos. Cada patrocinador se identifica de manera única mediante 'id' y cuenta con detalles como 'nombre' para la identificación, 'enlace_web' para acceder a su pagina en línea, y 'ruta_imagen' que indica la ubicación del logotipo o imagen representativa del patrocinador. Esta estructura

proporciona una gestión eficiente de la colaboración con patrocinadores, permitiendo fácil acceso a información crucial para su participación en los eventos.



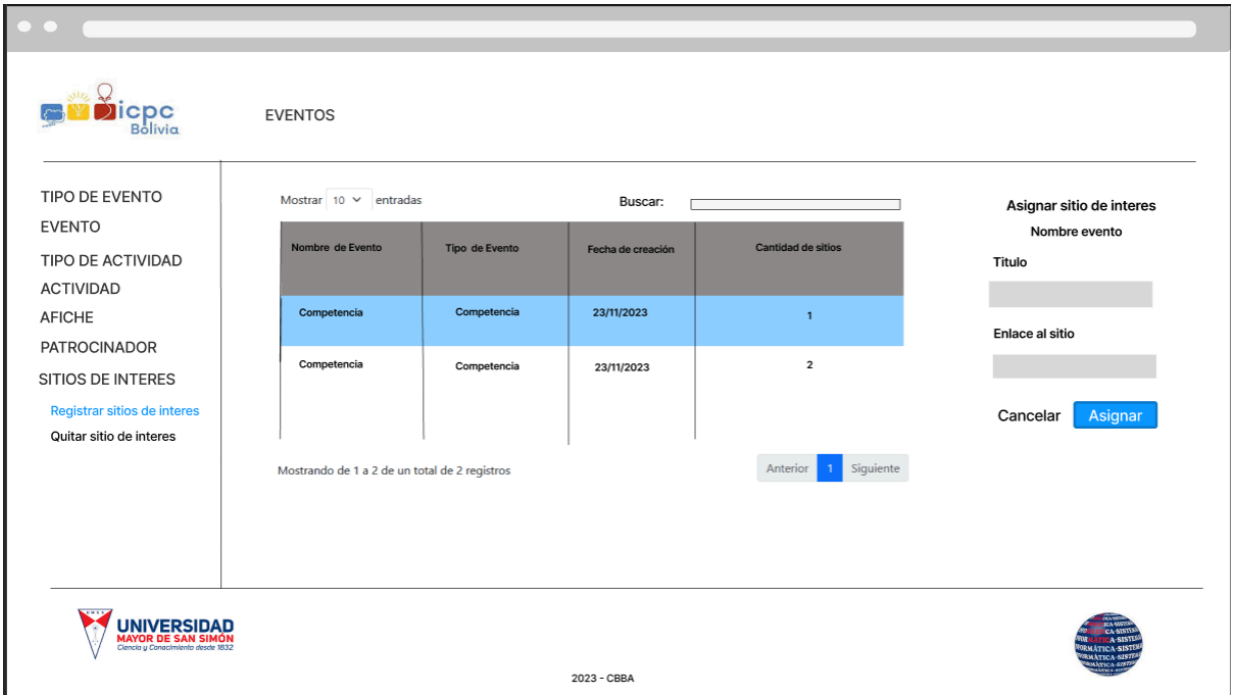
7.6. SITIOS DE INTERÉS

7.6.1. Registrar sitio de interés

7.6.1.1. Historia de usuario de registrar sitio de interés

Historia de Usuario	
Título: Registrar sitio de interés	**ID: 14**
Estimación: 5	**Importancia: Media**
Descripción: Como usuario con privilegio para administrar un evento, quiero registrar sitios de interés relacionados al evento con el fin de proporcionar información adicional que sea accesible para los participantes y asistentes del evento.	

Mockups



Criterios de aceptación.

1. Al hacer clic en la sección “SITIOS DE INTERÉS” en el menú lateral y seleccionar la opción “Registrar sitio de interés” se muestra una tabla con los eventos creados previamente.
2. La tabla tiene las columnas “Nombre del evento”, “Tipo de evento”, “Fecha de creación ”, “Cantidad de sitios”.
3. Al seleccionar un evento de la tabla, se habilita el formulario para subir un sitio de interés específico para ese evento.
4. El formulario contiene los siguientes campos obligatorios: "Título *" y "Enlace al sitio * ".
5. El campo "Título *" tiene un límite máximo de 64 caracteres.
6. El campo "Enlace al sitio *" tiene un límite máximo de 128 caracteres.
7. El campo “Enlace al sitio *” requiere que se tenga el prefijo “http://” o “https://”
8. El formulario tiene un botón "Asignar" para confirmar la asignación del sitio de interés.
9. El formulario tiene un botón "Cancelar" que restablece los campos del formulario
10. Si los campos cumplen con lo requerido, al hacer clic al botón "Asignar" se asigna el enlace de interés al evento seleccionado.
11. Si el campo “Título” está vacío, al hacer clic en el botón “Asignar” aparece un mensaje de validación “El título no puede estar vacío.” debajo del campo mencionado.
12. Si el campo “Enlace al sitio *” está vacío, al hacer clic en el botón “Asignar” aparece un mensaje de validación “El enlace al sitio no puede estar vacío.” debajo del campo mencionado.

13. Al asignar un enlace de interés a un evento, se muestra una alerta con el mensaje temporal "Sitio asignado correctamente".

7.6.1.2. Código fuente de registrar sitio de interés

Controlador de registrar sitio de interés

La función store en el controlador SitioController se encarga de almacenar un nuevo sitio de interés en la base de datos. Este método es invocado cuando se realiza una solicitud para agregar un sitio de interés asociado a un evento. Dentro de la función, se crea una instancia del modelo Sitio, se asignan los valores provenientes del objeto Request a los atributos correspondientes del nuevo sitio, y finalmente, se guarda el sitio en la base de datos. En caso de éxito, se retorna una respuesta JSON indicando que el sitio se ha asignado correctamente. Por otro lado, si se produce algún error durante el proceso, se captura la excepción del tipo QueryException y se retorna el mensaje de error asociado. Este manejo de excepciones garantiza una gestión adecuada de posibles fallos durante la inserción del sitio de interés en la base de datos.

Ruta del archivo: app/Http/Controllers/SitioController.php

```php
/**
 * Almacena un nuevo sitio de interés en la base de datos.
 */
public function store(Request $request)
{
    try {
        // Crea una nueva instancia de Sitio
        $sitio = new Sitio();

        // Asigna los valores del request a los atributos del sitio
        $sitio->titulo = $request->input('titulo');
        $sitio->enlace = $request->input('enlace');
        $sitio->id_evento = $request->input('id_evento');

        // Guarda el sitio en la base de datos
        $sitio->save();

        // Retorna una respuesta JSON indicando éxito
        return response()->json(['mensaje' => 'Sitio asignado correctamente', 'error' => false]);
    } catch (QueryException $e) {
        // En caso de error, retorna el mensaje de error
        return $e->getMessage();
    }
}
```

Código JavaScript de registrar sitio de interés

La funcionalidad principal se basa en la gestión de una tabla de eventos, donde se emplea DataTables para una visualización clara y eficiente. Al cargar la página, se inicializa la tabla con opciones específicas, y se deshabilitan los campos de entrada y botones relacionados hasta que un evento se seleccione. La función seleccionarEvento se encarga de resaltar visualmente el evento seleccionado, habilitando simultáneamente los campos para ingresar información sobre el sitio de interés y el botón de asignación. La validación de datos se ejecuta antes de realizar la asignación, utilizando el objeto FormData para

recopilar la información del formulario. Posteriormente, se realiza una solicitud HTTP POST a través de Axios para asignar el sitio al evento seleccionado. La función sumarContador se encarga de actualizar dinámicamente el contador de sitios asociados al evento, reflejando la asignación reciente. Finalmente, la función resetInputs restablece los campos del formulario después de una asignación exitosa. Los comentarios en el código proporcionan una guía detallada sobre cada sección y función, facilitando la comprensión y mantenimiento del código.

Ruta del archivo: public/js/SitiosInteres/subirSitio.js

```
// Variables globales para el control de la tabla y su inicialización
let tablaDeTipos;
let tablaInicializada = false;
// Elementos del DOM
const tituloSitio = document.getElementById("tituloSitio");
const urlSitio = document.getElementById("urlSitio");
const cancelar = document.getElementById("asignarSitioCancelar");
const asignar = document.getElementById("asignarSitioAsignar");
const eventoSeleccionado = document.getElementById("nombreEvento");
// Opciones de configuración para la tabla de eventos
const dataTableOptions = {
    pageLength: 10,
    lengthMenu: [5, 10, 15, 20],
    destroy: true,
    order: [[2, 'desc']],
    language: {
        lengthMenu: "Mostrar _MENU_ entradas",
        zeroRecords: "Ningún evento encontrado",
        info: "Mostrando de _START_ a _END_ de un total de _TOTAL_ registros",
        infoEmpty: "Ningún evento encontrado",
        infoFiltered: "(filtrados desde _MAX_ registros totales)",
        search: "Buscar:",
        loadingRecords: "Cargando...",
        paginate: {
            first: "Primero",
            last: "Último",
            next: "Siguiente",
            previous: "Anterior",
        },
    },
};
// Función para inicializar la tabla de eventos
const initDataTable = async () => {
    if (tablaInicializada) {
        tablaDeTipos.destroy();
    }
    DataTable.datetime('DD-MM-YYYY');
    tablaDeTipos = $("#tablaEvento").DataTable(dataTableOptions);
    tablaInicializada = true;
};
```

```
// Manejador de eventos cuando la ventana se ha cargado
window.addEventListener("load", () => {
    initDataTable();
    if (!seleccionado) {
        eventoSeleccionado.textContent = "Selecciona un evento";
        tituloSitio.disabled = true;
        urlSitio.disabled = true;
        cancelar.disabled = true;
        asignar.disabled = true;
    }
});

// Variables para el evento seleccionado y su identificador
let seleccionado;
let idSeleccionado;

// Función para seleccionar un evento y habilitar campos relacionados
const seleccionarEvento = (id, nombre) => {
    if (seleccionado) {
        seleccionado.classList.remove("table-primary");
    }
    tituloSitio.disabled = false;
    urlSitio.disabled = false;
    cancelar.disabled = false;
    asignar.disabled = false;
    seleccionado = document.getElementById(id);
    seleccionado.classList.add("table-primary");
    idSeleccionado = id;
    eventoSeleccionado.textContent = nombre;
};

// Función para validar los datos del formulario antes de realizar la asignación del sitio
const validarDatos = () => {
    const form = document.getElementById("formularioAgregarSitio");
    if (form.checkValidity()) {
        form.classList.remove("was-validated");
        crearFormData(form);
    }
    form.classList.add("was-validated");
};
```

```
// Función para crear FormData y realizar la asignación del sitio
const crearFormData = (form) => {
    const formData = new FormData(form);
    formData.append("id_evento", idSeleccionado);
    axios
        .post("/api/sitio", formData)
        .then((response) => {
            mostrarAlerta(
                "Éxito",
                response.data.mensaje,
                response.error ? "danger" : "success"
            );
            resetInputs();
            sumarContador();
        })
        .catch((err) => {
            console.error(err);
        });
};

// Función para actualizar el contador de sitios asociados al evento
const sumarContador = async () => {
    const contadorSitios = document.getElementById(
        `contadorSitios${idSeleccionado}`
    );
    contadorSitios.textContent = parseInt(contadorSitios.textContent) + 1;
};

// Función para restablecer los campos del formulario
const resetInputs = () => {
    let form = document.getElementById("formularioAgregarSitio");
    form.classList.remove("was-validated");
    tituloSitio.value = "";
    urlSitio.value = "";
};
```

Vista de registrar sitio de interés

La vista consta de dos columnas principales. La primera columna presenta una tabla que muestra eventos disponibles con detalles como el nombre, tipo, fecha de creación y la cantidad de sitios de interés asociados. Al hacer clic en un evento, se resalta y se actualiza un formulario en la segunda columna para asignar un nuevo sitio de interés a ese evento específico. La segunda columna contiene un formulario que incluye campos para el título y el enlace del nuevo sitio de interés. Además, muestra el nombre del evento seleccionado para mayor claridad. La interfaz está diseñada con validaciones de formulario y feedback visual para mejorar la experiencia del usuario. También utiliza DataTables para una presentación ordenada y paginada de los eventos. El script subirSitio.js se encarga de las interacciones y validaciones en la página.

Ruta del archivo: resources/views/sitiosInteres/subirSitio.blade.php

```
@extends('layouts.app')

@section('content')
    <div class="container">
        <div class="row mb-2">
            <h2>Registrar sitio de interés en un evento</h2>
        </div>
        <div class="row g-5">
            <!-- Columna de la tabla de eventos -->
            <div class="col-sm-12 col-md-8">
                <!-- Tabla que muestra los eventos disponibles -->
                <table class="table table-responsive table-striped text-secondary table-hover cursor" id="tablaEvento">
                    <caption>Eventos</caption>
                    <thead>
                        <!-- Encabezados de la tabla -->
                        <tr>
                            <th scope="col" class="col-sm-4 col-md-4">Nombre del evento</th>
                            <th scope="col" class="col-sm-0 col-md-3 text-center">Tipo de evento</th>
                            <th scope="col" class="col-sm-3 col-md-2 text-center">Fecha de creación</th>
                            <th scope="col" class="col-sm-3 col-md-2 text-center font-sm">Cantidad de sitios</th>
                        </tr>
                    </thead>
                    <tbody id="datosTabla">
                        <!-- Bucle que recorre los eventos y los presenta en la tabla -->
                        @foreach ($eventos as $evento)
                            @if (strtotime($evento->fin_evento) >= time())
                                <tr onclick="seleccionarEvento({{ $evento->id }}, '{{ $evento->nombre }}', event)"
                                    id="{{ $evento->id }}">
                                    <td>{{ $evento->nombre }}</td>
                                    <td class="text-center">{{ $evento->tipoEvento->nombre }}</td>
                                    <td class="text-center">{{ date('d-m-Y', strtotime($evento->created_at)) }}</td>
                                    <td class="text-center" id="contadorSitios{{ $evento->id }}">
                                        {{ $evento->sitios->count() }}</td>
                                </tr>
                            @endif
                        @endforeach
                    </tbody>
                </table>
            </div>
            <!-- Columna del formulario de asignación de sitio -->
            <div class="col-sm-12 col-md-4">
                <!-- Contenedor del formulario y detalles del evento seleccionado -->
                <div class="container d-flex flex-column justify-content-center align-items-center border p-3">
                    <!-- Título de la sección de asignación de sitio -->
                    <div class="col-12 d-flex justify-content-center align-items-center">
                        <h4>Asignar sitio de interés</h4>
                    </div>
                    <!-- Nombre del evento seleccionado -->
                    <h5 id="nombreEvento" class="text-center fw-bold"></h5>
                    <!-- Formulario para ingresar detalles del sitio de interés -->
                    <form class="needs-validation" novalidate id="formularioAgregarSitio">
                        <div class="col-md-12 mt-2">
                            <label for="tituloSitio" class="form-label">Título *</label>
                            <!-- Campo para ingresar el título del sitio de interés -->
                            <input name="titulo" type="text" class="form-control custom-input" id="tituloSitio"
                                value="" placeholder="Ingrese un título" maxlength="64" required>
                            <div class="invalid-feedback">
                                El título no puede estar vacío.
                            </div>
                        </div>
```

```
                        <div class="col-md-12 mt-2">
                            <label for="urlSitio" class="form-label">Enlace al sitio *</label>
                            <!-- Campo para ingresar el enlace al sitio de interés -->
                            <input name="enlace" type="url" class="form-control custom-input" id="urlSitio"
                                value="" pattern="https?://.+" placeholder="https://www.ejemplo.com" maxlength="128"
                                required>
                            <div class="invalid-feedback">
                                El enlace al sitio no puede estar vacío.
                            </div>
                        </div>
                    </form>
                    <!-- Botones para cancelar la asignación o confirmarla -->
                    <div class="d-flex justify-content-center mt-3 gap-3">
                        <button type="button" class="btn btn-light" onclick="resetInputs()"
                            id="asignarSitioCancelar">Cancelar</button>
                        <button type="button" class="btn btn-primary" onclick="validarDatos()"
                            id="asignarSitioAsignar">Asignar</button>
                    </div>
                </div>
            </div>
        </div>
    </div>
    <!-- Enlaces a las bibliotecas y scripts necesarios -->
    <link href="https://cdn.datatables.net/1.13.6/css/dataTables.bootstrap5.min.css" rel="stylesheet">
    <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.7.1/jquery.min.js"></script>
    <script src="https://cdn.datatables.net/1.13.6/js/jquery.dataTables.min.js"></script>
    <script src="https://cdn.datatables.net/1.13.6/js/dataTables.bootstrap5.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/moment.js/2.29.2/moment.min.js"></script>
    <!-- Script personalizado para la funcionalidad de la página -->
    <script src="{{ asset('js/SitiosInteres/subirSitio.js') }}" defer></script>
@endsection
```

7.6.1.3. Interfaz de usuario de registrar sitio de interés

La interfaz de usuario destinada al registro de sitios de interés en eventos proporciona una experiencia intuitiva y eficiente para asignar nuevos sitios a eventos específicos. A continuación se describen los elementos clave de la interfaz:

Tabla de Eventos:

Nombre del Evento: Identifica de manera distintiva el nombre de cada evento en la tabla.

Tipo de Evento: Muestra el tipo de evento para una rápida referencia.

Fecha de Creación: Indica la fecha en que se creó el evento.

Cantidad de Sitios: Muestra el número actual de sitios de interés asociados a cada evento, proporcionando una visión instantánea de su participación.

Acciones:

Selección de Evento: Al hacer clic en un evento específico, se resalta, y se activa el formulario en la columna derecha para asignar un nuevo sitio de interés.

Formulario para Registro:

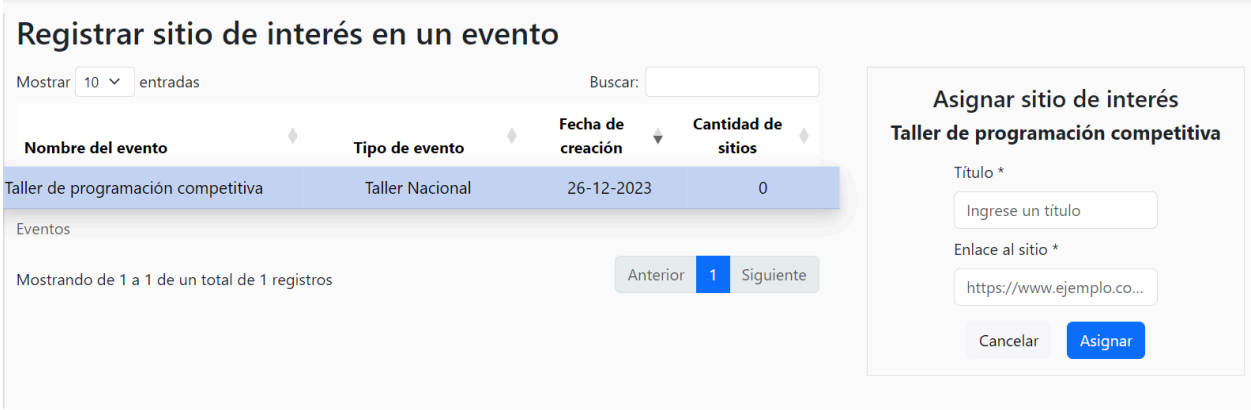
Título del Sitio: Permite ingresar un título para el nuevo sitio de interés.

Enlace al Sitio: Solicita el enlace al sitio de interés con validación de formato.

Botones de Acción: Incluye botones para cancelar la operación y asignar el sitio, con validaciones visuales y desactivación temporal para evitar operaciones duplicadas.

Diseño y Usabilidad:

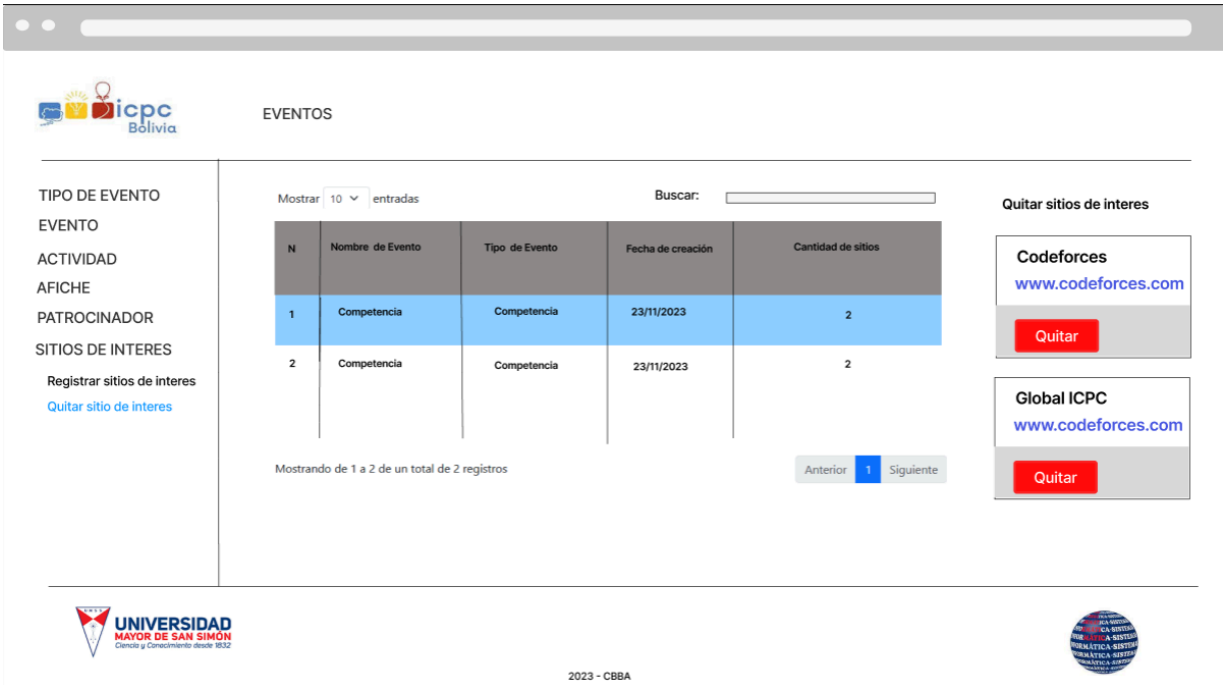
Interfaz Intuitiva: La disposición clara de eventos y el formulario facilitan el proceso de registro de sitios.
Validaciones Visuales: Utiliza colores y mensajes de validación para proporcionar una confirmación visual del estado de los datos ingresados.
Eficiencia Operativa: La interfaz está diseñada para permitir la asignación eficiente de sitios de interés, mejorando la experiencia general del usuario.



7.6.2. Quitar sitio de interés

7.6.2.1. Historia de usuario de quitar sitio de interés

Historia de Usuario	
Título: Quitar sitios de interés	**ID: 21**
Estimación: 3	**Importancia: Baja**
Descripción Como usuario con privilegio para quitar sitios de interés, deseo tener la capacidad de quitar sitios de interés asociados a un evento específico, para mantener la información actualizada para que los usuarios solo vean enlaces relevantes y útiles.	
Mockup	



Criterios de Aceptación

1. Al hacer clic en la sección “SITIOS DE INTERÉS” en el menú lateral y seleccionar la opción “Quitar sitio de interés” se muestra la tabla con los eventos existentes.
2. La tabla tiene las siguientes columnas: “Nombre del evento”, “Tipo de evento”, “Fecha de creación”, “Cantidad de sitios”.
3. En la columna “Cantidad de sitios” de la tabla, se muestran la cantidad de sitios que fueron asignados a cada uno de los eventos existentes.
4. Al hacer clic sobre un evento de la tabla, se muestran los sitios de interés asignados al evento.
5. Cada sitio de interés se muestra con el nombre, enlace y el botón de “Quitar” en la parte inferior.
6. Al hacer clic en el botón “Quitar”, se abre un modal con el mensaje “¿Está seguro de quitar este sitio de interés?”
7. Al hacer clic en el botón “Sí” del modal, se muestra una alerta temporal con el mensaje “Quitado exitosamente”.
8. Al hacer clic en el botón “Sí” del modal, el sitio de interés deja de ser visible.
9. Al hacer clic en el botón “Si” del modal, se cierra el modal.
10. Al hacer clic en el botón “Si” del modal, se reduce en uno el valor de la columna “Cantidad de sitios” para el evento seleccionado.
11. Al hacer clic en el botón “No” del modal, se cierra el modal.

7.6.2.2. Código fuente de quitar sitio de interés

Controlador de quitar sitio de interés

La función destroy en el controlador SitioController de Laravel se encarga de eliminar un sitio de interés asociado a un evento. En primer lugar, se intenta localizar el sitio de interés mediante su ID utilizando el modelo Sitio. Si el sitio existe, se procede a eliminarlo, y se retorna una respuesta JSON con un mensaje indicando el éxito de la operación y un indicador de error establecido en false. En caso de que el sitio no sea encontrado, se devuelve un mensaje de error y el indicador de error se establece en true. Además, la función maneja excepciones de consulta, capturando cualquier error de base de datos y proporcionando un mensaje de error correspondiente en la respuesta JSON. Este enfoque asegura que la operación de eliminación sea manejada de manera segura y proporciona información clara sobre el resultado de la acción.

Ruta del archivo: app/Http/Controllers/SitioController.php

```php
/**
 * Elimina un sitio de interés asociado a un evento.
 */
public function destroy($id)
{
    try {
        // Busca el sitio de interés por su ID.
        $sitio = Sitio::find($id);

        // Verifica si el sitio existe antes de intentar eliminarlo.
        if ($sitio) {
            // Elimina el sitio de interés.
            $sitio->delete();

            // Devuelve una respuesta JSON indicando éxito y un mensaje.
            return response()->json(['mensaje' => 'Sitio quitado exitosamente', 'error' => false]);
        } else {
            // Si el sitio no se encuentra, devuelve un mensaje de error.
            return response()->json(['mensaje' => 'Sitio no encontrado', 'error' => true]);
        }
    } catch (QueryException $e) {
        // Captura cualquier excepción de consulta y devuelve un mensaje de error.
        return response()->json(['mensaje' => $e->getMessage(), 'error' => true]);
    }
}
```

Código JavaScript de quitar sitio de interés

Comienza declarando variables y configurando opciones para DataTable. Luego, inicializa la DataTable al cargar la página, estableciendo eventos y condiciones iniciales. La función seleccionarEvento se activa al hacer clic en un evento, resalta la selección y carga los sitios de interés asociados. La función cargarSitios utiliza una solicitud AJAX para obtener los sitios de interés asociados a un evento específico. Después, la función mostrarSitios crea dinámicamente tarjetas para cada sitio de interés y las agrega al contenedor correspondiente en la interfaz. El código también incluye funciones para seleccionar y quitar sitios de interés, utilizando modales de confirmación para evitar acciones accidentales. Además, se implementa un observador de redimensionamiento para ajustar la altura del contenedor de sitios en función de la tabla de eventos.

Ruta del archivo: public/js/SitiosInteres/quitarSitio.js

```
// Declaración de variables globales
let tablaDeTipos;
let tablaInicializada = false;
const eventoSeleccionado = document.getElementById("nombreEvento");
const sitiosContenedor = document.getElementById("sitiosContenedor");

// Opciones para la inicialización de la DataTable
const dataTableOptions = {
    // Configuración de la DataTable
};

// Función para inicializar la DataTable
const initDataTable = async () => {
    if (tablaInicializada) {
        tablaDeTipos.destroy();
    }
    DataTable.datetime("DD-MM-YYYY");
    tablaDeTipos = $("#tablaEvento").DataTable(dataTableOptions);
    tablaInicializada = true;
};

// Evento al cargar la página
window.addEventListener("load", () => {
    initDataTable();
    if (!seleccionado) {
        eventoSeleccionado.textContent = "Selecciona un evento";
    }
});

// Variables para seguimiento del evento seleccionado y su ID
let seleccionado;
let idSeleccionado;

// Función para seleccionar un evento y cargar sus sitios de interés
const seleccionarEvento = async (id, nombre) => {
    if (seleccionado) {
        seleccionado.classList.remove("table-primary");
        sitiosContenedor.innerHTML = "";
    }
```

```
    seleccionado = document.getElementById(id);
    seleccionado.classList.add("table-primary");
    idSeleccionado = id;
    eventoSeleccionado.textContent = nombre;
    const response = await cargarSitios(id);
    mostrarSitios(response);
};

// Función para cargar los sitios de interés de un evento
const cargarSitios = async (id) => {
    let data = await axios.get("/api/sitio/" + id).then((response) => {
        return response.data;
    });
    return data;
};

// Función para mostrar los sitios de interés en la interfaz
const mostrarSitios = (response) => {
    response.map((sitio) => {
        // Crear contenedor para cada sitio de interés
        const sitioContenedor = document.createElement("div");
        sitioContenedor.id = "sitio" + sitio.id;
        // Agregar contenido HTML al contenedor
        sitioContenedor.innerHTML = `
            <!-- Estructura de tarjeta para mostrar un sitio de interés -->
        `;
        // Agregar el contenedor al contenedor principal
        sitiosContenedor.appendChild(sitioContenedor);
    });
};

// Variable para el sitio de interés seleccionado
let sitioSeleccionado;

// Función para seleccionar un sitio de interés
const seleccionarSitio = (id) => {
    sitioSeleccionado = id;
};
```

```
// Función para eliminar un sitio de interés
const quitarSitio = async () => {
    if (sitioSeleccionado) {
        // Realizar petición para eliminar el sitio de interés
        const response = await axios
            .delete("/api/sitio/" + sitioSeleccionado)
            .then((response) => {
                // Remover el contenedor del sitio eliminado
                document.getElementById("sitio" + sitioSeleccionado).remove();
                // Mostrar mensaje de éxito o error
                mostrarAlerta(
                    "Éxito",
                    response.data.mensaje,
                    response.error ? "danger" : "success"
                );
                // Actualizar la interfaz con la respuesta
                restarContador();
                return response.data;
            });
        // Mostrar los sitios de interés actualizados
        mostrarSitios(response);
    }
};

// Función para actualizar el contador de sitios de interés
const restarContador = async () => {
    const contadorSitios = document.getElementById(
        `contadorSitios${idSeleccionado}`
    );
    contadorSitios.textContent = parseInt(contadorSitios.textContent) - 1;
};

// Función para resetear los inputs del formulario (no se proporciona el código del formulario)
const resetInputs = () => {
    let form = document.getElementById("formularioAgregarSitio");
    form.classList.remove("was-validated");
    tituloSitio.value = "";
    urlSitio.value = "";
};

// Observador de redimensionamiento para ajustar la altura del contenedor de sitios
const resize_ob = new ResizeObserver(function (entries) {
    let rect = entries[0].contentRect;
    let height = rect.height;
    vh = parseInt((height / window.innerHeight) * 78) + 1;
    document.getElementById("sitiosContenedor").style.height = vh + "vh";
});

// Observar cambios en el tamaño del contenedor de la DataTable
resize_ob.observe(document.getElementById("tablaEvento"));
```

Vista de quitar sitio de interés

La interfaz se compone de dos secciones principales. La primera, representada por una tabla interactiva, exhibe información esencial sobre los eventos disponibles, como el nombre, tipo, fecha de creación y la cantidad de sitios de interés asociados. Al hacer clic en un evento específico, se resalta y se desencadena la carga dinámica de los sitios de interés asociados a través de JavaScript, mostrándolos en un contenedor adyacente. La segunda sección permite la acción de quitar sitios de interés de un evento seleccionado. Muestra el nombre del evento y presenta un contenedor que exhibe los sitios de interés

asociados, junto con la opción de confirmar la eliminación mediante un modal interactivo. El código utiliza bibliotecas externas como DataTable y jQuery para mejorar la presentación y funcionalidad de la tabla de eventos. Además, se ha implementado un script diferido para gestionar las interacciones en la interfaz relacionadas con la eliminación de sitios de interés.

Ruta del archivo: resources/views/sitiosInteres/quitarSitio.blade.php

```
@extends('layouts.app')

@section('content')
    <div class="container">
        <!-- Encabezado de la página -->
        <div class="row mb-2">
            <h2>Quitar sitio de interés de un evento</h2>
        </div>

        <div class="row g-5">
            <!-- Sección de la tabla de eventos -->
            <div class="col-sm-12 col-md-8">
                <table class="table table-responsive table-striped text-secondary table-hover cursor" id="tablaEvento">
                    <caption>Eventos</caption>
                    <thead>
                        <tr>
                            <th scope="col" class="col-sm-4 col-md-4">Nombre del evento</th>
                            <th scope="col" class="col-sm-0 col-md-3 text-center">Tipo de evento</th>
                            <th scope="col" class="col-sm-3 col-md-2 text-center">Fecha de creación</th>
                            <th scope="col" class="col-sm-3 col-md-2 text-center font-sm">Cantidad de sitios</th>
                        </tr>
                    </thead>
                    <tbody id="datosTabla">
                        <!-- Iteración sobre eventos para mostrar en la tabla -->
                        @foreach ($eventos as $evento)
                            @if (strtotime($evento->fin_evento) >= time())
                                <tr onclick="seleccionarEvento({{ $evento->id }}, '{{ $evento->nombre }}', event)"
                                    id="{{ $evento->id }}">
                                    <td>{{ $evento->nombre }}</td>
                                    <td class="text-center">{{ $evento->tipoEvento->nombre }}</td>
                                    <td class="text-center">{{ date('d-m-Y', strtotime($evento->created_at)) }}</td>
                                    <td class="text-center" id="contadorSitios{{ $evento->id }}">
                                        {{ $evento->sitios->count() }}</td>
                                </tr>
                            @endif
                        @endforeach
                    </tbody>
                </table>
            </div>

            <!-- Sección de quitar sitio de interés -->
            <div class="col-sm-12 col-md-4">
                <div class="container d-flex flex-column border p-3">
                    <!-- Encabezado de la sección de quitar sitio de interés -->
                    <div class="col-12 d-flex justify-content-center align-items-center">
                        <h4>Quitar sitio de interés</h4>
                    </div>

                    <!-- Nombre del evento seleccionado -->
                    <h5 id="nombreEvento" class="text-center fw-bold"></h5>

                    <!-- Contenedor para mostrar los sitios de interés -->
                    <div class="col-md-12">
                        <div class="row row-cols-1 g-3 mt-2" style="height:48vh; overflow-y: auto;" id="sitiosContenedor">
                            <!-- Aquí se muestran los sitios de interés del evento mediante el JavaScript -->
                        </div>
                    </div>
                </div>
```

```
                <!-- Modal de confirmación para quitar sitio de interés -->
                @component('components.modal')
                    @slot('modalId', 'modalQuitarSitio')
                    @slot('modalTitle', 'Confirmación')
                    @slot('modalContent')
                        ¿Está seguro de quitar este sitio de interés?
                    @endslot
                    @slot('modalButton')
                        <button type="button" class="btn btn-secondary w-25 mx-8" data-bs-dismiss="modal">No</button>
                        <button type="reset" class="btn btn-danger w-25 mx-8" data-bs-dismiss="modal"
                            onclick="quitarSitio()">Sí</button>
                    @endslot
                @endcomponent
            </div>
        </div>
    </div>

    <!-- Enlaces a bibliotecas y scripts externos -->
    <link href="https://cdn.datatables.net/1.13.6/css/dataTables.bootstrap5.min.css" rel="stylesheet">
    <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.7.1/jquery.min.js"></script>
    <script src="https://cdn.datatables.net/1.13.6/js/jquery.dataTables.min.js"></script>
    <script src="https://cdn.datatables.net/1.13.6/js/dataTables.bootstrap5.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/moment.js/2.29.2/moment.min.js"></script>
    <script src="{{ asset('js/SitiosInteres/quitarSitio.js') }}" defer></script>
@endsection
```

7.6.2.3. Interfaz de usuario de quitar sitio de interés

La interfaz de usuario diseñada para quitar sitios de interés de un evento proporciona una experiencia intuitiva y eficiente en la gestión de estos elementos asociados. A continuación, se describen los componentes clave de esta interfaz:

Tabla de Eventos:

Nombre del Evento: Identifica de manera distintiva cada evento en la tabla.

Tipo de Evento: Muestra el tipo de evento para una rápida referencia.

Fecha de Creación: Indica la fecha en que se creó el evento, proporcionando contexto temporal.

Cantidad de Sitios: Muestra el número actual de sitios de interés asociados a cada evento.

Acciones:

Selección de Evento: Al hacer clic en un evento específico, este se resalta y se actualiza el panel lateral con los detalles del evento seleccionado.

Botón "Quitar": Al hacer clic en el botón "Quitar", se activa un modal de confirmación para asegurar la eliminación del sitio de interés.

Panel Lateral:

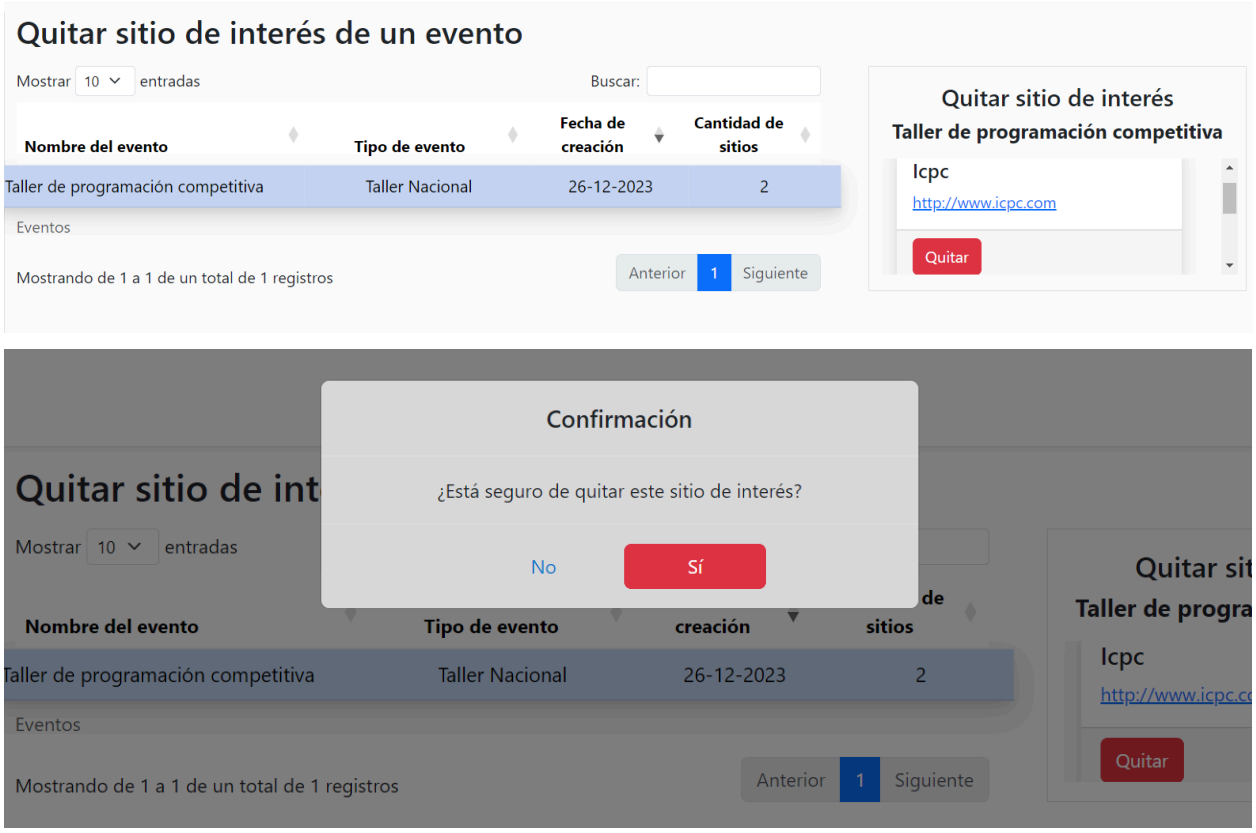
Nombre del Evento Seleccionado: Muestra el nombre del evento seleccionado, brindando claridad sobre la acción que se está realizando.

Lista de Sitios de Interés: Presenta una cuadrícula con información sobre los sitios de interés asociados al evento, ofreciendo detalles como el título, enlace y un botón para confirmar la eliminación.

Diseño y Usabilidad:

Interfaz Responsiva: El diseño se adapta eficazmente a diferentes tamaños de pantalla, garantizando una experiencia de usuario consistente.

Confirmación Modal: La interfaz utiliza un modal interactivo para confirmar la eliminación de un sitio, evitando acciones accidentales.
Eficiencia Operativa: La disposición clara de la información y la capacidad de eliminación directa desde la interfaz mejoran la eficiencia en la gestión de sitios de interés asociados a eventos.



7.6.3. Tabla de base de datos de sitios de interés

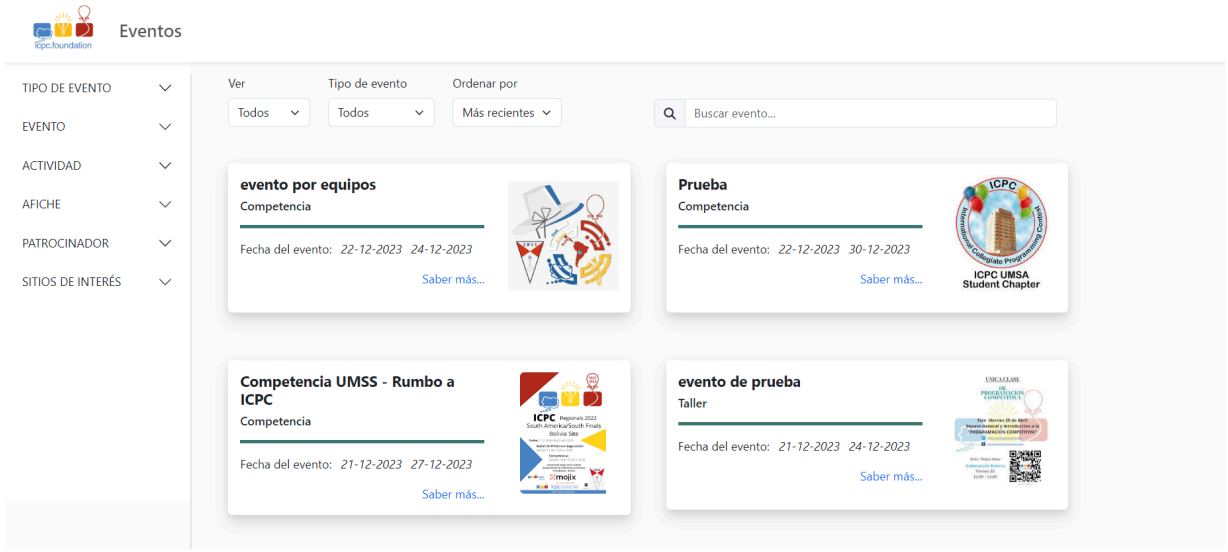
La tabla 'sitios' en nuestra base de datos gestiona información relacionada con sitios web asociados a nuestros eventos. Cada sitio se identifica de manera única mediante 'id' y se vincula a un evento específico a través de 'id_evento'. Los campos 'nombre' y 'enlace' proporcionan detalles descriptivos y la dirección web del sitio respectivamente. Esta estructura permite un acceso fácil y organizado a la información clave sobre los sitios asociados a cada evento en nuestro sistema.

sitios		
PK	id	BIGINT(20)
*	nombre	VARCHAR(255)
*	enlace	VARCHAR(255)
FK	id_evento	BIGINT(20)

7.7. Eventos

7.7.1. Mostrar eventos

7.7.1.1. Historia de usuarios de mostrar eventos

Historia de Usuario	
Título: Mostrar Eventos	**ID: 15**
Estimación: 13	**Importancia: Alta**
Descripción Yo como usuario, quiero ver todos los eventos registrados en el sistema, con la siguiente información: nombre del evento, tipo del evento, fecha de realización del evento, un afiche para tener constancia de la correcta creación del evento.	
Mockups 	
Criterios de Aceptación	

1. Al hacer clic en la sección “Eventos” que se encuentra en el encabezado de la página, se cargan en tarjetas los eventos creados previamente.
2. En cada tarjeta de evento se muestra el nombre, tipo de evento, fecha de inicio-fin del evento, un afiche.
3. Cada tarjeta de evento tiene un enlace llamado “Saber más” que al hacer clic nos dirige a la vista del evento seleccionado.
4. Se tiene un buscador que permite buscar los eventos según el nombre dado en la búsqueda.
5. Se tiene un filtro llamado “Ver” que filtra a través de las siguientes opciones: Todos, En curso, Futuros, Pasados.
6. Se tiene un filtro llamado “Tipo de evento” que muestra todos los tipos de evento creados previamente.
7. Al seleccionar un tipo de evento, se filtran los eventos con el tipo de evento seleccionado.
8. Se tiene un filtro “Ordenar por”, que tiene las opciones “A-Z”, “Z-A”, “Más recientes”, “Más antiguos”.

7.7.1.2. Código fuente de mostrar eventos

Controlador de mostrar eventos

La función show en este controlador está diseñada para recuperar y mostrar información detallada sobre un evento específico identificado por su ID único. Dentro de un bloque try-catch, la función intenta obtener el evento utilizando el modelo Evento de Laravel, junto con la relación tipoEvento. Esta relación se carga incluso si el tipo de evento asociado ha sido eliminado lógicamente (con la ayuda de withTrashed()). En caso de que el evento no se encuentre, se devuelve una respuesta JSON indicando que el evento no fue encontrado, junto con un código de estado HTTP 404. En el caso de que ocurra alguna excepción durante la ejecución, se captura en el bloque catch y se devuelve una respuesta de error con el mensaje de la excepción y un código de estado HTTP 500. Si la operación tiene éxito, la función retorna una respuesta JSON que contiene la información detallada del evento y un indicador de éxito, proporcionando así una interfaz consistente y manejo de errores para la recuperación de eventos en la aplicación.

Ruta del archivo: app/Http/Controllers/EventoController.php

```php
/**
 * Muestra la información detallada de un evento específico.
 */
public function show($id)
{
    try {
        // Recupera el evento con el tipo de evento asociado, incluso si el tipo de evento está
        //eliminado lógicamente.
        $evento = Evento::with(["tipoEvento" => function ($q) {
            $q->withTrashed();
        }])->find($id);

        // Verifica si se encontró el evento.
        if (!$evento) {
            // Si no se encuentra el evento, devuelve una respuesta de error 404.
            return response()->json(['mensaje' => 'Evento no encontrado', 'error' => true], 404);
        }

        // Devuelve la información detallada del evento.
        return response()->json(['evento' => $evento, 'error' => false]);

    } catch (\Exception $e) {
        // Si ocurre una excepción, devuelve una respuesta de error con el mensaje de la excepción.
        return response()->json(['mensaje' => $e->getMessage(), 'error' => true], 500);
    }
}
```

Código JavaScript de mostrar evento

Este código JavaScript tiene como objetivo gestionar la visualización y filtrado de eventos en una interfaz web. La funcionalidad principal se basa en la obtención de datos de eventos y tipos de eventos desde una API, así como la creación dinámica de elementos en la interfaz de usuario para mostrar tarjetas de eventos. En la carga inicial de la página, se llaman las funciones getEventos y getTiposEventos mediante eventos de carga (load). La primera función realiza una solicitud GET a la API de eventos, obteniendo la información y almacenándola en diversas variables para su posterior manipulación. La segunda función obtiene los tipos de eventos y los presenta en un elemento select del HTML. El código incluye funciones para buscar eventos, filtrarlos por diferentes criterios (como estado, tipo y orden) y mostrarlos en la interfaz. Se utilizan librerías como Axios para realizar solicitudes HTTP y Moment.js para manipular fechas de manera sencilla. Se destacan funciones específicas, como buscarEvento para la búsqueda dinámica según un criterio ingresado, filtrarVer y filtrarTipo para aplicar filtros de visualización, y mostrarEventos para generar el contenido HTML de las tarjetas de eventos en base a los datos obtenidos y los filtros aplicados.

Ruta del archivo: public/js/eventos.js

```
// Variables para almacenar información de eventos
let eventos;
let eventosFiltrados;
let eventosVer;
let eventosTipo;
let eventosOrden;
let eventosBuscar;

// Evento que se dispara cuando la ventana ha terminado de cargar
window.addEventListener("load", () => {
    // Se obtienen los eventos y tipos de eventos al cargar la página
    getEventos();
    getTiposEventos();
});

// Función asincrónica para obtener la lista completa de eventos
const getEventos = async () => {
    // Se realiza una solicitud GET a la API de eventos
    let datos = await axios.get("/api/evento").then((response) => {
        return response.data;
    });
    // Se almacenan los eventos en las variables correspondientes
    eventos = await datos;
    eventosFiltrados = eventosVer = eventosTipo = eventosOrden = eventosBuscar = [...eventos];
};

// Función para obtener la lista de tipos de eventos y llenar un select en el HTML
const getTiposEventos = async () => {
    axios.get("/api/tipo-evento")
    .then(function (response) {
        // Se obtienen los tipos de eventos y se llena el select en el HTML
        const select = document.getElementById("select_tipo_evento");
        const tiposDeEvento = response.data;
        tiposDeEvento.forEach(function (tipo) {
            const option = document.createElement("option");
            option.value = tipo.id;
            option.text = tipo.nombre;
            select.appendChild(option);
        })
    })
```

```
    .catch(function (error) {
        console.error(error);
    });
}

// Función para buscar eventos según un criterio ingresado en un campo de búsqueda
const buscarEvento = () => {
    let buscado = document.getElementById("buscadorDeEvento").value;
    eventosBuscar = eventos.filter(evento => {
        let datos = evento.nombre.toLowerCase();
        datos += " " + moment(evento.inicio_evento).format("DD-MM-YYYY");
        datos += " " + moment(evento.fin_evento).format("DD-MM-YYYY");
        datos += " " + evento.tipo_evento.nombre;
        return datos.includes(buscado.toLowerCase());
    });
    mostrarEventos();
};

// Función para filtrar eventos según la opción seleccionada en un select
const filtrarVer = () =>{
    let seleccionado = document.getElementById("select_ver");
    let valor = parseInt(seleccionado.value);
    switch (valor) {
        case 1:
            eventosVer = eventos;
            break;
        case 2:
            eventosVer = eventos.filter(evento =>
                ((moment()).isSame(moment(evento.inicio_evento)) ||(moment()).isAfter(moment(
                  evento.inicio_evento))) &&
                moment().isBefore(moment(evento.fin_evento))
            );
            break;
        case 3:
            eventosVer = eventos.filter(evento =>
                moment(evento.inicio_evento).isAfter(moment())
            );
            break;
        case 4:
            eventosVer = eventos.filter(evento =>
```

```
                    moment(evento.fin_evento).isBefore(moment())
                );
                break;
        }
        mostrarEventos();
}

// Función para filtrar eventos según el tipo seleccionado en un select
const filtrarTipo = () =>{
    let seleccionado = document.getElementById("select_tipo_evento");
    let tipo = seleccionado.options[seleccionado.selectedIndex].text;
    if(tipo !== "Todos"){
        eventosTipo = eventos.filter(evento =>{
            return evento.tipo_evento.nombre == tipo;
        })
    } else {
        eventosTipo = eventos;
    }
    mostrarEventos();
}

// Función para obtener la intersección de múltiples arreglos
const interseccionMultiple = (arreglos) =>{
    if (!arreglos || arreglos.length === 0) {
      return [];
    }

    // Tomar el primer arreglo como base
    const base = arreglos[0];

    // Aplicar la intersección sucesiva con los demás arreglos
    const intersection = arreglos.slice(1).reduce((result, arreglos) => {
      return result.filter(element => arreglos.includes(element));
    }, base);

    return intersection;
}
```

```
// Función para mostrar los eventos en la interfaz
const mostrarEventos = () => {
    let div = document.getElementById("tarjetasRow");
    let contenido = "";
    let seleccionado = document.getElementById("select_por");
    eventosFiltrados = interseccionMultiple([eventos, eventosVer, eventosTipo, eventosBuscar]);

    switch(seleccionado.value){
        case "1":
            eventosFiltrados.sort((a, b) => a.nombre.localeCompare(b.nombre));
            break;
        case "2":
            eventosFiltrados.sort((a, b) => b.nombre.localeCompare(a.nombre));
            break;
        case "3":
            eventosFiltrados.sort((a,b)=> new Date(b.created_at) - new Date(a.created_at))
            break;
        case "4":
            eventosFiltrados.sort((a,b)=> new Date(a.created_at) - new Date(b.created_at))
            break;
    }

    if(eventosFiltrados.length != 0){
        eventosFiltrados.map(evento => {
            let rutaImagen = evento.afiches.length > 0 ? evento.afiches[0].ruta_imagen :
            "/image/aficheDefecto.png";
            contenido +=
                `
            <div class="col-md-auto">
                <div class="tarjeta card mb-3" style="width: 540px; height: 200px">
                    <div class="row g-0">
                        <div class="col-md-8">
                            <div class="card-body">
                                <h5 class="card-title fw-bold" id="nombreEvento">${evento.nombre}</h5>
                                <h6 id="tipoDeEvento">${evento.tipo_evento.nombre}</h6>
                                <hr style="height:4px; opacity:1; border:none; background-color:
                                ${evento.tipo_evento.color};" >
                                </hr>
                                <p class="cart-text">
                                    <span>Fecha del evento:</span>
```

```
                                    <span>Fecha del evento:</span>
                                    <span id="fechaInicioEvento"
                                        class="mx-2 fst-italic">${moment(evento.inicio_evento).format(
                                        "DD-MM-YYYY")}</span>
                                    <span id="fechaFinEvento"
                                        class="fst-italic">${moment(evento.fin_evento).format(
                                        "DD-MM-YYYY")}</span>
                                </p>
                                <div class="row text-end">
                                    <a href="/eventos/${evento.nombre}"
                                        id="linkEvento" class="text-decoration-none stretched-link">
                                        Saber más...</a>
                                </div>
                            </div>
                        </div>
                        <div class="d-flex p-3 align-self-center col-md-4" style="height: 195px">
                            <img src="${rutaImagen}"
                                class="img-fluid rounded-start object-fit-scale" alt="...">
                        </div>
                    </div>
                </div>
            </div>
                `;
        });
    } else {
        contenido = `
        <div class="row mt-5">
            <div class="alert alert-info w-50 mx-auto text-center" role="alert">
                No existen eventos que cumplan con tu criterio de búsqueda.
            </div>
        </div>
            `;
    }

    div.innerHTML = contenido;
};
```

Vista de mostrar eventos

Corresponde a la vista de detalle de un evento, donde se muestra información detallada sobre el evento, incluyendo su descripción, sitios de interés, y actividades asociadas. La estructura del código está organizada en secciones claramente definidas, cada una abordando un aspecto específico de la visualización del evento. Se utiliza el sistema de plantillas de Laravel con la extensión @extends para heredar de un diseño principal (layouts.app). Las secciones @section definen bloques de contenido que se insertarán en el diseño principal. La vista incluye detalles como el nombre del evento, tipo, fechas y horarios, requisitos académicos, costo de inscripción, y más. Se implementa lógica condicional para manejar casos en los que el evento tiene o no afiches, participantes, equipos, descripción, sitios de interés y actividades. El código también hace uso de componentes (por ejemplo, x-carrusel, components.modal-inscribir-participante, y components.modal-inscribir-grupo) para modularizar y reutilizar partes del código, lo que favorece la mantenibilidad y la legibilidad del mismo.

Ruta del archivo: resources/views/eventos/evento.blade.php

```
@extends('layouts.app')

@section('content')
    <div class="container">
        <!-- Header Section -->
        <div class="row">
            <div class="col-md-12">
                <h2>Detalle de evento</h2>
            </div>
            <div class="col-md-12">
                <!-- Breadcrumb Navigation -->
                <nav style="--bs-breadcrumb-divider: '>';" aria-label="breadcrumb">
                    <ol class="breadcrumb">
                        <li class="breadcrumb-item"><a href="{{ URL::to('/eventos') }}">Eventos</a></li>
                        <li class="breadcrumb-item active" aria-current="page">Detalle de evento</li>
                    </ol>
                </nav>
            </div>
        </div>

        <!-- Main Content Section -->
        <div class="row">
            <div class="col-md-12 col-main  g-5">
                <div class="row">
                    @if ($evento->afiches->count() > 0)
                        <!-- Display carousel if there are posters available -->
                        <div class="col-md-5 col-sm-5">
                            <x-carrusel :evento="$evento" />
                        </div>
                        <!-- Display event details with reduced width if posters are available -->
                        <div class="col-md-7 col-sm-7  mt-3 " style="font-size: small">
                    @else
                        <!-- Display event details with full width if no posters are available -->
                        <div class="col-md-12 col-sm-12  mt-3 " style="font-size: small">
                    @endif
                    <div class="row">
                        <!-- Display Event Name and Type -->
                        <div class="col-lg-9 col-md-12 col-sm-12 col-12">
                            <h3>{{ $evento->nombre }}</h3>
                            <p class="fs-6 mb-0">{{ $evento->tipoEvento->nombre }}
                            <p>
                        </div>
                        <!-- Display Registration Button if applicable -->
                        @if (
                            $evento->actividades->where('inscripcion', 1)->filter(function ($actividad) {
                                return $actividad->inicio_actividad <= date('Y-m-d\TH:i') && date('Y-m-d\TH:i') <= $actividad->fin_actividad ;
                            })->count() > 0)
                            <div class="col-lg-3 col-md-12 col-sm-12 col-12">
                                <div class="row mt-3 d-flex">
                                    <!-- Button to trigger registration modal -->
                                    <button type="button" class="btn btn-primary w-100 justify-content-center"
                                        data-bs-toggle="modal" data-bs-target="#modal-inscribir">
                                        {{ $evento->equipo_maximo == null ? 'Inscribirme' : 'Inscribir equipo' }}
                                    </button>
                                    <!-- Modal for participant or team registration -->
                                    @if ($evento->equipo_maximo == null)
                                        @component('components.modal-inscribir-participante')
                                        @else
                                            @component('components.modal-inscribir-grupo')
```

```
                                @endif
                                @slot('evento', $evento)
                            @endcomponent
                        </div>
                    </div>
                @endif
            </div>
            <!-- Display Participant and Team Count -->
            @if ($participantes->count() > 0)
                @php
                    $plural = $participantes->count() > 1 ? 's' : '';
                @endphp
                <div class="d-flex justify-content-end fs-6">
                    Este evento tiene {{ $participantes->count() }}
                    participante{{ $plural }} inscrito{{ $plural }}.
                </div>
            @endif
            @if ($equipos > 0)
                @php
                    $plural = $equipos > 1 ? 's' : '';
                @endphp
                <div class="d-flex justify-content-end fs-6">
                    Este evento tiene {{ $equipos }}
                    equipo{{ $plural }} inscrito{{ $plural }}.
                </div>
            @endif

            <!-- Event Information Section -->
            <hr>
            <h5>Información del evento:</h5>
            <div class="row mt-3">
```

```
                <!-- Display Event Start Date -->
                <div class="col-8">
                    <strong>Inicio del evento:</strong>
                    {{ date('d-m-Y', strtotime($evento->inicio_evento)) }}
                </div>
                <!-- Display Event Start Time -->
                <div class="col-4">
                    <strong>Hora:</strong>
                    {{ date('H:i', strtotime($evento->inicio_evento)) }}
                </div>
            </div>
            <div class="row">
                <!-- Display Event End Date -->
                <div class="col-8">
                    <strong>Fin del evento:</strong>
                    {{ date('d-m-Y', strtotime($evento->fin_evento)) }}
                </div>
                <!-- Display Event End Time -->
                <div class="col-4 ">
                    <strong>Hora:</strong>
                    {{ date('H:i', strtotime($evento->fin_evento)) }}
                </div>
            </div>

            <!-- Additional Event Information -->
            <div class="row mt-3">
                <div class="col-12">
                    <strong>Grado académico requerido:</strong>
                    {{ $evento->grado_academico == null ? 'Ninguno' : $evento->grado_academico }}
                </div>
```

```
            </div>
            @if ($evento->institucion)
                <div class="row mt-3">
                    <div class="col-12">
                        <strong>Instituciones admitidas:</strong>
                        {{ $evento->institucion }}
                    </div>
                </div>
            @endif
            @if ($evento->region)
                <div class="row mt-3">
                    <div class="col-12">
                        <strong>Región:</strong>
                        {{ $evento->region }}
                    </div>
                </div>
            @endif
            @if ($evento->genero)
                <div class="row mt-3">
                    <div class="col-12">
                        <strong>Género admitido:</strong>
                        {{ $evento->genero }}
                    </div>
                </div>
            @endif
            @if ($evento->edad_minima || $evento->edad_maxima)
                <div class="row mt-3">
                    <div class="col-12">
                        <strong>Límite de edad:</strong>
                        {{ $evento->edad_minima ? 'Desde los ' . $evento->edad_minima . ' años' : '' }}
                        {{ $evento->edad_maxima ? ($evento->edad_minima ? 'h' : 'H') . 'asta los ' . $evento->edad_maxima . ' años' : '' }}
                    </div>
                </div>
            @endif
            @if ($evento->equipo_minimo || $evento->equipo_maximo)
                <div class="row mt-3">
                    <div class="col-12">
                        <strong>El evento es por equipos: de
                            {{ $evento->equipo_minimo ? $evento->equipo_minimo : 2 }} a
                            {{ $evento->equipo_maximo }} participantes</strong>
                    </div>
                </div>
            @endif
            <div class="row mt-3 fs-5">
                @if ($evento->precio_inscripcion == null)
                    <!-- Display Free Event -->
                    <div class="row mt-3">
                        <div class="col-12">
                            <strong>Costo de inscripción:</strong>
                            GRATUITO
                        </div>
                    </div>
                @else
                    <!-- Display Event Registration Cost -->
                    <div class="row mt-3">
                        <div class="col-12">
                            <strong>Costo de inscripción:</strong>
                            {{ $evento->precio_inscripcion }} Bs.
                        </div>
                    </div>
```

```
                    @endif
                </div>
            </div>
        </div>
    </div>

    <!-- Description Section -->
    @if ($evento->descripcion !== null)
        <div class="row mt-5">
            <div class="col-md-12 {{ $evento->sitios->count() > 0 ? 'col-lg-9' : 'col-lg-12' }} border-end ">
                <h3 class="">Descripción</h3>
                <hr>
                <p class="px-4 text-break " style="text-align: justify">{!! nl2br($evento->descripcion) !!}</p>
            </div>
        @else
            <div class="row mt-5">
    @endif

    <!-- Sites of Interest Section -->
    @if ($evento->sitios->count() > 0)
        <div class="col-md-12 {{ $evento->descripcion !== null ? 'col-lg-3' : 'col-lg-12' }}">
            <h3>Sitios de interés</h3>
            <hr>
            <!-- Display Sites of Interest -->
            @foreach ($evento->sitios as $sitio)
                <div class="">
                    <a href="{{ $sitio->enlace }}" class="text-decoration-none text-primary fs-6 mt-2 text-truncate"
                        title="{{ $sitio->enlace }}" target="_blank"> {{ $sitio->titulo }}</a>
                </div>
            @endforeach
        </div>
    @endif
    </div>

    <!-- Activities Section -->
    @if ($evento->actividades->count() > 0)
        <div class="row mt-5">
            <div class="col-md-12">
                <h3>Actividades</h3>
                <hr>
                <!-- Display Event Activities -->
                @foreach ($evento->actividades as $actividad)
                    <div class="card my-3
                    @if ($actividad->fin_actividad < date('Y-m-d\TH:i')) shadow-none
                    @elseif($actividad->inicio_actividad <= date('Y-m-d\TH:i') && date('Y-m-d\TH:i') <= $actividad->fin_actividad)
                        shadow-lg bg-body rounded
                    @else
                        shadow bg-body rounded @endif
                    "
                        style="min-height: auto">
                        <div class="card-body mt-3">
                            <h5 class="card-title text-center fw-bold">{{ $actividad->nombre }}</h5>
                            <hr>
                            <div class="row">
                                <!-- Display Activity Information -->
                                @if ($actividad->inscripcion == 1)
                                    <p class="card-text col-lg-2 col-sm-3 col-4"><strong> Inicio: </strong>
                                        {{ date('d-m-Y', strtotime($actividad->inicio_actividad)) }}
                                    </p>
```

```
                                    <p class="card-text col-lg-7 col-sm-7 col-5">
                                        <strong>Hora: </strong> {{ date('H:i', strtotime($actividad->inicio_actividad)) }}
                                    </p>
                                    <div class="alert alert-primary text-center col-lg-3 col-sm-2 col-3"
                                        style="height:25px;padding:0;width:200px">
                                        Actividad de inscripción
                                    </div>
                                @else
                                    <p class="card-text col-lg-2 col-sm-4 col-6"><strong> Inicio: </strong>
                                        {{ date('d-m-Y', strtotime($actividad->inicio_actividad)) }}
                                    </p>
                                    <p class="card-text col-lg-10 col-sm-8 col-6">
                                        <strong>Hora: </strong> {{ date('H:i', strtotime($actividad->inicio_actividad)) }}
                                    </p>
                                @endif
                                <p class="card-text col-lg-2 col-sm-4 col-6"><strong> Fin: </strong>
                                    {{ date('d-m-Y', strtotime($actividad->fin_actividad)) }}
                                </p>
                                <p class="card-text col-lg-10 col-sm-8 col-6">
                                    <strong>Hora: </strong> {{ date('H:i', strtotime($actividad->fin_actividad)) }}
                                </p>
                                <p class="card-text col-12 px-4 text-break" style="text-align: justify">
                                    {!! nl2br($actividad->descripcion) !!}
                                </p>
                            </div>
                        </div>
                    </div>
                @endforeach
            </div>
        </div>
    @endif
```

```
    @if ($evento->descripcion !== null)
        <!-- Si hay una descripción, se muestra una sección con un ancho específico -->
        <div class="row mt-5">
            <div class="col-md-12 {{ $evento->sitios->count() > 0 ? 'col-lg-9' : 'col-lg-12' }} border-end ">
                <h3 class="">Descripción</h3>
                <hr>
                <!-- Sección de descripción del evento -->
                <p class="px-4 text-break " style="text-align: justify">{!! nl2br($evento->descripcion) !!}</p>
            </div>
        @else
            <!-- Si no hay descripción, se muestra una sección con el ancho completo -->
            <div class="row mt-5">
    @endif

    <!-- Sección de sitios de interés -->
    @if ($evento->sitios->count() > 0)
        <!-- Si hay sitios de interés, se muestra una columna específica para ellos -->
        <div class="col-md-12 {{ $evento->descripcion !== null ? 'col-lg-3' : 'col-lg-12' }}">
            <h3>Sitios de interés</h3>
            <hr>
            <!-- Sección que muestra los sitios de interés -->
            @foreach ($evento->sitios as $sitio)
                <div class="">
                    <a href="{{ $sitio->enlace }}" class="text-decoration-none text-primary fs-6 mt-2 text-truncate"
                        title="{{ $sitio->enlace }}" target="_blank"> {{ $sitio->titulo }}</a>
                </div>
            @endforeach
        </div>
    @endif
    </div>
```

```
<!-- Sección de actividades -->
@if ($evento->actividades->count() > 0)
    <!-- Si hay actividades, se muestra una sección específica para ellas -->
    <div class="row mt-5">
        <div class="col-md-12">
            <h3>Actividades</h3>
            <hr>
            <!-- Sección que muestra las actividades del evento -->
            @foreach ($evento->actividades as $actividad)
                <div class="card my-3
                @if ($actividad->fin_actividad < date('Y-m-d\TH:i')) shadow-none
                @elseif($actividad->inicio_actividad <= date('Y-m-d\TH:i') && date('Y-m-d\TH:i') <= $actividad->fin_actividad)
                    shadow-lg bg-body rounded
                @else
                    shadow bg-body rounded @endif
            "
                    style="min-height: auto">
                    <div class="card-body mt-3">
                        <!-- Título y detalles de la actividad -->
                        <h5 class="card-title text-center fw-bold">{{ $actividad->nombre }}</h5>
                        <hr>
                        <div class="row">
                            <!-- Información de la actividad y botón de inscripción si es aplicable -->
                            @if ($actividad->inscripcion == 1)
                                <p class="card-text col-lg-2 col-sm-3 col-4"><strong> Inicio: </strong>
                                    {{ date('d-m-Y', strtotime($actividad->inicio_actividad)) }}
                                </p>
                                <p class="card-text col-lg-7 col-sm-7 col-5">
                                    <strong>Hora: </strong> {{ date('H:i', strtotime($actividad->inicio_actividad)) }}
                                </p>
                                <div class="alert alert-primary text-center col-lg-3 col-sm-2 col-3"
                                    style="height:25px;padding:0;width:200px">
                                    Actividad de inscripción
                                </div>
                            @else
                                <p class="card-text col-lg-2 col-sm-4 col-6"><strong> Inicio: </strong>
                                    {{ date('d-m-Y', strtotime($actividad->inicio_actividad)) }}
                                </p>
                                <p class="card-text col-lg-10 col-sm-8 col-6">
                                    <strong>Hora: </strong> {{ date('H:i', strtotime($actividad->inicio_actividad)) }}
                                </p>
                            @endif
                            <p class="card-text col-lg-2 col-sm-4 col-6"><strong> Fin: </strong>
                                {{ date('d-m-Y', strtotime($actividad->fin_actividad)) }}
                            </p>
                            <p class="card-text col-lg-10 col-sm-8 col-6">
                                <strong>Hora: </strong> {{ date('H:i', strtotime($actividad->fin_actividad)) }}
                            </p>
                            <!-- Descripción de la actividad -->
                            <p class="card-text col-12 px-4 text-break" style="text-align: justify">
                                {!! nl2br($actividad->descripcion) !!}
                            </p>
                        </div>
                    </div>
                </div>
            @endforeach
        </div>
    </div>
@endif
```

```
@if ($evento->descripcion !== null)
    <!-- Si hay una descripción, se muestra una sección con un ancho específico -->
    <div class="row mt-5">
        <div class="col-md-12 {{ $evento->sitios->count() > 0 ? 'col-lg-9' : 'col-lg-12' }} border-end ">
            <h3 class="">Descripción</h3>
            <hr>
            <!-- Sección de descripción del evento -->
            <p class="px-4 text-break " style="text-align: justify">{!! nl2br($evento->descripcion) !!}</p>
        </div>
    @else
        <!-- Si no hay descripción, se muestra una sección con el ancho completo -->
        <div class="row mt-5">
@endif

<!-- Sección de sitios de interés -->
@if ($evento->sitios->count() > 0)
    <!-- Si hay sitios de interés, se muestra una columna específica para ellos -->
    <div class="col-md-12 {{ $evento->descripcion !== null ? 'col-lg-3' : 'col-lg-12' }}">
        <h3>Sitios de interés</h3>
        <hr>
        <!-- Sección que muestra los sitios de interés -->
        @foreach ($evento->sitios as $sitio)
            <div class="">
                <a href="{{ $sitio->enlace }}" class="text-decoration-none text-primary fs-6 mt-2 text-truncate"
                    title="{{ $sitio->enlace }}" target="_blank"> {{ $sitio->titulo }}</a>
            </div>
        @endforeach
    </div>
@endif
</div>
```

```
<!-- Sección de actividades -->
@if ($evento->actividades->count() > 0)
    <!-- Si hay actividades, se muestra una sección específica para ellas -->
    <div class="row mt-5">
        <div class="col-md-12">
            <h3>Actividades</h3>
            <hr>
            <!-- Sección que muestra las actividades del evento -->
            @foreach ($evento->actividades as $actividad)
                <div class="card my-3
                @if ($actividad->fin_actividad < date('Y-m-d\TH:i')) shadow-none
                @elseif($actividad->inicio_actividad <= date('Y-m-d\TH:i') && date('Y-m-d\TH:i') <= $actividad->fin_actividad)
                    shadow-lg bg-body rounded
                @else
                    shadow bg-body rounded @endif
                "
                    style="min-height: auto">
                    <div class="card-body mt-3">
                        <!-- Título y detalles de la actividad -->
                        <h5 class="card-title text-center fw-bold">{{ $actividad->nombre }}</h5>
                        <hr>
                        <div class="row">
                            <!-- Información de la actividad y botón de inscripción si es aplicable -->
                            @if ($actividad->inscripcion == 1)
                                <p class="card-text col-lg-2 col-sm-3 col-4"><strong> Inicio: </strong>
                                    {{ date('d-m-Y', strtotime($actividad->inicio_actividad)) }}
                                </p>
                                <p class="card-text col-lg-7 col-sm-7 col-5">
                                    <strong>Hora: </strong> {{ date('H:i', strtotime($actividad->inicio_actividad)) }}
                                </p>
```

```
                                <div class="alert alert-primary text-center col-lg-3 col-sm-2 col-3"
                                    style="height:25px;padding:0;width:200px">
                                    Actividad de inscripción
                                </div>
                            @else
                                <p class="card-text col-lg-2 col-sm-4 col-6"><strong> Inicio: </strong>
                                    {{ date('d-m-Y', strtotime($actividad->inicio_actividad)) }}
                                </p>
                                <p class="card-text col-lg-10 col-sm-8 col-6">
                                    <strong>Hora: </strong> {{ date('H:i', strtotime($actividad->inicio_actividad)) }}
                                </p>
                            @endif
                            <p class="card-text col-lg-2 col-sm-4 col-6"><strong> Fin: </strong>
                                {{ date('d-m-Y', strtotime($actividad->fin_actividad)) }}
                            </p>
                            <p class="card-text col-lg-10 col-sm-8 col-6">
                                <strong>Hora: </strong> {{ date('H:i', strtotime($actividad->fin_actividad)) }}
                            </p>
                            <!-- Descripción de la actividad -->
                            <p class="card-text col-12 px-4 text-break" style="text-align: justify">
                                {!! nl2br($actividad->descripcion) !!}
                            </p>
                        </div>
                    </div>
                </div>
            @endforeach
        </div>
    </div>
@endif
```

```
    </div> <!-- Cierre del contenedor principal -->

<!-- Sección de descripción (continuación) -->
@if ($evento->descripcion !== null)
    <!-- Si hay una descripción, se muestra una sección con un ancho específico -->
    <div class="row mt-5">
        <div class="col-md-12 {{ $evento->sitios->count() > 0 ? 'col-lg-9' : 'col-lg-12' }} border-end ">
            <h3 class="">Descripción</h3>
            <hr>
            <!-- Sección de descripción del evento -->
            <p class="px-4 text-break " style="text-align: justify">{!! nl2br($evento->descripcion) !!}</p>
        </div>
    @else
        <!-- Si no hay descripción, se muestra una sección con el ancho completo -->
        <div class="row mt-5">
@endif

<!-- Sección de sitios de interés -->
@if ($evento->sitios->count() > 0)
    <!-- Si hay sitios de interés, se muestra una columna específica para ellos -->
    <div class="col-md-12 {{ $evento->descripcion !== null ? 'col-lg-3' : 'col-lg-12' }}">
        <h3>Sitios de interés</h3>
        <hr>
        <!-- Sección que muestra los sitios de interés -->
        @foreach ($evento->sitios as $sitio)
            <div class="">
                <a href="{{ $sitio->enlace }}" class="text-decoration-none text-primary fs-6 mt-2 text-truncate"
                    title="{{ $sitio->enlace }}" target="_blank"> {{ $sitio->titulo }}</a>
            </div>
        @endforeach
```

```
    </div>
@endif
</div>

<!-- Sección de actividades -->
@if ($evento->actividades->count() > 0)
    <!-- Si hay actividades, se muestra una sección específica para ellas -->
    <div class="row mt-5">
        <div class="col-md-12">
            <h3>Actividades</h3>
            <hr>
            <!-- Sección que muestra las actividades del evento -->
            @foreach ($evento->actividades as $actividad)
                <div class="card my-3
                @if ($actividad->fin_actividad < date('Y-m-d\TH:i')) shadow-none
                @elseif($actividad->inicio_actividad <= date('Y-m-d\TH:i') && date('Y-m-d\TH:i') <= $actividad->fin_actividad)
                    shadow-lg bg-body rounded
                @else
                    shadow bg-body rounded @endif
                "
                    style="min-height: auto">
                    <div class="card-body mt-3">
                        <!-- Título y detalles de la actividad -->
                        <h5 class="card-title text-center fw-bold">{{ $actividad->nombre }}</h5>
                        <hr>
                        <div class="row">
                            <!-- Información de la actividad y botón de inscripción si es aplicable -->
                            @if ($actividad->inscripcion == 1)
                                <p class="card-text col-lg-2 col-sm-3 col-4"><strong> Inicio: </strong>
                                    {{ date('d-m-Y', strtotime($actividad->inicio_actividad)) }}
                                </p>
                                <p class="card-text col-lg-7 col-sm-7 col-5">
                                    <strong>Hora: </strong> {{ date('H:i', strtotime($actividad->inicio_actividad)) }}
                                </p>
                                <div class="alert alert-primary text-center col-lg-3 col-sm-2 col-3"
                                    style="height:25px;padding:0;width:200px">
                                    Actividad de inscripción
                                </div>
                            @else
                                <p class="card-text col-lg-2 col-sm-4 col-6"><strong> Inicio: </strong>
                                    {{ date('d-m-Y', strtotime($actividad->inicio_actividad)) }}
                                </p>
                                <p class="card-text col-lg-10 col-sm-8 col-6">
                                    <strong>Hora: </strong> {{ date('H:i', strtotime($actividad->inicio_actividad)) }}
                                </p>
                            @endif
                            <p class="card-text col-lg-2 col-sm-4 col-6"><strong> Fin: </strong>
                                {{ date('d-m-Y', strtotime($actividad->fin_actividad)) }}
                            </p>
                            <p class="card-text col-lg-10 col-sm-8 col-6">
                                <strong>Hora: </strong> {{ date('H:i', strtotime($actividad->fin_actividad)) }}
                            </p>
                            <!-- Descripción de la actividad -->
                            <p class="card-text col-12 px-4 text-break" style="text-align: justify">
                                {!! nl2br($actividad->descripcion) !!}
                            </p>
                        </div>
                    </div>
                </div>
            @endforeach
        </div>
    </div>
```

```
    @endif
@if ($evento->descripcion !== null)
        <!-- Si hay una descripción, se muestra una sección con un ancho específico -->
        <div class="row mt-5">
            <div class="col-md-12 {{ $evento->sitios->count() > 0 ? 'col-lg-9' : 'col-lg-12' }} border-end ">
                <h3 class="">Descripción</h3>
                <hr>
                <!-- Sección de descripción del evento -->
                <p class="px-4 text-break " style="text-align: justify">{!! nl2br($evento->descripcion) !!}</p>
            </div>
        @else
            <!-- Si no hay descripción, se muestra una sección con el ancho completo -->
            <div class="row mt-5">
    @endif
    <!-- Sección de sitios de interés -->
    @if ($evento->sitios->count() > 0)
        <!-- Si hay sitios de interés, se muestra una columna específica para ellos -->
        <div class="col-md-12 {{ $evento->descripcion !== null ? 'col-lg-3' : 'col-lg-12' }}">
            <h3>Sitios de interés</h3>
            <hr>
            <!-- Sección que muestra los sitios de interés -->
            @foreach ($evento->sitios as $sitio)
                <div class="">
                    <a href="{{ $sitio->enlace }}" class="text-decoration-none text-primary fs-6 mt-2 text-truncate"
                        title="{{ $sitio->enlace }}" target="_blank"> {{ $sitio->titulo }}</a>
                </div>
            @endforeach
        </div>
    @endif
    </div>
```

```
    <!-- Sección de actividades -->
    @if ($evento->actividades->count() > 0)
        <!-- Si hay actividades, se muestra una sección específica para ellas -->
        <div class="row mt-5">
            <div class="col-md-12">
                <h3>Actividades</h3>
                <hr>
                <!-- Sección que muestra las actividades del evento -->
                @foreach ($evento->actividades as $actividad)
                    <div class="card my-3
                    @if ($actividad->fin_actividad < date('Y-m-d\TH:i')) shadow-none
                    @elseif($actividad->inicio_actividad <= date('Y-m-d\TH:i') && date('Y-m-d\TH:i') <= $actividad->fin_actividad)
                        shadow-lg bg-body rounded
                    @else
                        shadow bg-body rounded @endif
                    "
                        style="min-height: auto">
                        <div class="card-body mt-3">
                            <!-- Título y detalles de la actividad -->
                            <h5 class="card-title text-center fw-bold">{{ $actividad->nombre }}</h5>
                            <hr>
                            <div class="row">
                                <!-- Información de la actividad y botón de inscripción si es aplicable -->
                                @if ($actividad->inscripcion == 1)
                                    <p class="card-text col-lg-2 col-sm-3 col-4"><strong> Inicio: </strong>
                                        {{ date('d-m-Y', strtotime($actividad->inicio_actividad)) }}
                                    </p>
                                    <p class="card-text col-lg-7 col-sm-7 col-5">
                                        <strong>Hora: </strong> {{ date('H:i', strtotime($actividad->inicio_actividad)) }}
                                    </p>
                                    <div class="alert alert-primary text-center col-lg-3 col-sm-2 col-3"
```

```
                                    style="height:25px;padding:0;width:200px">
                                    Actividad de inscripción
                                </div>
                            @else
                                <p class="card-text col-lg-2 col-sm-4 col-6"><strong> Inicio: </strong>
                                    {{ date('d-m-Y', strtotime($actividad->inicio_actividad)) }}
                                </p>
                                <p class="card-text col-lg-10 col-sm-8 col-6">
                                    <strong>Hora: </strong> {{ date('H:i', strtotime($actividad->inicio_actividad)) }}
                                </p>
                            @endif
                            <p class="card-text col-lg-2 col-sm-4 col-6"><strong> Fin: </strong>
                                {{ date('d-m-Y', strtotime($actividad->fin_actividad)) }}
                            </p>
                            <p class="card-text col-lg-10 col-sm-8 col-6">
                                <strong>Hora: </strong> {{ date('H:i', strtotime($actividad->fin_actividad)) }}
                            </p>
                            <!-- Descripción de la actividad -->
                            <p class="card-text col-12 px-4 text-break" style="text-align: justify">
                                {!! nl2br($actividad->descripcion) !!}
                            </p>
                        </div>
                    </div>
                </div>
            @endforeach
        </div>
    </div>
    @endif
</div> <!-- Cierre del contenedor principal -->
@endsection
```

7.7.1.3. Interfaz de usuario de mostrar eventos

La interfaz de usuario para explorar el detalle de un evento proporciona información detallada de manera estructurada. A continuación, se describe cada elemento de la interfaz:

Presentación General:

Título: "Detalle de evento" en la parte superior de la página, indicando claramente el propósito de la sección.

Breadcrumb: Un sistema de navegación visible que muestra la ruta desde la página de eventos hasta el detalle actual, mejorando la usabilidad.

Información Principal del Evento:

Nombre: Se muestra el nombre del evento en un formato destacado.

Tipo: Indicación del tipo de evento.

Afiches: Si hay afiches disponibles, se presenta un carrusel visual.

Botón de Inscripción: Permite a los usuarios inscribirse al evento, con modalidades específicas para participantes individuales o equipos, según el caso.

Detalles de Participación:

Cantidad de Participantes: Información sobre la cantidad de participantes individuales inscritos.

Cantidad de Equipos: Información sobre la cantidad de equipos inscritos.

Información Detallada del Evento:

Fechas y Horarios: Detalles sobre el inicio y fin del evento, incluyendo fechas y horas.

Requisitos Académicos: Especifica el grado académico requerido.

Instituciones Admitidas, Región, Género: Detalles adicionales sobre instituciones, región y género admitido.

Límite de Edad: Indica el rango de edad permitido.

Requisitos para Equipos: En caso de ser un evento por equipos, se informa sobre los requisitos mínimo y máximo de participantes.

Costo de Inscripción: Muestra si la inscripción es gratuita o el costo en bolivianos.

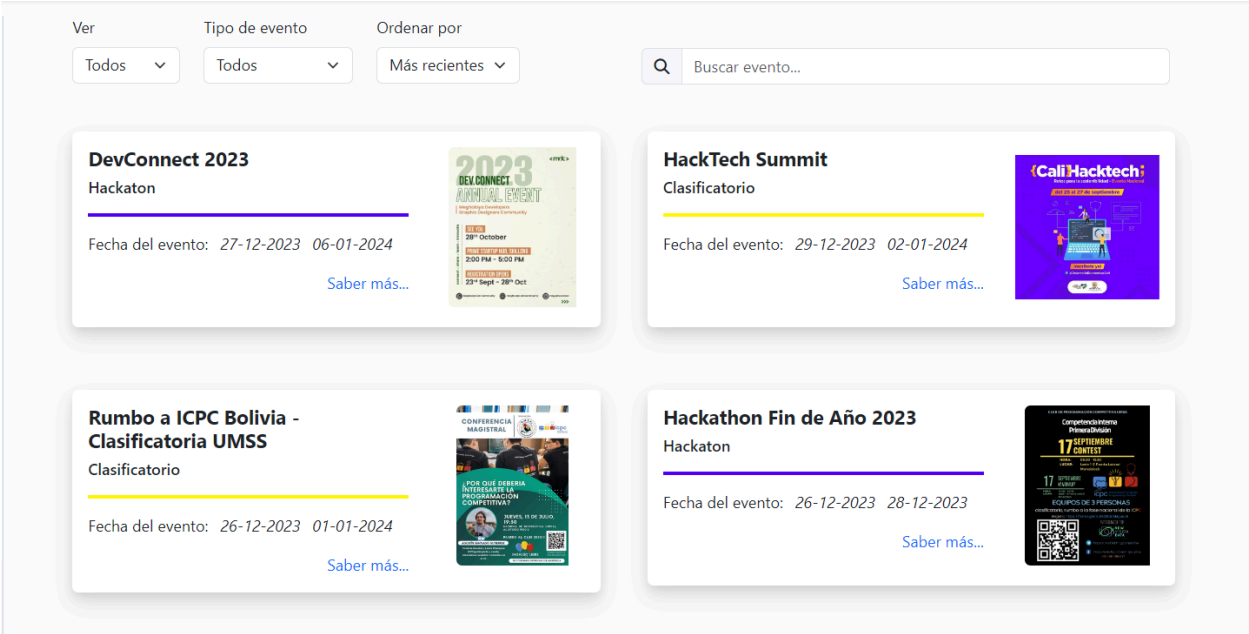
Descripción y Sitios de Interés:

Descripción del Evento: Sección de descripción, si está disponible, presentada en formato de párrafo justificado.

Sitios de Interés: Lista de sitios relevantes vinculados al evento, si los hay.

Actividades Relacionadas:

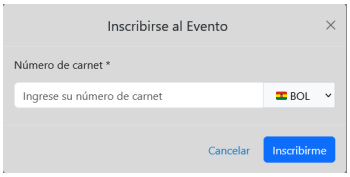
Sección de Actividades: Si el evento cuenta con actividades, se presentan de manera organizada, mostrando información sobre fechas, horarios y descripciones.

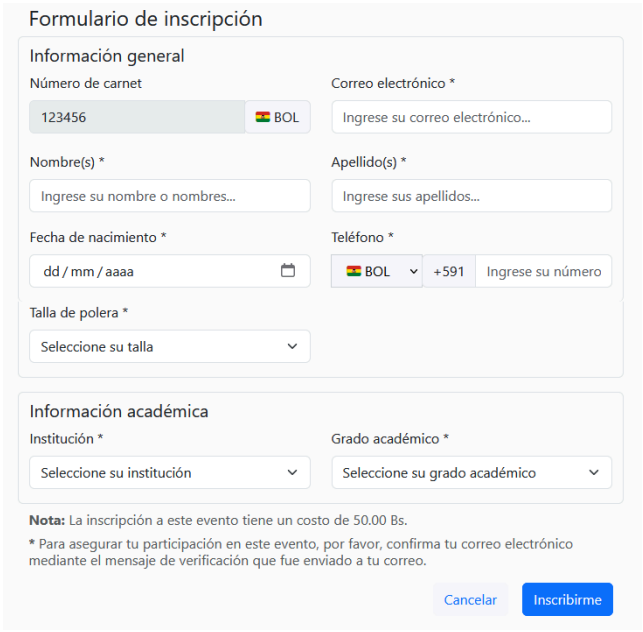


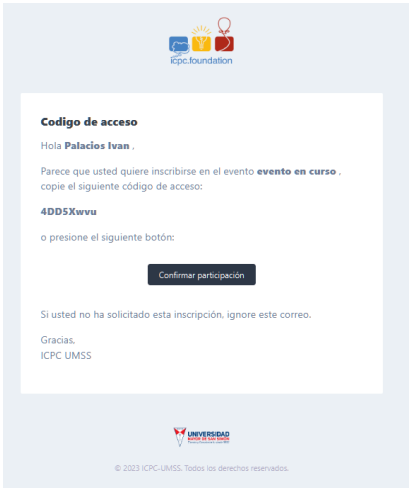
7.7.2. Inscribir a un participante a un evento

7.7.2.1. Historia de usuario de inscribir a un participante a un evento

Historia de Usuario	
Título: Inscribir a un participante a un evento	**ID: 31**
Estimación: 13	**Importancia: Alta**
Descripción: Yo como usuario interesado en participar en un evento, quiero inscribirme como participante en un evento, para asegurarse un lugar y recibir información relevante sobre el evento.	
MockUps	


Inscribirse al Evento
Número de carnet *
Ingrese su número de carnet
BOL
Cancelar
Inscribirme


Formulario de inscripción
Información general
Número de carnet
123456
BOL
Correo electrónico *
Ingrese su correo electrónico...
Nombre(s) *
Ingrese su nombre o nombres...
Apellido(s) *
Ingrese sus apellidos...
Fecha de nacimiento *
dd / mm / aaaa
Teléfono *
BOL
+591
Ingrese su número
Talla de polera *
Seleccione su talla
Información académica
Institución *
Seleccione su institución
Grado académico *
Seleccione su grado académico
Nota: La inscripción a este evento tiene un costo de 50.00 Bs.
* Para asegurar tu participación en este evento, por favor, confirma tu correo electrónico mediante el mensaje de verificación que fue enviado a tu correo.
Cancelar
Inscribirme


icpc.foundation
Codigo de acceso
Hola Palacios Ivan ,
Parece que usted quiere inscribirse en el evento evento en curso , copie el siguiente código de acceso:
4DD5Xwvu
o presione el siguiente botón:
Confirmar participación
Si usted no ha solicitado esta inscripción, ignore este correo.
Gracias,
ICPC UMSS
© 2023 ICPC-UMSS. Todos los derechos reservados.

Criterios de aceptación

1. Al hacer clic en la sección “EVENTOS” en el menú superior, se muestra una tarjeta por cada evento que existe en el sistema.
2. Al hacer clic en la tarjeta de un evento, se muestra una página con toda la información del evento y un botón “Inscribirme” (solo si el evento tiene una actividad de inscripción vigente)
3. En la vista de un evento se muestra un contador con la cantidad de participantes, si el evento tiene participantes inscritos .
4. Al hacer clic en el botón “Inscribirme” en el evento, se muestra el modal “Inscribirse al evento”, con el campo “Número de carnet *”, y los botones “Cancelar” e “Inscribirme”.
5. El campo “Número de carnet *”, es obligatorio.
6. Al hacer clic en el botón “Cancelar” en el modal “Inscribirse al evento”, el modal se cierra.
7. Al hacer clic en el botón “Inscribirme” en el modal “Inscribirse al evento”, si el número de carnet ingresado nunca fue usado en el sistema, se muestra el “Formulario de inscripción”.
8. Al hacer clic en el botón “Inscribirme” en el modal “Inscribirse al evento”, si el número de carnet ingresado fue usado en el sistema con anterioridad pero no en el mismo evento, se envía un código de acceso al correo del participante.
9. Al hacer clic en el botón “Inscribirme” en el modal “Inscribirse al evento”, si el número de carnet ingresado fue usado en el sistema con anterioridad pero no en el mismo evento, el botón “Inscribirme” es reemplazado por el botón “Verificar”.
10. Al hacer clic en el botón “Inscribirme” en el modal “Inscribirse al evento”, si el número de carnet ingresado fue usado en el sistema con anterioridad pero no en el mismo evento, se muestra el campo “Código de acceso *” y el enlace“Reenviar código”.
11. Al hacer clic en el botón “Reenviar código”, se envía un código de acceso al correo del participante.
12. El correo electrónico contiene el código de acceso del participante.

13. El campo “Código de acceso *”, es obligatorio.
14. Al hacer clic en el botón “Verificar” en el modal “Inscribirse al evento”, si el código de acceso ingresado es válido, se muestra el “Formulario de inscripción”.
15. Al hacer clic en el botón “Inscribirme”, si el número de carnet ingresado fue usado en el sistema y en el evento, se muestra una alerta temporal con el mensaje “El participante ya se encuentra inscrito en este evento”.
16. El “Formulario de inscripción”, tiene los siguientes campos: “Número de carnet”, “Correo electrónico *”, “Nombre(s) *”, “Apellido(s) *”, “Fecha de nacimiento *”, “Teléfono *” y los botones “Cancelar” e “Inscribirme”.
17. Si el participante ya estuvo registrado en algún otro evento anteriormente, al momento de verificar el código de acceso que le llega a su correo, se recuperan sus datos en el formulario de inscripción.
18. Si el evento tiene restricción de género, se muestra el selector “Género *”, el cual se encuentra deshabilitado, en el “Formulario de inscripción”
19. Si el evento requiere talla de polera, se muestra el selector “Talla de polera *”, en el “Formulario de inscripción”
20. Si el evento tiene restricción en las instituciones admitidas, se muestra el selector “Institución *”, en el “Formulario de inscripción”
21. Si el evento tiene restricción en el grado académico admitido, se muestra el selector “Grado académico *”, en el “Formulario de inscripción”.
22. Todos los campos en el “Formulario de inscripción” que no estén deshabilitados, son obligatorios.
23. Al hacer clic en el botón “Cancelar” en el “Formulario de inscripción”, todos los campos regresan a los valores iniciales.
24. Al hacer clic en el botón “Inscribirme” en el “Formulario de inscripción”, si toda la información es válida, se muestra una alerta temporal con el mensaje “Inscrito correctamente, por favor, verifica tu correo.”.
25. Al hacer clic en el botón “Inscribirme” en el “Formulario de inscripción”, si es un participante que ya estaba registrado en algún evento anteriormente, se muestra una alerta temporal con el mensaje “Inscrito correctamente”.
26. Al hacer clic en el botón “Inscribirme” en el “Formulario de inscripción”, si toda la información es válida, después de un breve periodo de tiempo se regresa al evento.
27. El correo de verificación tiene un botón para confirmar la participación.
28. Al hacer clic en el botón “Confirmar participación” del correo electrónico, se le envía a una vista con el mensaje “Correo verificado correctamente.”.
29. Si el correo ya ha sido verificado con anterioridad, se le muestra una vista con el mensaje “El correo ya ha sido verificado.”.

7.7.2.2. Código fuente de inscribir un participante a un evento

Controlador de inscribir un participante a un evento

Este controlador se encarga de gestionar la inscripción de participantes a eventos, así como de mostrar información relacionada con los participantes y eventos. En primer

lugar, el controlador contiene métodos para mostrar todos los participantes (index) y obtener los participantes inscritos en un evento específico cuyos correos han sido confirmados (participantesEvento). Además, implementa el método inscribirEvento, encargado de gestionar la inscripción de un participante a un evento. Este método verifica la existencia del participante a través de su número de identificación (CI) y correo electrónico, actualizando o creando el participante según sea necesario. Luego, se realiza la inscripción del participante al evento, se envía un correo de confirmación y se devuelve un mensaje de éxito o error. Los métodos privados storeInscribir y store son utilizados internamente para almacenar la información de la inscripción y del nuevo participante, respectivamente.

Ruta del archivo: app/Http/Controller/ParticipanteController.php

```php
/**
 * Muestra todos los participantes.
 */
public function index()
{
    $participantes = Participante::all();
    return $participantes;
}
/**
 * Obtiene los participantes inscritos en un evento específico cuyos correos han sido confirmados.
 */
public function participantesEvento($id)
{
    $participantes = Inscrito::where('id_evento', $id)->with(['participante' => function ($q) {
        $q->where('correo_confirmado', 1);
    }])->whereHas('participante', function ($q) {
        $q->where('correo_confirmado', 1);
    })->get();
    return $participantes;
}
/**
 * Maneja la inscripción de un participante a un evento.
 */
public function inscribirEvento(Request $request)
{
    try {
        // Busca si existe un participante con el mismo CI y correo no confirmado.
        $participante = Participante::where('correo_confirmado', 0)
            ->where('ci', $request->ci)
            ->first();
        if ($participante) {
            // Actualiza los datos del participante.
            $participante = $this->update($request, $participante->id);
            // Verifica si ya está inscrito en el evento.
            $inscrito = Inscrito::where("id_evento", $request->id_evento)
                ->where("id_participante", $participante->id)
                ->first();
```

```
            // Si no está inscrito, procede con la inscripción.
            if (!$inscrito) {
                $this->storeInscribir($request, $participante->id);
            }

            // Envía un correo de confirmación al participante.
            $evento = Evento::find($request->id_evento);
            Mail::to($request->correo)->send(new ConfirmacionParticipante($participante, $evento));

            return ['mensaje' => 'Inscrito correctamente, por favor, verifica tu correo.', 'error'
            => false];
        }

        // Si ya existe el participante y se proporciona el ID, realiza la inscripción directamente.
        if ($request->id_participante) {
            $this->storeInscribir($request, $request->id_participante);
            return ['mensaje' => 'Inscrito correctamente.', 'error' => false];
        } else {
            // Si no existe el participante, lo crea y realiza la inscripción.
            $participante = $this->store($request);
            $this->storeInscribir($request, $participante->id);

            // Envía un correo de confirmación al participante.
            $evento = Evento::find($request->id_evento);
            Mail::to($request->correo)->send(new ConfirmacionParticipante($participante, $evento));

            return ['mensaje' => 'Inscrito correctamente, por favor, verifica tu correo.', 'error'
            => false];
        }
    } catch (\Exception $e) {
        return ['mensaje' => $e->getMessage(), 'error' => true];
    }
}
```

```
    * Almacena la información de la inscripción de un participante a un evento.
    */
    private function storeInscribir(Request $request, $id_participante)
    {
        try {
            $inscribir = new Inscrito();
            $inscribir->id_evento = $request->id_evento;
            $inscribir->id_participante = $id_participante;
            $inscribir->institucion = $request->institucion;
            $inscribir->grado_academico = $request->grado_academico;
            $inscribir->genero = $request->genero;
            $inscribir->talla = $request->talla;
            $inscribir->save();
            return ['mensaje' => 'Inscripción almacenada correctamente.', 'error' => false];
        } catch (\Exception $e) {
            return ['mensaje' => $e->getMessage(), 'error' => true];
        }
    }
    /**
    * Almacena la información de un nuevo participante.
    */
    private function store(Request $request)
    {
        try {
            $participante = new Participante();
            $participante->ci                = $request->ci;
            $participante->nombres           = $request->nombres;
            $participante->apellidos         = $request->apellidos;
            $participante->correo            = $request->correo;
            $participante->codigo_telefono   = $request->codigo_telefono;
            $participante->telefono          = $request->telefono;
            $participante->fecha_nacimiento  = $request->fecha_nacimiento;
            $participante->pais              = $request->pais;
            $participante->codigo            = Str::random(8);
            $participante->save();
            return $participante;
        } catch (\Exception $e) {
            return ['mensaje' => $e->getMessage(), 'error' => true];
        }
    }
```

Código JavaScript de inscribir participante a un evento

Proporciona la funcionalidad para gestionar la inscripción de participantes a eventos en un formulario modal. Se inicia al cargar la página, donde se carga una lista de países y se almacena el valor del atributo "onclick" de un botón principal. La función validarDatos se activa al intentar enviar el formulario, verificando si los datos ingresados son válidos antes de proceder con la inscripción. La lógica de inscripción incluye la verificación de la existencia del participante y su estado de inscripción, así como el manejo de casos en los que el participante ya está inscrito o registrado. Se utiliza la librería Axios para realizar peticiones HTTP al servidor. El código también gestiona la visualización y validación de un código de acceso, permitiendo a los participantes verificar su inscripción mediante un código enviado a su correo electrónico. Además, se actualiza dinámicamente el patrón del campo de código de acceso para garantizar su validez. En resumen, el código proporciona una experiencia de usuario interactiva y guiada para la inscripción a eventos, integrando funciones de verificación de datos y envío de códigos de acceso.

Ruta del archivo: public/js/Inscripciones/inscribirParticipante.js

```
// Obtiene referencias a elementos del DOM
const formModalInscripcion = document.getElementById("formModalInscripcion");
const inputCI = document.getElementById("carnetParticipante");
const feedback = document.getElementById("validacionCarnetFeedback");
const displayCodigo = document.getElementById("displayCodAcceso");
const inputCod = document.getElementById("codParticipante");
let idEvento;
let idParticipante;
let onclick;
// Función que se ejecuta cuando la ventana ha terminado de cargar
window.addEventListener("load", async () => {
    // Obtiene el valor del atributo "onclick" del botón principal
    onclick = document.getElementById("botonPrincipal").getAttribute("onclick");
    // Carga la lista de países en el formulario
    cargarPaises();
});
// Función para cargar la lista de países en el formulario
const cargarPaises = () => {
    let select = document.getElementById("selectPais");
    let options = "";
    PAISES.map(pais => {
        options += `
            <option title=${pais.name_es} value="${pais.emoji} ${pais.code_3}" ${pais.code_3 === "BOL" ?
            "selected" : ""}>
                ${pais.emoji} ${pais.code_3}
            </option>
        `;
    });
    select.innerHTML = options;
};
// Función para validar datos antes de la inscripción
const validarDatos = (idEv) => {
    idEvento = idEv;
    if (formModalInscripcion.checkValidity()) {
        verificarInscripcion(idEvento);
    } else {
        formModalInscripcion.classList.add("was-validated");
    }
};
```

```
// Función para establecer el mensaje de retroalimentación del número de carnet
const setCarnetFeedBack = () => {
    if (inputCI.value === "") {
        feedback.innerText = "El número de carnet no puede estar vacío.";
    } else {
        feedback.innerText = "Número de carnet no válido.";
    }
};

// Función para restablecer el formulario modal
const resetModal = () => {
    displayCodigo.style.display = "none";
    inputCod.removeAttribute("required");
    let boton = document.getElementById("botonPrincipal");
    boton.innerText = "Inscribirme";
    boton.setAttribute("onclick", onclick);
    formModalInscripcion.reset();
    formModalInscripcion.classList.remove("was-validated");
};
// Función para verificar la inscripción de un participante
const verificarInscripcion = async (idEvento) => {
    let formData = new FormData();
    formData.append("ci", inputCI.value);
    formData.append("id_evento", idEvento);
    let estaInscrito = await axios.post("/api/participante/existe", formData).then(response => {
        return response.data;
    });
    if (estaInscrito.error) {
        participanteYaInscrito(estaInscrito);
    } else {
        if (estaInscrito.participante) {
            estaRegistrado(estaInscrito.participante);
        } else {
            localStorage.setItem("paisCarnet", document.getElementById("selectPais").value);
            window.location.href = "/eventos/inscripcion-evento/" + idEvento + "/" + inputCI.value;
        }
    }
};
```

```
// Función para manejar el caso en que el participante ya está inscrito
const participanteYaInscrito = (response) => {
    $('#modal-inscribir').modal('hide');
    resetModal();
    mostrarAlerta(
        "Éxito",
        response.mensaje,
        response.error ? "danger" : "success"
    );
};
// Función para manejar el caso en que el participante ya está registrado
const estaRegistrado = (participante) => {
    idParticipante = participante.id;
    enviarCodigoAcceso();
    let mensaje = "El código de acceso se envió al correo ";
    let correo = participante.correo.split("@");
    correo[0] = correo[0].substring(0, 2) + correo[0].substring(2).replace(/./g, '*');
    document.getElementById("correoParticipante").innerText = mensaje + correo[0] + "@" + correo[1];
    displayCodigo.style.display = "block";
    inputCod.setAttribute("required", "");
    let boton = document.getElementById("botonPrincipal");
    boton.innerText = "Verificar";
    boton.setAttribute("onclick", "validarCodigoAcceso()");
};
// Función para enviar el código de acceso al participante
const enviarCodigoAcceso = async () => {
    let data = await axios.post("/api/participante/enviarCodigo/" + idEvento + "/" + idParticipante
    ).then(response => {
        return response.data;
    });
    return data;
};
```

```
// Función para reenviar el código de acceso
const reEnviarCodigo = async () => {
    let res = await enviarCodigoAcceso();
    mostrarAlerta(
        "Éxito",
        res.mensaje,
        res.error ? "danger" : "success"
    );
};

// Función para validar el código de acceso ingresado por el participante
const validarCodigoAcceso = () => {
    if (formModalInscripcion.checkValidity()) {
        verificarCodigoAcceso();
    } else {
        formModalInscripcion.classList.add("was-validated");
        actualizarPattern();
    }
};

// Función para verificar el código de acceso con el servidor
const verificarCodigoAcceso = async () => {
    let formData = new FormData();
    formData.append("codigo", inputCod.value);
    let data = await axios.post("/api/participante/verificarCodigo/" + idParticipante, formData
    ).then(response => {
        return response.data;
    });
    if (data.error) {
        actualizarPattern();
        formModalInscripcion.classList.add("was-validated");
    } else {
        let ci = inputCI.value;
        resetModal();
        window.location.href = "/eventos/inscripcion-evento/" + idEvento + "/" + ci;
    }
};
```

```
// Variables para actualizar el patrón del campo de código de acceso
let patternBase = "^(";
const actualizarPattern = () => {
    let nuevoPattern = patternBase + "(?!" + inputCod.value + "$)";
    patternBase = nuevoPattern;
    nuevoPattern = nuevoPattern + ").+";
    inputCod.setAttribute("pattern", nuevoPattern);
};
```

Vista de inscribir un participante a un evento

La vista extiende la plantilla base de la aplicación y presenta varios campos, incluidos datos personales, información académica y detalles específicos del evento. El formulario está prellenado con datos del participante cuando está disponible, y algunos campos pueden estar deshabilitados en función de si la información ya ha sido proporcionada. El formulario incluye validaciones de campos requeridos y restricciones como la fecha mínima y máxima de nacimiento. Además, se proporcionan mensajes informativos dinámicos sobre la restricción de edad y género del evento. El código también maneja opciones adicionales, como la talla de la camiseta y la confirmación del correo electrónico. Se incluyen notas opcionales sobre el costo de inscripción y la importancia de verificar el correo electrónico para confirmar la participación. La vista incorpora estilos personalizados y scripts JavaScript para mejorar la experiencia del usuario, como

la selección de emojis de banderas de países y la interacción con la interfaz del formulario.

Ruta del archivo: resources/views/inscripciones/participante.blade.php

```
@extends('layouts.app')

@section('content')
    <!-- Contenedor principal del formulario de inscripción -->
    <div class="container mb-3" id="formularioInscripcionEvento">
        <div class="d-flex justify-content-center">
            <div class="col-md-8">
                <!-- Título del evento y encabezado del formulario -->
                <h2 id="nombreEvento">{{ $evento->nombre }}</h2>
                <h4>Formulario de inscripción</h4>

                <!-- Inicio del formulario con validación -->
                <form class="needs-validation" novalidate id="formInscripcionParticipante">
                    <!-- Sección de información general -->
                    <div class="row border border-bottom-0 rounded-top">
                        <h5 class="mt-2">Información general</h5>
                        <!-- Campo para el número de carnet y código de país -->
                        <div class="col-md-6">
                            <div class="mb-3">
                                <label for="carnetParticipante" class="form-label">Número de carnet</label>
                                <!-- Campo de entrada y visualización del código de país -->
                                <div class="input-group">
                                    <input type="text" class="form-control" id="carnetParticipante"
                                        value="{{ $participante->ci }}" maxlength="16" disabled>
                                    <span class="input-group-text rounded-start"
                                        id="codPaisCarnet">{{ $participante->pais }}</span>
                                </div>
                            </div>
                        </div>
                        <!-- Campo para el correo electrónico -->
                        <div class="col-md-6">
                            <div class="mb-3">
                                <label for="correoParticipante" class="form-label">Correo electrónico *</label>
                                <input type="email" class="form-control" id="correoParticipante" maxlength="64"
                                    placeholder="Ingrese su correo electrónico..." required
                                    value="{{ $participante->correo }}" {{ $participante->correo ? 'disabled' : '' }}>
                            </div>
                        </div>
                    </div>

                    <!-- Sección de nombres y apellidos -->
                    <div class="row border border-top-0 border-bottom-0">
                        <!-- Campo para el nombre -->
                        <div class="col-md-6">
                            <div class="mb-3 mt-1">
                                <label for="nombreParticipante" class="form-label">Nombre(s) *</label>
                                <input type="text" class="form-control" id="nombreParticipante" maxlength="64"
                                    placeholder="Ingrese su nombre o nombres..." pattern="[a-zA-Z ]+" required
                                    value="{{ $participante->nombres }}" {{ $participante->nombres ? 'disabled' : '' }}>
                            </div>
                        </div>
                        <!-- Campo para el apellido -->
                        <div class="col-md-6">
                            <div class="mb-3 mt-1">
                                <label for="apellidoParticipante" class="form-label">Apellido(s) *</label>
                                <input type="text" class="form-control" id="apellidoParticipante" maxlength="64"
                                    placeholder="Ingrese sus apellidos..." pattern="[a-zA-Z ]+" required
                                    value="{{ $participante->apellidos }}" {{ $participante->apellidos ? 'disabled' : '' }}>
                            </div>
```

```
            </div>
        </div>

        <!-- Sección de fecha de nacimiento y teléfono -->
        <div class="row border border-top-0 border-bottom-0 rounded-bottom">
            <!-- Campo para la fecha de nacimiento -->
            <div class="col-md-6">
                <div class="mb-3">
                    <!-- Cálculo de la edad mínima y máxima permitida -->
                    @php
                        $edad_minima = $evento->edad_minima != null ? $evento->edad_minima : '10';
                        $edad_maxima = $evento->edad_maxima != null ? $evento->edad_maxima : '99';
                    @endphp
                    <label for="fechaNacParticipante" class="form-label">Fecha de nacimiento *</label>
                    <input type="date" class="form-control" id="fechaNacParticipante"
                        min={{ date('Y-m-d', strtotime('-' . $edad_maxima . ' year')) }}
                        max={{ date('Y-m-d', strtotime('-' . $edad_minima . ' year')) }} required
                        value="{{ $participante->fecha_nacimiento }}"
                        {{ $participante->fecha_nacimiento ? 'disabled' : '' }}>
                    <div class="form-text">
                        <!-- Mensaje sobre la restricción de edad -->
                        @if ($evento->edad_minima != null && $evento->edad_maxima == null)
                            Debes ser mayor de {{ $edad_minima }} años para inscribirte al evento.
                        @elseif ($evento->edad_minima == null && $evento->edad_maxima != null)
                            Debes ser menor de {{ $edad_maxima }} años para inscribirte al evento.
                        @elseif ($evento->edad_minima != null && $evento->edad_maxima != null)
                            Debes tener entre {{ $edad_minima }} y {{ $edad_maxima }} años
                            para
                            inscribirte al evento.
                        @endif
                    </div>
                </div>
            </div>
            <!-- Campo para el teléfono -->
            <div class="col-md-6">
                <div class="mb-3">
                    <label for="telefonoParticipante" class="form-label">Teléfono *</label>
                    <div class="input-group">
                        <!-- Campos ocultos para el código de país y el ID del participante -->
                        <input hidden value="{{ $participante->codigo_telefono }}" id="codPaisLit">
                        <input hidden value="{{ $participante->id }}" id="idParticipante">
                        <!-- Menú desplegable para seleccionar el código de país -->
                        <select class="custom-select" id="selectPais" onchange="setCodPais()"
                            {{ $participante->codigo_telefono ? 'disabled' : '' }}>
                        </select>
                        <!-- Visualización del código de país seleccionado -->
                        <span class="input-group-text rounded-start" id="codPais"></span>
                        <!-- Campo para el número de teléfono -->
                        <input type="tel" class="form-control" id="telefonoParticipante" maxlength="15"
                            pattern="[0-9]{6,15}" placeholder="Ingrese su número telefónico..." required
                            value="{{ $participante->telefono }}"
                            {{ $participante->codigo_telefono ? 'disabled' : '' }}>
                    </div>
                </div>
            </div>
        </div>
```

```
<!-- Sección de género y talla de polera -->
<div class="row border border-top-0 rounded-bottom">
    <!-- Campo para el género (opcional) -->
    <div class="col-md-6" {{ $evento->genero != null ? '' : 'hidden' }}>
        <div class="mb-3">
            <label for="generoParticipante" class="form-label">Género *</label>
            <select id="generoParticipante" class="form-select form-select" aria-label=".form-select-sm"
                disabled>
                <option value={{ $evento->genero ? $evento->genero : '' }} selected>
                    {{ $evento->genero ? $evento->genero : '' }}
                </option>
            </select>
            <div class="form-text">
                <!-- Mensaje sobre la restricción de género -->
                El evento solo admite personas de género {{ strtolower($evento->genero) }}.
            </div>
        </div>
    </div>
    <!-- Campo para la talla de polera (opcional) -->
    <div class="col-md-6" {{ $evento->talla == 'on' ? '' : 'hidden' }}>
        <div class="mb-3">
            <label for="tallaParticipante" class="form-label">Talla de polera *</label>
            <select id="tallaParticipante" class="form-select form-select" aria-label=".form-select-sm"
                {{ $evento->talla == 'on' ? 'required' : '' }}>
                <option value="" selected>Seleccione su talla</option>
                <option value="XS">XS</option>
                <option value="S">S</option>
                <option value="M">M</option>
                <option value="L">L</option>
                <option value="XL">XL</option>
                <option value="XXL">XXL</option>
            </select>
        </div>
    </div>
</div>

<!-- Sección de información académica (opcional) -->
<div class="row border rounded mt-3"
    {{ $evento->institucion != null || $evento->grado_academico != null ? '' : 'hidden' }}>
    <h5 class="mt-2">Información académica</h5>
    <!-- Campo para la institución (opcional) -->
    <div class="col-md-6" {{ $evento->institucion != null ? '' : 'hidden' }}>
        <label for="institucionParticipante" class="form-label">Institución *</label>
        <select id="institucionParticipante" class="form-select form-select"
            aria-label=".form-select-sm" {{ $evento->institucion != null ? 'required' : '' }}>
            <option value="" selected>Seleccione su institución</option>
            <!-- Obtener y mostrar las instituciones del evento -->
            @php
                $instituciones = explode('-', $evento->institucion);
            @endphp
            @foreach ($instituciones as $institucion)
                <option value={{ $institucion }}>{{ $institucion }}</option>
            @endforeach
        </select>
    </div>
    <!-- Campo para el grado académico (opcional) -->
    <div class="col-md-6 mb-3" {{ $evento->grado_academico != null ? '' : 'hidden' }}>
        <label for="gradoAcademicoParticipante" class="form-label">Grado académico *</label>
        <select id="gradoAcademicoParticipante" class="form-select form-select"
            aria-label=".form-select-sm" {{ $evento->grado_academico != null ? 'required' : '' }}>
```

```
                                <option value="" selected>Seleccione su grado académico</option>
                                <!-- Obtener y mostrar los grados académicos del evento -->
                                @php
                                    $grados = explode('-', $evento->grado_academico);
                                @endphp
                                @foreach ($grados as $grado)
                                    <option value={{ $grado }}>{{ $grado }}</option>
                                @endforeach
                            </select>
                        </div>
                    </div>
                </form>

                <!-- Nota sobre el costo de la inscripción (opcional) -->
                @if ($evento->precio_inscripcion != null)
                    <h6 class="text-muted mt-2"><b>Nota:</b> La inscripción a este evento tiene un costo de
                        {{ $evento->precio_inscripcion }} Bs.</h6>
                @endif

                <!-- Nota sobre la confirmación del correo electrónico (opcional) -->
                @if (!$participante->nombres)
                    <h6 class="text-muted mt-2">
                        <b>*</b> Para asegurar tu participación en este evento, por favor, confirma tu correo
                        electrónico mediante el mensaje de verificación que se enviará a tu correo.
                    </h6>
                @endif

                <!-- Botones para cancelar e inscribirse -->
                <div class="d-flex justify-content-end mt-3">
                    <button type="button" class="btn btn-light text-primary me-3" onclick="resetForm()">
                        Cancelar
                    </button>
                    <button type="button" class="btn btn-primary" onclick="validarInputs()">
                        Inscribirme
                    </button>
                </div>
            </div>
        </div>
    </div>

    <!-- Enlaces y scripts adicionales -->
    <link href="{{ asset('css/participante.css') }}" rel="stylesheet">
    <script type="module" defer>
        // Importación y polifill para emojis de banderas de países
        import { polyfillCountryFlagEmojis } from "https://cdn.skypack.dev/country-flag-emoji-polyfill";
        polyfillCountryFlagEmojis();
    </script>
    <style>
        /* Estilos adicionales... */
        * {
            font-family: "Twemoji Country Flags", "Segoe UI", "Helvetica Neue", "Noto Sans", "Liberation Sans",
            "Apple Color Emoji", "Segoe UI Emoji", "Segoe UI Symbol", "Noto Color Emoji";
        }
    </style>
    <script src="{{ asset('js/Inscripciones/codPaises.js') }}" defer></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/dayjs/1.11.10/dayjs.min.js"></script>
    <script src="{{ asset('js/Inscripciones/participante.js') }}" defer></script>
@endsection
```

7.7.2.3.Interfaz de usuario de inscribir un participante a un evento

La interfaz de usuario diseñada para el formulario de inscripción de eventos ofrece una experiencia intuitiva y organizada. A continuación, se detallan los elementos clave de la interfaz:

Encabezado:

Nombre del Evento: Muestra de manera prominente el nombre del evento en el encabezado para una identificación clara.

Secciones del Formulario:

Información General:

Número de Carnet: Campo que muestra el número de carnet del participante, deshabilitado para edición.

Correo Electrónico: Campo para ingresar el correo electrónico, con la opción de estar prellenado y deshabilitado si ya existe.

Nombre(s) y Apellido(s): Campos para ingresar el nombre y apellido del participante, con restricciones de patrones y deshabilitados si ya están proporcionados.

Fecha de Nacimiento: Campo de fecha con validación dinámica según el rango especificado por el evento.

Teléfono: Campo para ingresar el número de teléfono, con un menú desplegable para seleccionar el código de país.

Género: Campo desplegable que muestra el género del participante, si es requerido por el evento.

Talla de Polera: Campo desplegable para seleccionar la talla de la polera, si esta opción está habilitada.

Institución y Grado Académico: Campos desplegables que muestran la institución y el grado académico, si se requieren estos datos.

Mensajes Informativos:

Nota de Costo: Se muestra un mensaje si el evento tiene un costo de inscripción, indicando la cantidad en Bolivianos (Bs.).

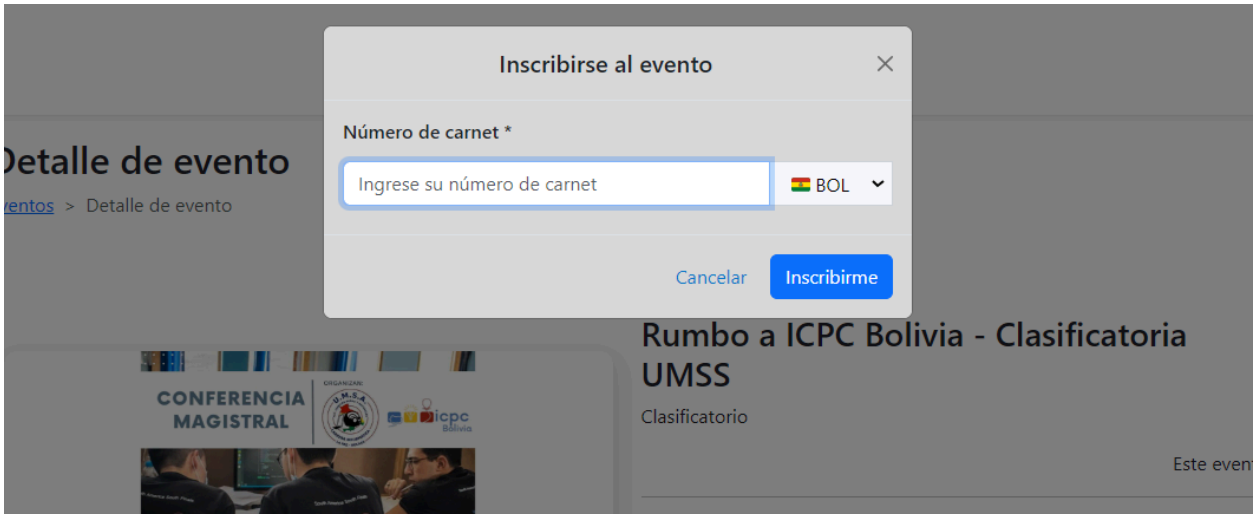
Solicitud de Confirmación: Se presenta un mensaje instando al participante a confirmar su correo electrónico para asegurar la participación.

Botones de Acción:

Botones al Final del Formulario:

Cancelar: Botón para cancelar la operación actual y restablecer el formulario.

Inscribirme: Botón para validar los datos ingresados y enviar la inscripción al evento.



Rumbo a ICPC Bolivia - Clasificatoria UMSS

Formulario de inscripción

Información general

Número de carnet

34567890 BOL

Correo electrónico *

Ingrese su correo electrónico...

Nombre(s) *

Ingrese su nombre o nombres...

Apellido(s) *

Ingrese sus apellidos...

Fecha de nacimiento *

dd/mm/aaaa

Teléfono *

BOL +591 Ingrese su número telefónico...

Talla de polera *

Seleccione su talla

Información académica

Institución *

Seleccione su institución

Grado académico *

Seleccione su grado académico

* Para asegurar tu participación en este evento, por favor, confirma tu correo electrónico mediante el mensaje de verificación que se enviará a tu correo.

Cancelar Inscribirme

7.7.3. Inscribir un equipo a un evento

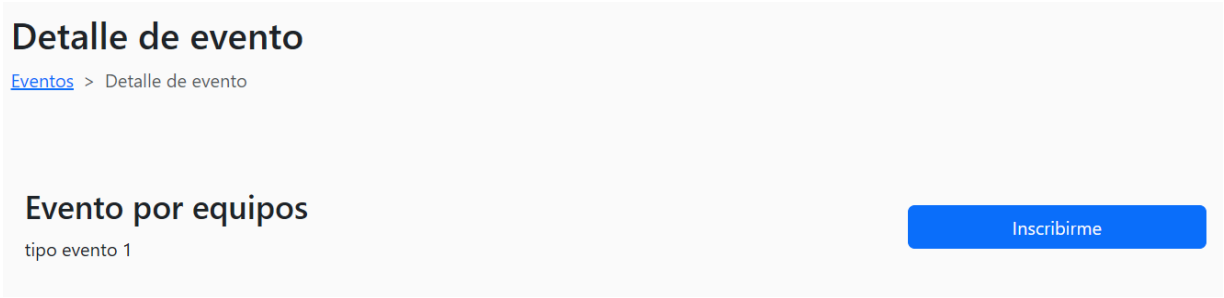
7.7.3.1. Historia de usuario de inscribir a un equipo a un evento

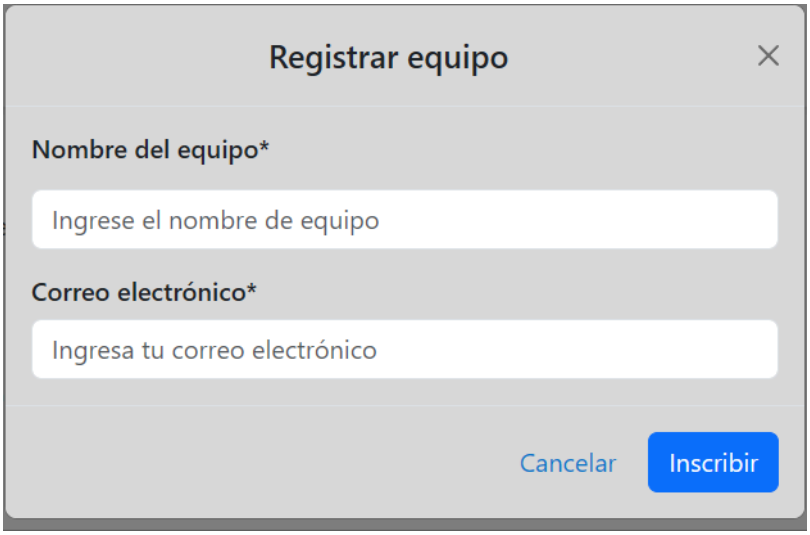
Historia de Usuario	
Título: Inscribir un equipo a un evento	**ID: 32**
Estimación: 8	**Importancia: Media**

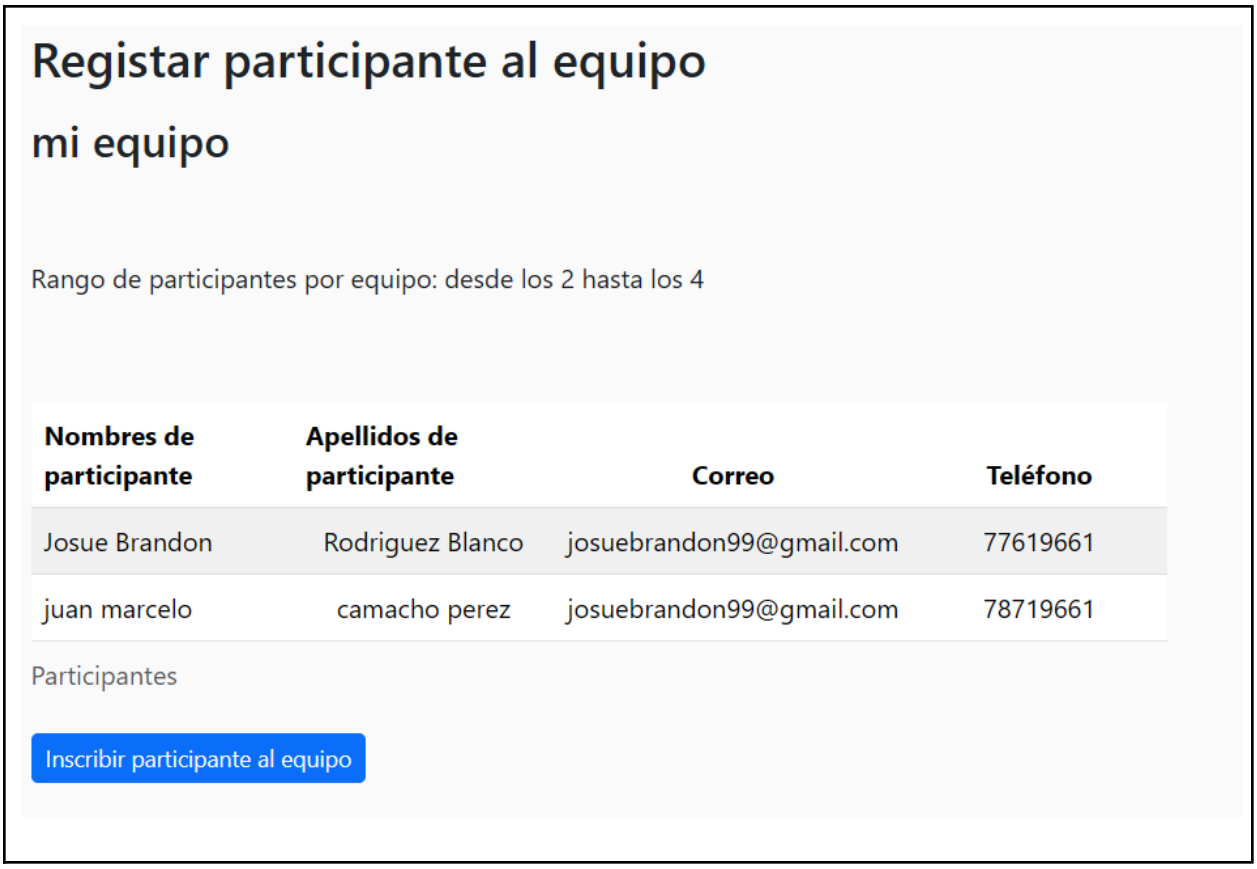
Descripción:

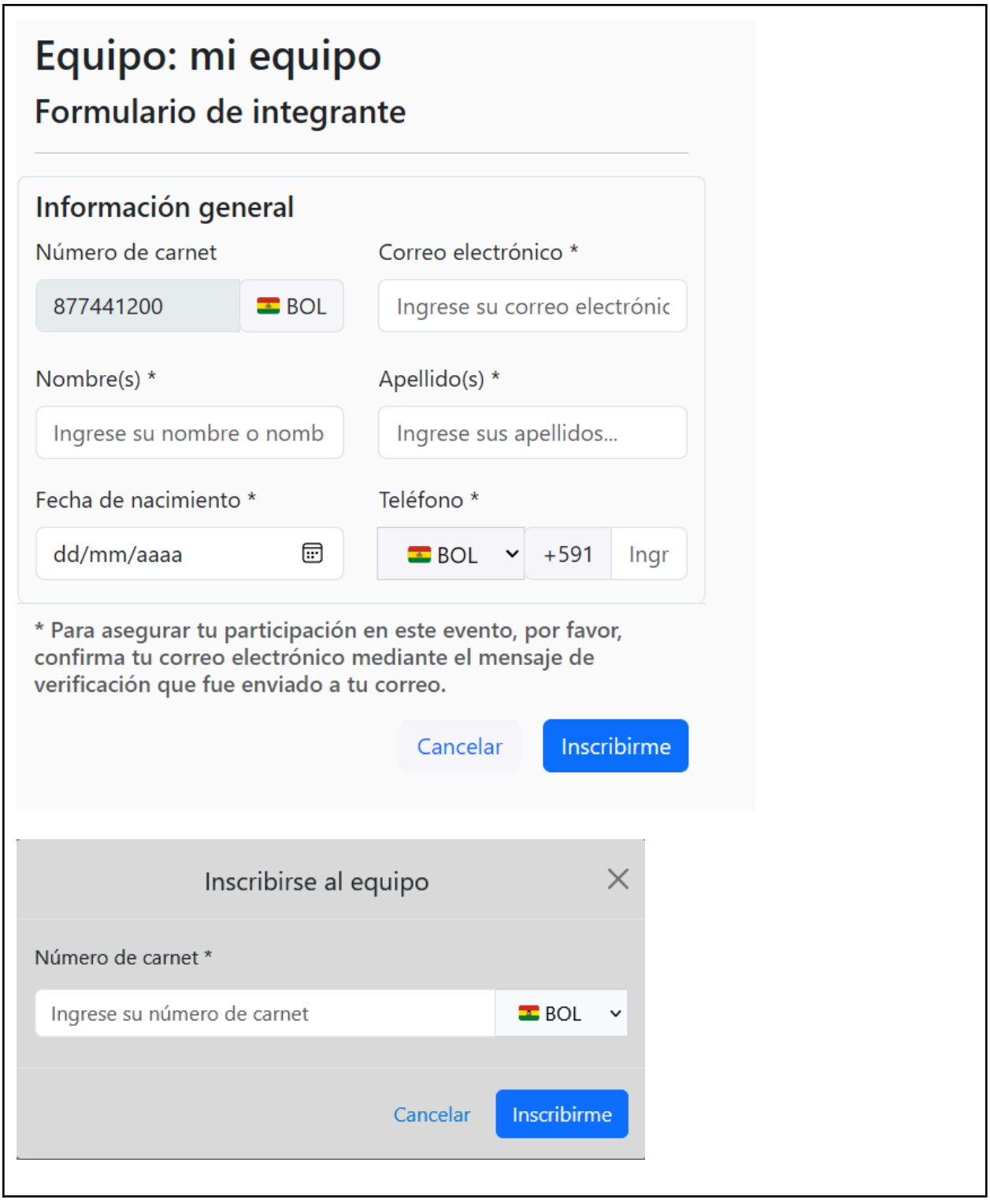
Yo como usuario interesado en participar en un evento por equipos, quiero inscribir mi equipo en un evento, para asegurarme un lugar y recibir información relevante sobre la el evento por equipos.

MockUps






Registar participante al equipo
mi equipo
Rango de participantes por equipo: desde los 2 hasta los 4
Nombres de participante
Apellidos de participante
Correo
Teléfono
Josue Brandon
Rodriguez Blanco
josuebrandon99@gmail.com
77619661
juan marcelo
camacho perez
josuebrandon99@gmail.com
78719661
Participantes
Inscribir participante al equipo


Equipo: mi equipo
Formulario de integrante
Información general
Número de carnet
877441200
BOL
Correo electrónico *
Ingrese su correo electrónic
Nombre(s) *
Ingrese su nombre o nomb
Apellido(s) *
Ingrese sus apellidos...
Fecha de nacimiento *
dd/mm/aaaa
Teléfono *
BOL
+591
Ingr
* Para asegurar tu participación en este evento, por favor, confirma tu correo electrónico mediante el mensaje de verificación que fue enviado a tu correo.
Cancelar
Inscribirme
Inscribirse al equipo
Número de carnet *
Ingrese su número de carnet
BOL
Cancelar
Inscribirme

Criterios de aceptación

1. Al hacer clic en la sección “EVENTOS” en el menú superior, se muestra una tarjeta por cada evento que existe en el sistema.
2. Al hacer clic en la tarjeta de un evento, se muestra una página con toda la información del evento y un botón “Inscribirme”.
3. En la vista del evento se muestra un contador con la cantidad de equipos.
4. Al hacer clic en el botón “Inscribirme” en el evento, se muestra el modal “Registrar equipo” con los campos “Nombre del equipo”, “Correo electrónico” y los botones “Cancelar” e “Inscribir”.
5. El campo “Número de carnet *”, es obligatorio.
6. El campo “Correo electrónico*”, es obligatorio.
7. Si el campo “Correo electrónico*”, no cumple con el formato de un correo electrónico, aparece un mensaje de “Correo inválido.”;
8. Al hacer clic en el botón “Cancelar” en el modal “Registrar equipo”, el modal se cierra.
9. Al hacer clic en el botón “Inscribir” en el modal “Registrar equipo”, si el nombre de equipo nunca se registró en el sistema y el correo electrónico es válido, se envía un mensaje de correo electrónico al correo escrito en el campo “Correo electrónico”, con un código de verificación de validez.
10. Al hacer clic en el botón “Inscribir” en el modal “Registrar equipo”, si el nombre de equipo nunca se registró en el sistema y el correo electrónico es válido, se despliega el campo de “Código de acceso*” y el botón “Reenviar código”.
11. Al hacer clic en el botón “Inscribir” en el modal “Registrar equipo”, si el nombre de equipo esta registrado en el sistema y el correo electrónico no está registrado con el nombre de equipo, se muestra una alerta con el mensaje de “El nombre del equipo ya registrado a este evento.”.
12. Al hacer clic en el botón “Inscribir” en el modal “Registrar equipo”, si el nombre de equipo esta registrado en el sistema y el correo electrónico está registrado con el nombre de equipo, se envía un mensaje de correo electrónico al correo escrito en el campo “Correo electrónico*”, con un código de verificación de validez.
13. Al hacer clic en el botón “Inscribir” en el modal “Registrar equipo”, si el nombre de equipo esta registrado en el sistema y el correo electrónico está registrado con el nombre de equipo, se despliega el campo de “Código de acceso*” y el botón “Reenviar código”.
14. Al desplegar el campo de “Código de acceso” y el botón “Reenviar código”, los campos “Nombre del equipo*” y “Correo electrónico*” se deshabilitan.
15. Si está desplegado el campo de “Código de acceso*” y el botón “Reenviar código”, al hacer clic en el botón “Cancelar” se oculta el campo “Código de acceso*” y el botón “Reenviar código”.
16. Si está desplegado el campo de “Código de acceso*” y el botón “Reenviar código”, al hacer clic en el botón “Cancelar” se habilita los campos “Nombre del equipo*” y “Correo electrónico”.
17. El campo “Código de acceso*” es obligatorio.
18. Al colocar un código incorrecto en el campo “Código de acceso” y hacer clic en el

botón “Inscribir”, se muestra el mensaje de “Código no válido.”.

19. Al colocar el código correcto en el campo “Código de acceso” y hacer clic en el botón “Inscribir”, nos lleva a la vista de “Registrar participante al equipo”.
20. En la vista de “Registrar participantes al equipo” se muestra una tabla con todos los participantes del equipo con los campos “Nombre del participante”, “Apellido del participante”, “Correo” y “Teléfono” y el botón “Inscribir participante al equipo”.
21. Al hacer clic en el botón “Inscribir participante al equipo”, se muestra el modal “Inscribirse al equipo”, con el campo “Número de carnet*”, y los botones “Cancelar” e “Inscribirme”.
22. El campo “Número de carnet*”, es obligatorio.
23. Al hacer clic en el botón “Cancelar” en el modal “Inscribirse al equipo”, el modal se cierra.
24. Al hacer clic en el botón “Inscribirme” en el modal “Inscribirse al evento”, si el número de carnet ingresado nunca fue usado en el sistema, se muestra el “Formulario de integrante”.
25. Al hacer clic en el botón “Inscribirme” en el modal “Inscribirse al equipo”, si el número de carnet ingresado fue usado en el sistema con anterioridad pero no en el equipo, se envía un código de acceso al correo del participante.
26. Al hacer clic en el botón “Inscribirme” en el modal “Inscribirse al evento”, si el número de carnet ingresado fue usado en el sistema con anterioridad pero no en el evento, el botón “Inscribirme” es reemplazado por el botón “Verificar”.
27. Al hacer clic en el botón “Inscribirme” en el modal “Inscribirse al evento”, si el número de carnet ingresado fue usado en el sistema con anterioridad pero no en el evento, se muestra el campo “Código de acceso *” y el botón "Reenviar código ".
28. El campo “Código de acceso *”, es obligatorio.
29. Al hacer clic en el botón “Reenviar código”, se envía un código de acceso al correo del participante.
30. Al hacer clic en el botón “Verificar” en el modal “Inscribirse al evento”, si el código de acceso ingresado es válido, se muestra el “Formulario de integrate al equipo”.
31. Al hacer clic en el botón “Inscribirme”, si el número de carnet ingresado fue usado en el sistema y está registrado en el equipo, se muestra una alerta temporal con el mensaje “El participante ya se encuentra inscrito en este evento”.
32. El “Formulario de integrante”, tiene los siguientes campos: “Número de carnet”, “Correo electrónico *”, “Nombre(s) *”, “Apellido(s) *”, “Fecha de nacimiento *”, “Teléfono *” y los botones “Cancelar” e “Inscribirme”.
33. Si el evento tiene restricción de género, se muestra el selector “Género *”, el cual se encuentra deshabilitado, en el “Formulario de integrante al equipo”
34. Si el evento requiere talla de polera, se muestra el selector “Talla de polera *”, en el “Formulario de integrante”
35. Si el evento tiene restricción en las instituciones admitidas, se muestra el selector “Institución *”, en el “Formulario de integrante”
36. Si el evento tiene restricción en el grado académico admitido, se muestra el selector “Grado académico *”, en el “Formulario de integrante”.

37. Todos los campos en el “Formulario de inscripción” que no estén deshabilitados, son obligatorios.
38. Al hacer clic en el botón “Cancelar” en el “Formulario de integrante”, todos los campos regresan a los valores iniciales.
39. Al hacer clic en el botón “Inscribirme” en el “Formulario de integrante”, si toda la información es válida, se muestra una alerta temporal con el mensaje “Inscrito correctamente, por favor, verifica tu correo.”.
40. Al hacer clic en el botón “Inscribirme” en el “Formulario de integrante”, si toda la información es válida, después de un breve periodo de tiempo se regresa al evento.

7.7.3.2. Código fuente de inscribir equipo a un evento

Controlador de inscribir equipo a un evento

Este controlador, denominado EquipoController, se encarga de manejar diversas operaciones relacionadas con la gestión de equipos en un contexto de eventos. En primer lugar, la función index devuelve todos los equipos existentes. La función store se encarga de almacenar un nuevo equipo en la base de datos, generando un código aleatorio para identificación. La función storeInscribir registra la inscripción de un equipo a un evento específico. El método addIntegrante realiza la incorporación de un nuevo integrante al equipo, gestionando la creación de participantes si no existen y enviando correos de confirmación. Además, este método utiliza otras funciones como updateParticipante, storeIntegrante, y existeParticipante para mantener la coherencia en la base de datos y evitar duplicaciones. El controlador también incluye funciones relacionadas con la verificación de códigos y correos electrónicos, así como el envío de mensajes por correo electrónico. Se emplean clases de correo, como EnviarCodigoEquipo y ConfirmacionEquipo, para enviar información importante a los participantes.

Ruta del archivo: app/Http/Controllers/EquipoController.php

```php
<?php

namespace App\Http\Controllers;

use App\Models\Equipo;
use App\Models\EquipoInscrito;
use App\Models\Integrante;
use App\Models\Participante;
use App\Models\Evento;
use Illuminate\Support\Str;
use Illuminate\Http\Request;
use Illuminate\Database\QueryException;
use Illuminate\Support\Facades\Mail;
use App\Mail\EnviarCodigoEquipo;
use App\Mail\ConfirmacionEquipo;
use App\Mail\EnviarCodigoEquipoParticipante;
use Illuminate\Support\Facades\Log;

class EquipoController extends Controller
{
    /**
     * Muestra todos los equipos.
     */
    public function index()
    {
        $equipos = Equipo::all();
        return $equipos;
    }

    /**
     * Almacena un nuevo equipo en la base de datos.
     */
    public function store(Request $request)
    {
        try {
            $equipo = new Equipo();
            $equipo->nombre = $request->nombre;
            $equipo->correo_general = $request->correo_general;
            $equipo->codigo = Str::random(8);
```

```
            $equipo->save();
            return $equipo;
        } catch (QueryException $e) {
            return ['mensaje' => $e->getMessage(), 'error' => true];
        }
    }

    /**
     * Inscribir un equipo en un evento específico.
     */
    public function storeInscribir(Request $request, $id_equipo)
    {
        try {
            $inscrito = new EquipoInscrito();
            $inscrito->id_equipo = $id_equipo;
            $inscrito->id_evento = $request->id_evento;
            $inscrito->save();
            return $inscrito;
        } catch (QueryException $e) {
            return ['mensaje' => $e->getMessage(), 'error' => true];
        }
    }

    /**
     * Agrega un integrante al equipo y envía confirmación por correo.
     */
    public function addIntegrante(Request $request, $id_equipo)
    {
        try {
            // Verifica si el participante existe pero no ha confirmado su correo.
            $participante = Participante::where('correo_confirmado', 0)
                ->where('ci', $request->ci)
                ->first();

            if ($participante) {
                $participante = $this->updateParticipante($request, $participante->id);
                $this->storeIntegrante($request, $id_equipo, $participante->id);
                $equipo = Equipo::find($id_equipo);
                $evento = Evento::find($request->id_evento);
```

```
                Mail::to($request->correo)->send(new ConfirmacionEquipo($equipo,
                $evento, $participante));
                return ['mensaje' => 'Inscrito correctamente, por favor,
                verifica tu correo.'];
            }

            // Si el participante existe y ya está confirmado, solo se agrega como integrante.
            if ($request->id_participante) {
                $this->storeIntegrante($request, $id_equipo, $request->id_participante);
                return ['mensaje' => 'Inscrito correctamente.', 'error' => false];
            } else {
                // Si el participante no existe, se crea uno nuevo y se agrega como integrante.
                $participante = new Participante();
                $participante->ci = $request->ci;
                $participante->nombres = $request->nombres;
                $participante->apellidos = $request->apellidos;
                $participante->correo = $request->correo;
                $participante->codigo_telefono = $request->codigo_telefono;
                $participante->telefono = $request->telefono;
                $participante->fecha_nacimiento = $request->fecha_nacimiento;
                $participante->pais = $request->pais;
                $participante->codigo = Str::random(8);
                $participante->save();
                $this->storeIntegrante($request, $id_equipo, $participante->id);
                $equipo = Equipo::find($id_equipo);
                $evento = Evento::find($request->id_evento);
                Mail::to($request->correo)->send(new ConfirmacionEquipo($equipo, $evento,
                $participante));
                return ['mensaje' => 'Inscrito correctamente, por favor, verifica tu correo.'];
            }
        } catch (QueryException $e) {
            return ['mensaje' => $e->getMessage(), 'error' => true];
        }
    }
public function storeIntegrante(Request $request, $id_equipo, $id_participante)
    {
        try {
            $integrante = new Integrante();
            $integrante->institucion = $request->institucion;
```

```
            $integrante->grado_academico = $request->grado_academico;
            $integrante->genero = $request->genero;
            $integrante->talla = $request->talla;
            $integrante->id_participante = $id_participante;
            $integrante->id_equipo = $id_equipo;
            $integrante->save();
            return $integrante;
        } catch (QueryException $e) {
            return ['mensaje' => $e->getMessage(), 'error' => true];
        }
    }

    public function updateParticipante(Request $request, $id)
    {
        try {
            $participante = Participante::find($id);
            $participante->ci                = $request->ci;
            $participante->nombres           = $request->nombres;
            $participante->apellidos         = $request->apellidos;
            $participante->correo            = $request->correo;
            $participante->codigo_telefono   = $request->codigo_telefono;
            $participante->telefono          = $request->telefono;
            $participante->fecha_nacimiento  = $request->fecha_nacimiento;
            $participante->pais              = $request->pais;
            $participante->codigo            = Str::random(8);
            $participante->save();
            return $participante;
        } catch (QueryException $e) {
            return ['mensaje' => $e->getMessage(), 'error' => true];
        }
    }
    public function existeParticipante($ci, $id_equipo, $id_evento)
    {
        $participante = Participante::where('ci', $ci)
            ->where('correo_confirmado', 1)
            ->first();

        $equipoInscrito = EquipoInscrito::where('id_evento', $id_evento)
            ->where('id_equipo', $id_equipo)
```

```
            ->whereHas('equipos', function ($q) use ($ci, $id_evento) {
                $q->whereHas('integrantes', function ($q) use ($ci, $id_evento) {
                    $q->whereHas('participantes', function ($q) use ($ci, $id_evento) {
                        $q->where('ci', $ci);
                        $q->where('id_evento', $id_evento);
                        $q->where('correo_confirmado', 1);
                    });
                });
            })
            ->first();
        if ($equipoInscrito) {
            return [
                'error' => true,
                'inscrito' => true,
                'mensaje' => 'El participante ya se encuentra inscrito en este evento.'
            ];
        }
        if ($participante) {
            $integrante = Integrante::where('id_participante', $participante->id)->where(
              'id_equipo', $id_equipo)->first();
            if ($integrante) {
                return [
                    'inscrito' => false,
                    'integrante' => true,
                    'mensaje' => 'Participante ya pertenece a este equipo.'
                ];
            } else {
                return [
                    'integrante' => false,
                    'inscrito' => false,
                    'participante' => $participante
                ];
            }
        } else {
            return ['error' => false];
        }
    }

    public function inscribirEquipoEvento(Request $request)
```

```
{
    try {
        $mensaje = "Equipo inscrito correctamente.";
        $equipo = Equipo::where('correo_verificado', 0)->find($request->id_equipo);

        if ($equipo) {
            $equipo = $this->update($request, $request->id_equipo);
            $equipoInscrito = EquipoInscrito::where('id_evento', $request->id_evento)
                ->where('id_equipo', $equipo->id)
                ->first();
            if (!$equipoInscrito) {
                $this->storeInscribir($request, $equipo->id);
            }
            $evento = Evento::find($request->id_evento);
            Mail::to($equipo->correo_general)->send(new EnviarCodigoEquipo($equipo, $evento));
            return ['mensaje' => $mensaje, 'error' => false, 'equipo' => $equipo];
        }
        if ($request->id_equipo) {
            $this->storeInscribir($request, $request->id_equipo);
            $equipo = Equipo::find($request->id_equipo);
            $evento = Evento::find($request->id_evento);
            Mail::to($equipo->correo_general)->send(new EnviarCodigoEquipo($equipo, $evento));
            return ['mensaje' => $mensaje, 'error' => false, 'equipo' => $equipo];
        } else {
            $equipo = $this->store($request);
            $this->storeInscribir($request, $equipo->id);
            $evento = Evento::find($request->id_evento);
            Mail::to($equipo->correo_general)->send(new EnviarCodigoEquipo($equipo, $evento));
            return ['mensaje' => $mensaje, 'error' => false, 'equipo' => $equipo];
        }
    } catch (QueryException $e) {
        return ['mensaje' => $e->getMessage(), 'error' => true];
    }
}

public function existeEquipo(Request $request)
{
    try {
        $equipo = Equipo::where('nombre', $request->nombre)
```

```
            ->where('correo_general', $request->correo_general)
            ->first();

        $repetido = EquipoInscrito::where('id_evento', $request->id_evento)->with('equipos'
            ->whereHas('equipos', function ($q) use ($request) {
                $q->where('nombre', $request->nombre)
                    ->where('correo_verificado', 1);
            })->first();
        if ($equipo) {
            $inscrito = EquipoInscrito::where('id_evento', $request->id_evento)
                ->where('id_equipo', $equipo->id)
                ->wherehas('equipos', function ($q) use ($equipo) {
                    $q->where('nombre', $equipo->nombre)
                        ->where('correo_verificado', 1);
                })
                ->first();
        } else {
            $esRepetido = $repetido ? true : false;
            $inscrito = null;
        }
        if ($inscrito) {
            return [
                'inscrito' => true,
                'mensaje' => ['mensaje' => 'El equipo ya ha sido inscrito a este evento',
                'error' => true],
                'equipo' => $equipo
            ];
        } else {
            if (!$equipo) {
                return [
                    'inscrito' => false,
                    'error' => $esRepetido,
                    'Mensaje' => [
                        "mensaje" => $esRepetido ? " El nombre del equipo ya registrado
                        a este evento." : "",
                        "error" => $esRepetido
                    ],
                ];
            } else {
                return [
```

```
                    'inscrito' => false,
                    'equipo' => $equipo,
                    'Mensaje' => [
                        "mensaje" => "",
                        "error" => false
                    ]
                ];
            }
        }
    } catch (QueryException $e) {
        return ['mensaje' => $e->getMessage(), 'error' => true];
    }
}

public function update(Request $request, $id)
{
    try {
        $equipo = Equipo::find($id);
        $equipo->nombre = $request->nombre;
        $equipo->correo_general = $request->correo_general;
        $equipo->save();
        return $equipo;
    } catch (QueryException $e) {
        return ['mensaje' => $e->getMessage(), 'error' => true];
    }
}

public function mostrarEquipo($codigo, $id)
{
    $equipo = Equipo::with([
        'integrantes' => function ($q) {
            $q->whereHas('participantes', function ($q) {
                $q->where('correo_confirmado', 1);
            });
        }
    , 'integrantes.participantes'])->whereHas('integrantes', function ($q) {
        $q->whereHas('participantes', function ($q) {
            $q->where('correo_confirmado', 1);
        });
    })
```

```
            ->where('codigo', $codigo)
            ->first();
        if(!$equipo){
            $equipo = Equipo::where('codigo', $codigo)->first();
        }
        $evento = Evento::find($id);
        return view('inscripciones.tablaEquipo', ['equipo' => $equipo, 'evento' => $evento]);
    }

    public function enviarCorreo($id_equipo, $id_evento)
    {
        $equipo = Equipo::findorfail($id_equipo);
        $evento = Evento::findorfail($id_evento);
        Mail::to($equipo->correo_general)->locale('es')
            ->send(new EnviarCodigoEquipo($equipo, $evento));
    }

    public function verificarCodigo(Request $request, $id)
    {
        $equipo = Equipo::findorfail($id);
        if ($equipo->codigo == $request->codigo) {
            if ($equipo->correo_verificado == 0) {
                $equipo->correo_verificado = 1;
                $equipo->save();
                $this->borrarInscritosEquipo($equipo->id, $request->id_evento);
            }
            return ['error' => false, 'mensaje' => 'Código verificado correctamente.'];
        } else {
            return ['error' => true, 'mensaje' => 'El código de equipo no coincide.'];
        }
    }

    public function verificarCorreo($id_evento, $id_equipo, $codigo)
    {
        $participante = Participante::where('codigo', $codigo)
            ->where('correo_confirmado', 0)
            ->first();
        $equipo = Equipo::findorfail($id_equipo);
        if ($participante) {
            $participante->correo_confirmado = 1;
```

```
            $participante->save();
            $this->borrarInscritosEquipo($equipo->id, $id_evento);
            $verificado = (object)['error' => false, 'mensaje' => 'Correo verificado correctamente.'];
        } else {
            $verificado =  (object)['error' => true, 'mensaje' => 'El correo ya ha sido verificado.'];
        }
        return view("inscripciones.confirmacionCorreo", ['verificado' => $verificado]);
    }

    public function borrarInscritosEquipo($id_equipo, $id_evento)
    {
        try {
            $inscrito = EquipoInscrito::where('id_evento', '<>', $id_evento)
                ->where('id_equipo', $id_equipo)
                ->delete();
            return $inscrito;
        } catch (QueryException $e) {
            Log::error($e->getMessage());
            return ['mensaje' => $e->getMessage(), 'error' => true];
        }
    }

    public function enviarCodigoCorreoParticipante($id_participante, $id_equipo, $id_evento)
    {
        try {
            $participante = Participante::find($id_participante);
            $equipo = Equipo::find($id_equipo);
            $evento = Evento::find($id_evento);
            Mail::to($participante->correo)
                ->send(new EnviarCodigoEquipoParticipante($participante, $equipo, $evento));
        } catch (QueryException $e) {
            Log::error($e->getMessage());
            return ['mensaje' => $e->getMessage(), 'error' => true];
        }
    }
```

Código JavaScript de inscribir equipo a un evento

La lógica se implementa en un entorno de modal, donde los usuarios ingresan información como el nombre del equipo y el correo electrónico. La interfaz responde a eventos como cambios en los campos de entrada y clics en botones de confirmación o cancelación. La lógica de validación se extiende a la verificación de códigos, la existencia de equipos y la gestión de correos electrónicos. Se emplea la librería Axios para realizar solicitudes HTTP al servidor. Además, el código está estructurado de manera modular, con funciones específicas para manejar diferentes aspectos del flujo de trabajo.

Ruta del archivo: public/js/Inscripciones/inscribirEquipo.js

```
// Declaración de variables que representan elementos del DOM
const contenido1 = $("#collapGmail"); // Contenido del primer colapso en el modal
const contenido2 = $("#collapseVerificaiconGmail"); // Contenido del segundo colapso en el modal
const modal = document.getElementById("modal-inscribir"); // Elemento modal
const nombreInput = document.getElementById("nombreEquipo"); // Input para el nombre del equipo
const correoInput = document.getElementById('inputEmail'); // Input para el correo del equipo
const validarGmail1 = document.getElementById("codigoValidacion"); // Elemento para la validación del códi
const contenedorCorreo = document.getElementById('contenedorCorreo'); // Contenedor del correo en el modal
const buttonConfirmacion = document.getElementById('button-equipo-confirmacion'); // Botón de confirmació
const buttonCancelacion = document.getElementById('button-equipo-cancelar'); // Botón de cancelación
const codigoInput1 = document.getElementById('codigo1'); // Input para el código de validación
const form = document.getElementById('inscribirEquipo'); // Formulario de inscripción
const mensaCorreo = document.getElementById('mensajeErrorCorreo'); // Mensaje de error para el correo
const mensajeNombre = document.getElementById('mensajeErrorNombre'); // Mensaje de error para el nombre
// Variables de control y almacenamiento de datos
let crear = true; // Variable para controlar la creación de equipos
let Datos; // Variable para almacenar datos
let id_equipo; // ID del equipo
let id_evento; // ID del evento
// Funciones relacionadas con eventos del botón
const ingresarNombreEquipo = () => {
    // Disparar eventos de cambio en los inputs
    nombreInput.dispatchEvent(new Event("change"));
    correoInput.dispatchEvent(new Event("change"));

    // Validar si los inputs son válidos
    if (nombreInput.classList.contains("is-valid") && correoInput.classList.contains("is-valid")) {
        // Verificar si se debe validar el código de Gmail
        if (validarGmail1.style.display == 'none') {
            registrarCorreo();
        } else {
            // Validar el código de Gmail antes de crear el equipo
            codigoInput1.dispatchEvent(new Event("change"));
            if (codigoInput1.classList.contains("is-valid")) {
                crearEquipo();
            }
        }
    }
}
```

```
// Función para crear un equipo
const crearEquipo = async () => {
    // Crear FormData con el código y el ID del evento
    let formData = new FormData();
    formData.append("codigo", codigoInput1.value);
    formData.append("id_evento", id_evento);

    // Verificar el código del equipo
    if (crear) {
        const response = await axios.post("/api/equipo/verificarCodigo/" + id_equipo, formData)
            .catch(error => {
                // Manejar errores
                console.error('Error al obtener datos:', error);
            });

        // Procesar la respuesta del servidor
        if (response.data.error) {
            esValido(codigoInput1, false);
            mensaCorreo.innerText = "Código incorrecto.";
            participanteYaInscrito(response);
        } else {
            localStorage.setItem("codigo", codigoInput1.value);
            console.log(localStorage.getItem('codigo'));
            window.location.href = "/eventos/tabla-equipo/" + codigoInput1.value + "/" +
            nombreInput.getAttribute("evento_id");
        }
    }
}

// Función para cancelar la creación de equipo o regresar en la validación de correo
const cancelarEquipo = () => {
    if (!nombreInput.disabled) {
        $('#modal-inscribir').modal("hide");
    } else {
        correoInput.disabled = false;
        nombreInput.disabled = false;
        atrasCodigo();
    }
}
```

```
// Función para registrar el correo del equipo y deshabilitar los campos correspondientes
const registrarCorreo = async () => {
    await registrarEquipo();
    if (nombreInput.classList.contains("is-valid")) {
        validarGmail1.style.display = 'block';
        nombreInput.disabled = true;
        correoInput.disabled = true;
    }
}

// Función para verificar el código de Gmail
const verificarCodigo = () => {
    codigoInput1.dispatchEvent(new Event("change"));
    if (codigoInput1.classList.contains("is-valid")) {
        // Realizar acciones adicionales si es necesario
    }
}

// Función para registrar un equipo y gestionar la respuesta del servidor
const registrarEquipo = async () => {
    var formData = new FormData();

    // Agregar datos al FormData
    formData.append('nombre', nombreInput.value);
    formData.append('correo_general', correoInput.value);
    formData.append('id_evento', nombreInput.getAttribute("evento_id"));
    id_evento = nombreInput.getAttribute("evento_id");

    // Realizar la petición al servidor para verificar la existencia del equipo
    const response = await axios.post('/api/equipo/existe', formData)
        .catch(function (error) {
            console.error('Error en la petición:', error);
        });

    // Procesar la respuesta del servidor
    if (response.data.inscrito) {
        id_equipo = response.data.equipo.id;
        enviarCorreo(id_equipo, formData.get("id_evento"));
        formData.append('id_equipo', response.data.equipo.id);
```

```
            validarGmail1.style.display = 'block';
            nombreInput.disabled = true;
            correoInput.disabled = true;
            enviarCorreo(id_equipo, id_evento);
        } else {
            if (response.data.equipo) {
                id_equipo = response.data.equipo.id;
                formData.append('id_equipo', response.data.equipo.id)
            }
            if (response.data.Mensaje.error) {
                mostrarAlerta(
                    "Éxito",
                    response.data.Mensaje.mensaje,
                    response.data.Mensaje.error ? "danger" : "success"
                );
                esValido(correoInput, false);
                esValido(nombreInput, false);
                mensajeNombre.innerText = "Nombre ya está registrado.";
                mensaCorreo.innerHTML = "Correo no válido.";

            } else {
                registrarEquipoEvento(formData);
            }
        }
}

// Función para enviar el correo de confirmación al equipo
const enviarCorreo = (id_equipo, id_evento) => {
    axios.post('/api/equipo/enviarCorreo/' + id_equipo + '/' + id_evento)
        .then(response => {
            // Manejar la respuesta si es necesario
        })
        .catch(error => {
            // Manejar errores
            console.error('Error al obtener datos:', error);
        });
}
```

```
// Función para retroceder en la validación del correo
const atrasCorreo = () => {
    contenido1.collapse("hide");
    removerValidacion(correoInput);
    nombreInput.disabled = false;
}

// Función para retroceder en la validación del código de Gmail
const atrasCodigo = () => {
    removerValidacion(codigoInput1);
    validarGmail1.style.display = 'none';
    contenedorCorreo.style.display = "block";
}

// Función para registrar un equipo en un evento
const registrarEquipoEvento = (formData) => {
    axios.post("/api/equipo/inscribirEquipo", formData)
        .then(function (response) {
            if (response.data.equipo) {
                id_equipo = response.data.equipo.id;
            }
        })
        .catch(function (error) {
            console.error('Error en la petición:', error);
        });
}

// Función para reenviar el código de confirmación por correo
const reEnviarCodigo = async () => {
    let res = await enviarCorreo(id_equipo, id_evento);
}

// Función para enviar el código de acceso por correo
const enviarCodigoAcceso = async () => {
    let data = await axios.post("/api/participante/enviarCodigo/" + idEvento + "/" + idParticipante).then
        return response.data;
    });
    return data;
};
```

```
// Función para manejar el caso de participante ya inscrito
const participanteYaInscrito = (response) => {
    mostrarAlerta(
        "Éxito",
        response.data.Mensaje.mensaje,
        response.data.Mensaje.error ? "danger" : "success"
    );
};

// Funciones de validación de inputs
const inputRequired = (input) => {
    if (input.value == "") {
        esValido(input, false);
    } else {
        esValido(input, true);
    }
}

const esValido = (componente, bandera) => {
    // Validar la clase CSS del componente según la bandera
    if (bandera) {
        componente.classList.remove("is-invalid");
        componente.classList.add("is-valid");
    } else {
        componente.classList.remove("is-valid");
        componente.classList.add("is-invalid");
    }
}

const inputEmail = (input) => {
    // Validar la validez del correo y actualizar la clase CSS
    if (input.validity.valid) {
        esValido(input, true);
    } else {
        esValido(input, false);
        mensaCorreo.innerText = "Correo inválido.";
    }
}
```

```
const removerValidacion = (input) => {
    // Eliminar clases de validación y limpiar el valor del input
    input.value = "";
    input.classList.remove("is-valid");
    input.classList.remove("is-invalid");
}

// Eventos asociados al modal y los inputs
modal.addEventListener('hidden.bs.modal', function (event) {
    removerValidacion(nombreInput);
    nombreInput.disabled = false;
    removerValidacion(correoInput);
    correoInput.disabled = false;
    validarGmail1.style.display = 'none';
    removerValidacion(codigoInput1);
});

nombreInput.addEventListener("change", () => {
    // Validar el nombre del equipo cuando cambia
    inputRequired(nombreInput);
});

correoInput.addEventListener("change", () => {
    // Validar el correo del equipo cuando cambia
    inputRequired(correoInput);
    inputEmail(correoInput);
});

codigoInput1.addEventListener("change", () => {
    // Validar el código de confirmación cuando cambia
    inputRequired(codigoInput1);
});
```

Vista de inscribir equipo a un evento

La vista incluye un formulario que recopila información general del participante, como número de carnet, correo electrónico, nombres, apellidos, fecha de nacimiento y teléfono. Además, se manejan aspectos específicos del evento, como género, talla de polera, institución y grado académico, presentándolos en secciones correspondientes del formulario. La visibilidad de algunos campos depende de la configuración del evento, permitiendo una adaptabilidad dinámica a diferentes requisitos. El formulario también muestra mensajes informativos sobre el costo de la inscripción y la necesidad de confirmar el correo electrónico para asegurar la participación en el evento. Además, se incorporan botones de acción, como "Cancelar" e "Inscribirme", para proporcionar una experiencia de usuario interactiva. El código hace uso de hojas de estilo personalizadas, scripts para funcionalidades específicas y bibliotecas externas para mejorar la presentación y la experiencia del usuario. En resumen, este código proporciona una interfaz completa y adaptable para la inscripción de participantes en eventos, con un diseño claro y funcionalidades integradas.

Ruta del archivo: resources/views/inscripciones/participanteEquipo.blade.php

```
@extends('layouts.app')

@section('content')
    <div class="container mb-3" id="formularioInscripcionEvento">
        <div class="d-flex justify-content-center">
            <div class="col-md-8">
                <!-- Encabezado de la vista -->
                <h2 id="nombreEvento">Equipo: {{ $equipo->nombre }}</h2>
                <h4>Formulario de integrante</h4>
                <hr>
                <!-- Formulario de inscripción de participante -->
                <form class="needs-validation" novalidate id="formInscripcionParticipante">
                    <!-- Campos ocultos para identificación -->
                    <input type="hidden" id="idParticipante" value="{{ $participante->id }}">
                    <!-- Sección de información general -->
                    <div class="row border border-bottom-0 rounded-top">
                        <h5 class="mt-2">Información general</h5>
                        <!-- Campo de número de carnet -->
                        <div class="col-md-6">
                            <!-- ... -->
                        </div>
                        <!-- Campo de correo electrónico -->
                        <div class="col-md-6">
                            <!-- ... -->
                        </div>
                    </div>
                    <!-- Sección de nombres y apellidos -->
                    <div class="row border border-top-0 border-bottom-0">
                        <!-- Campo de nombres -->
                        <div class="col-md-6">
                            <!-- ... -->
                        </div>
                        <!-- Campo de apellidos -->
                        <div class="col-md-6">
                            <!-- ... -->
                        </div>
                    </div>
                    <!-- Sección de fecha de nacimiento y teléfono -->
                    <div class="row border border-top-0 border-bottom-0 rounded-bottom">
                        <!-- Campo de fecha de nacimiento -->
                        <div class="col-md-6">
                            <!-- ... -->
                        </div>
                        <!-- Campo de teléfono -->
                        <div class="col-md-6">
                            <!-- ... -->
                        </div>
                    </div>
                    <!-- Sección de género y talla (si aplican) -->
                    <div class="row border border-top-0 rounded-bottom">
                        <!-- Campo de género -->
                        <div class="col-md-6" {{ $evento->genero != null ? '' : 'hidden' }}>
                            <!-- ... -->
                        </div>
                        <!-- Campo de talla de polera -->
                        <div class="col-md-6" {{ $evento->talla == 'on' ? '' : 'hidden' }}>
                            <!-- ... -->
                        </div>
                    </div>
```

```
                    <!-- Sección de información académica (si aplica) -->
                    <div class="row border rounded mt-3"
                        {{ $evento->institucion != null || $evento->grado_academico != null ? '' : 'hidden' }}>
                        <h5 class="mt-2">Información académica</h5>
                        <!-- Campo de institución (si aplica) -->
                        <div class="col-md-6" {{ $evento->institucion != null ? '' : 'hidden' }}>
                            <!-- ... -->
                        </div>
                        <!-- Campo de grado académico (si aplica) -->
                        <div class="col-md-6 mb-3" {{ $evento->grado_academico != null ? '' : 'hidden' }}>
                            <!-- ... -->
                        </div>
                    </div>
                </form>
                <!-- Mensajes informativos adicionales -->
                @if ($evento->precio_inscripcion != null)
                    <!-- Mensaje de costo de inscripción -->
                    <h6 class="text-muted mt-2"><b>Nota:</b> La inscripción a este evento tiene un costo de
                        {{ $evento->precio_inscripcion }} Bs.</h6>
                @endif
                @if (!$participante->nombres)
                    <!-- Mensaje de confirmación de correo electrónico -->
                    <h6 class="text-muted mt-2">
                        <b>*</b> Para asegurar tu participación en este evento, por favor, confirma tu correo
                        electrónico mediante el mensaje de verificación que fue enviado a tu correo.
                    </h6>
                @endif
                <!-- Botones de acción -->
                <div class="d-flex justify-content-end mt-3">
                    <!-- Botón de cancelar -->
                    <button type="button" class="btn btn-light text-primary me-3" onclick="resetForm()">
                        Cancelar
                    </button>
                    <!-- Botón de inscribirse -->
                    <button type="button" class="btn btn-primary" onclick="validarInputs()">
                        Inscribirme
                    </button>
                </div>
            </div>
        </div>
    </div>
    <!-- Enlaces y scripts adicionales -->
    <link href="{{ asset('css/participante.css') }}" rel="stylesheet">
    <script type="module" defer>
        import {
            polyfillCountryFlagEmojis
        } from "https://cdn.skypack.dev/country-flag-emoji-polyfill";
        polyfillCountryFlagEmojis();
    </script>
    <style>
        * {
            font-family: "Twemoji Country Flags", "Segoe UI", "Helvetica Neue", "Noto Sans", "Liberation Sans",
            "Apple Color Emoji", "Segoe UI Emoji", "Segoe UI Symbol", "Noto Color Emoji";
        }
    </style>
    <script src="{{ asset('js/Inscripciones/codPaises.js') }}" defer></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/dayjs/1.11.10/dayjs.min.js"></script>
    <script src="{{ asset('js/Inscripciones/participanteEquipo.js') }}" defer></script>
@endsection
```

7.7.3.3. Interfaz de usuario de inscribir equipo a un evento

En la interfaz, se incluyen elementos como formularios, botones y mensajes informativos para facilitar la interacción del usuario con el sistema de inscripción. Algunos de los elementos clave incluyen:

Formulario de Inscripción:

Se presenta un formulario con campos esenciales, como el nombre del equipo, la dirección de correo electrónico y un código de validación.

Los campos se encuentran distribuidos de manera lógica y están etiquetados para una comprensión clara.

Validación de Correo Electrónico:

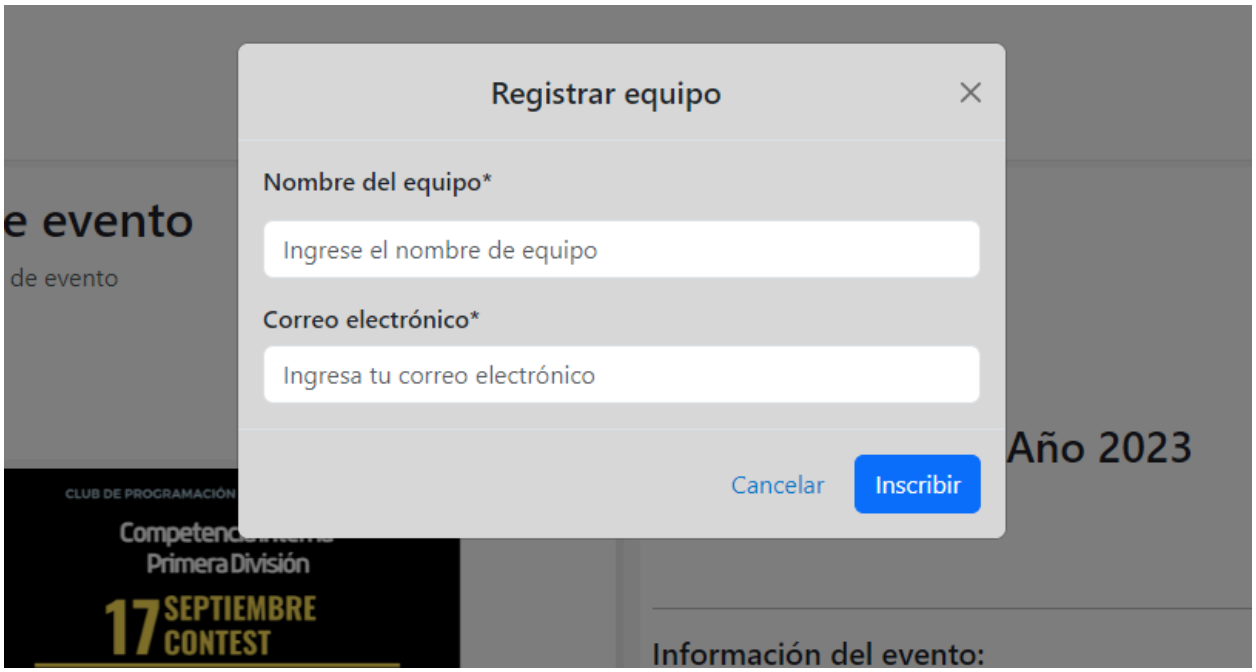
La interfaz incluye una sección para validar el correo electrónico, donde se muestra un código de verificación después de registrar la información del equipo.

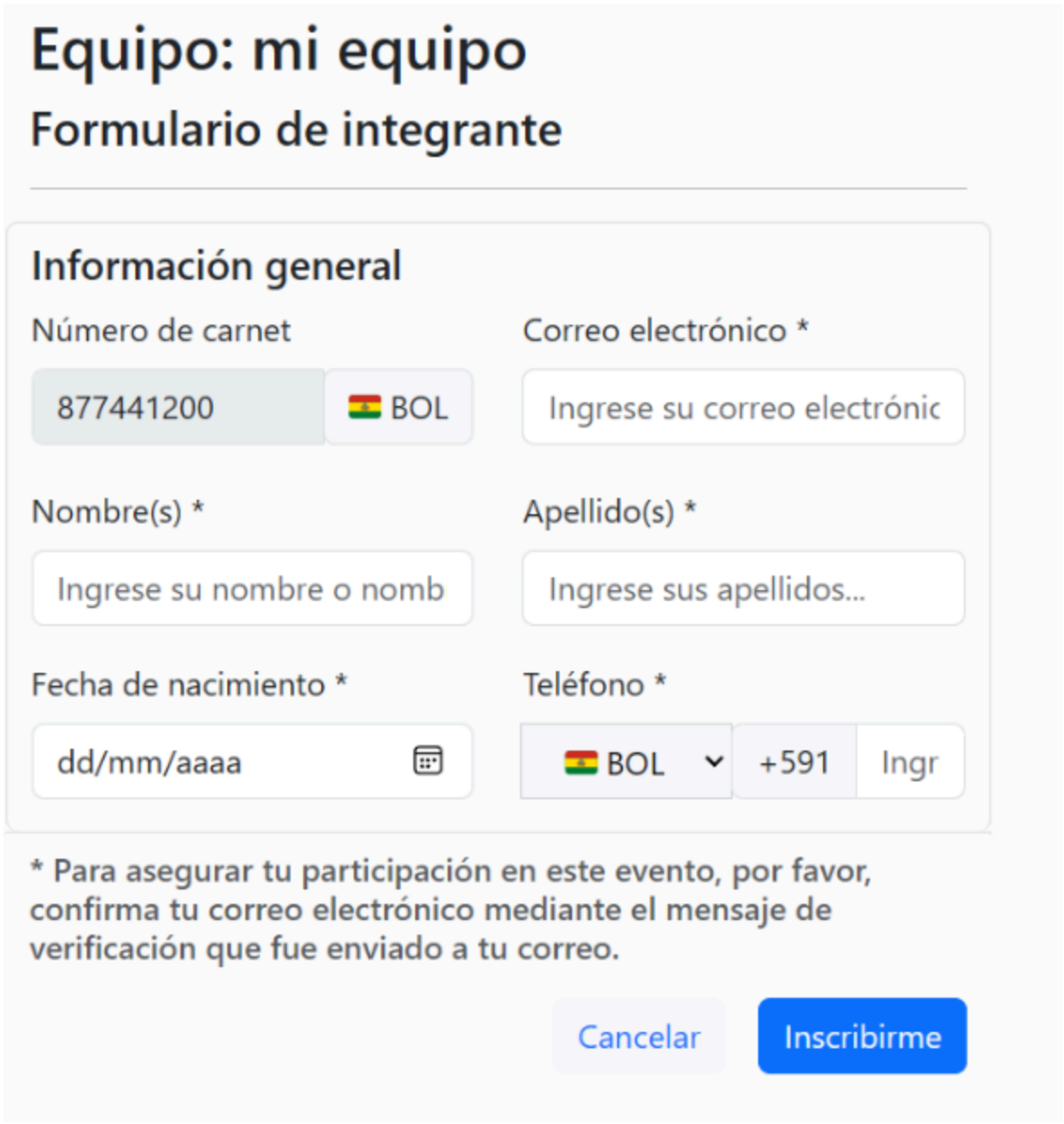
Manejo de Errores:

Se implementan mensajes de error para notificar al usuario sobre posibles problemas, como un código de validación incorrecto.

Botones de Acción:

Se incorporan botones como "Confirmar" y "Cancelar" para proporcionar opciones claras de acción al usuario.





8. Diagrama de paquetes

En la implementación de Laravel se construyen varios módulos, el principal diagrama muestra cómo está la relación entre los paquetes de la aplicación.

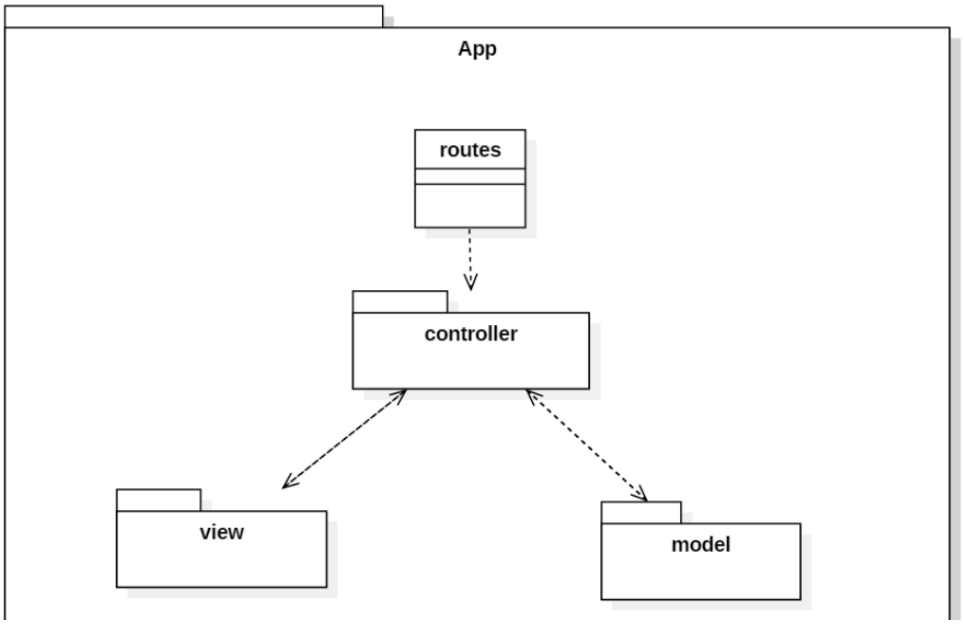


Figura 26. Diagrama de paquetes principales

9. Diagrama del paquete controller

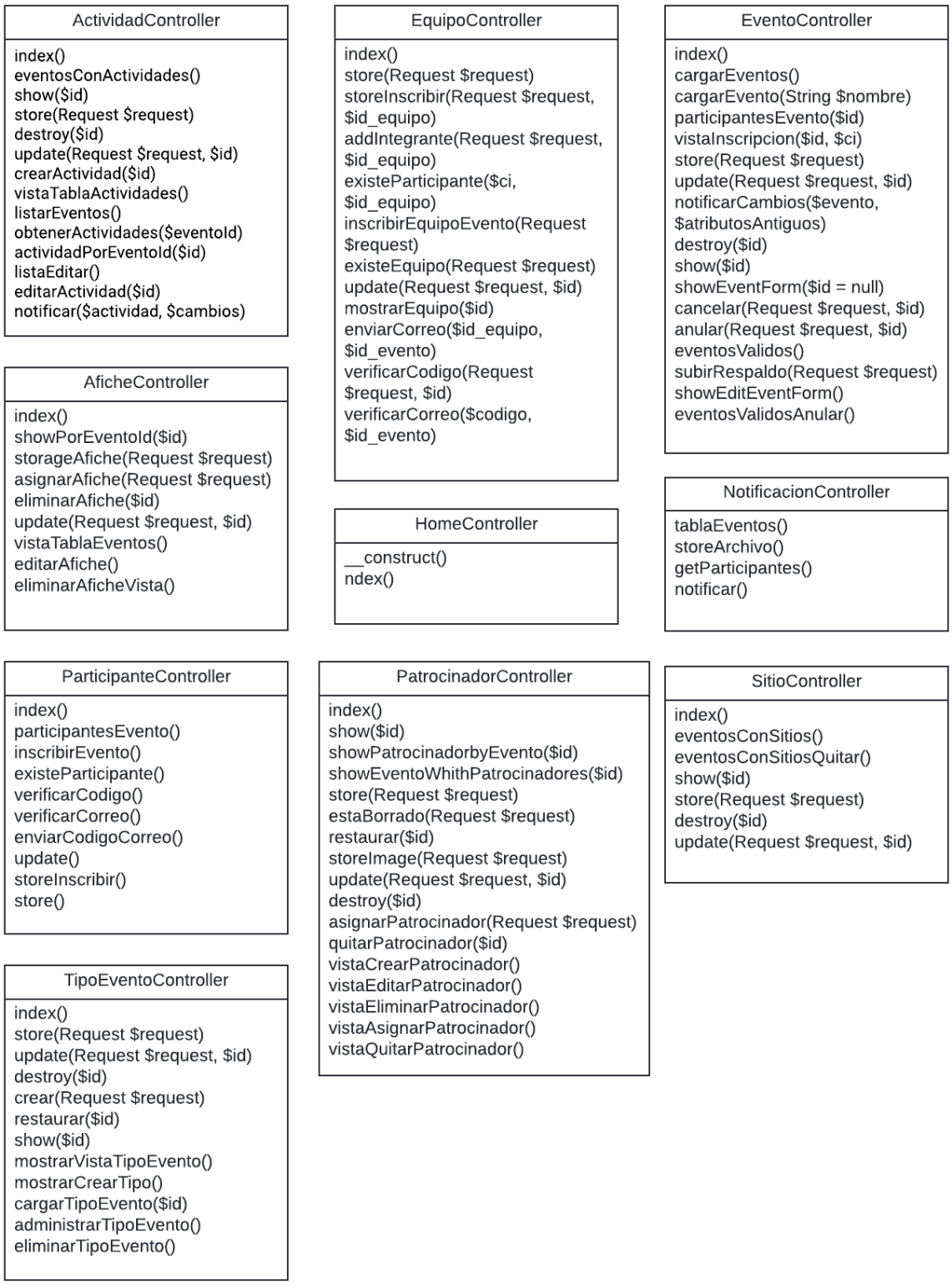


Figura 27. Diagrama de controladores del proyecto ICPC-UMSS

Información de Soporte

Infinity Code Ltda. Grupo Empresa
Calle Santa Ines
Cochabamba, Bolivia
Teléfonos: (+591) 77419661
Pagina Web: www.infinitycode.com
Correo de Soporte: infinity.code.ltda@gmail.com