UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
GRADUADO EN INGENIERÍA INFORMÁTICA

APLICACIÓN WEB DE GESTIÓN DE EVENTOS DEPORTIVOS

SPORT EVENT MANAGEMENT WEB APPLICATION

Realizado por
ÁLVARO SALAS CRIADO

Tutorizado por
SERGIO GÁLVEZ ROJAS

Departamento
LENGUAJES Y CIENCIAS DE LA COMPUTACIÓN

UNIVERSIDAD DE MÁLAGA

MÁLAGA, SEPTIEMBRE DE 2024

Fecha defensa: septiembre de 2024

Resumen

Este trabajo de fin de grado se enfoca en el desarrollo de una aplicación web orientada a la gestión de eventos deportivos, con la finalidad de centralizar y simplificar el proceso completo de creación, promoción y administración de estos eventos en una plataforma única. La aplicación está diseñada para que las empresas puedan anunciarse de manera efectiva, maximizando su alcance y visibilidad en el mercado.
Una de las funcionalidades clave de la plataforma es su sistema de venta de entradas. Además, se ha priorizado el diseño de la interfaz de usuario, asegurando que sea simple y fácil de usar, garantizando una experiencia accesible e intuitiva para todos los usuarios.
Durante el desarrollo del proyecto, se adoptó la metodología ágil Scrum, lo cual permitió una gestión flexible, facilitando la adaptación a los cambios y promoviendo la mejora continua en cada fase. Esto fue esencial para asegurar que el producto final no solo cumpliera con los objetivos iniciales, sino que también se adaptara a nuevas necesidades surgidas durante el proceso.
En cuanto a las tecnologías empleadas, se optó por Node.js y Express.js para el backend, logrando un entorno rápido y escalable. Para la gestión de la base de datos, se utilizó Sequelize como ORM y MySQL como sistema de gestión, garantizando un manejo eficiente y seguro de los datos. Entre las principales funcionalidades se encuentran la gestión de eventos, la subida de imágenes, la emisión de notificaciones automáticas y un sistema de autenticación para asegurar la seguridad de los datos y las operaciones.

Palabras clave: Gestión de eventos deportivos, Node.js, Express.js, MySQL, Scrum, autenticación, subida de fotos, notificaciones, venta de entradas.

Abstract

This final degree project focuses on the development of a web application aimed at managing sports events, with the purpose of centralizing and simplifying the entire process of creating, promoting, and managing these events on a single platform. The application is designed to allow companies to advertise effectively, maximizing their reach and visibility in the market.

One of the key features of the platform is its ticket sales system. Additionally, the design of the user interface has been prioritized, ensuring that it is simple and easy to use, providing an accessible and intuitive experience for all users.

During the development of the project, the agile Scrum methodology was adopted, which allowed for flexible management, facilitating adaptation to changes and promoting continuous improvement in each phase. This was essential to ensure that the final product not only met the initial objectives but also adapted to new needs that arose during the process.

As for the technologies used, Node.js and Express.js were chosen for the backend, achieving a fast and scalable environment. For database management, Sequelize was used as the ORM and MySQL as the database management system, ensuring efficient and secure data handling. Among the main functionalities are event management, image uploading, automatic notification sending, and an authentication system to ensure the security of data and operations.

Keywords: Sports event management, Node.js, Express.js, MySQL, Scrum, authentication, photo uploads, notifications, ticket sales.

Índice

1

Introducción

1.1 Motivación

Cada día, la gestión y participación en eventos deportivos presenta desafíos debido a la fragmentación existente en las plataformas y canales de difusión, lo cual dificulta la promoción efectiva de eventos y la creación de una comunidad deportiva cohesionada. Para abordar estas dificultades, se presenta ALLSPORT, una plataforma que busca simplificar y centralizar la organización de eventos deportivos, promoviendo una comunidad inclusiva y accesible para personas de todas las edades, géneros y niveles de habilidad. Este proyecto tiene como objetivo principal fomentar la participación en el deporte, mejorar la eficiencia en la gestión de eventos y fortalecer los lazos comunitarios, contribuyendo así a un estilo de vida activo y saludable.

1.2 Objetivos

El objetivo principal de ALLSPORT es desarrollar una plataforma unificada que centralice la gestión, promoción y participación de eventos, permitiendo a las empresas darse a conocer en un espacio común.
La plataforma se enfoca en crear una aplicación web que integre todas las funciones necesarias para una administración eficiente de eventos, facilitando a las empresas la promoción de sus actividades.
Asimismo, se busca fomentar una comunidad deportiva inclusiva, proporcionando un espacio donde todo tipo de amante del deporte puedan conectarse, compartir experiencias y participar activamente en eventos. Además, se implementará un sistema de calendarios interactivos que garantice una experiencia de usuario intuitiva y accesible, promoviendo la interacción y el compromiso con los eventos deportivos.

1.3 Metodología y tecnología

A lo largo del desarrollo del proyecto, se han utilizado una amplia variedad de tecnologías comprendiendo cada una de ellas desde la base. Este enfoque ha permitido desarrollar una solución completa y adquirir el conocimiento de cada herramienta y su integración con el flujo de trabajo.
Se ha usado **EJS** para generar vistas dinámicas en el lado del servidor y permite trabajar con **HTML**. El uso de **CSS** y **Bootstrap** se orientó a la creación de una interfaz de usuario activa, asegurando que el producto final sea fácil y accesible.
Node.js fue seleccionado como la plataforma de desarrollo por su capacidad para ejecutar **JavaScript** en el servidor, lo que permitió unificar el lenguaje tanto en el frontend como en el backend, facilitando la gestión de rutas, controladores y la integración con la base de datos. Para ello, **Express** sirvió como el framework que simplifica la creación de aplicaciones web y Apis, proporcionando una estructura clara y eficiente para el desarrollo backend.
En cuanto a la interacción con la base de datos, se optó por **Sequelize**, un ORM que no solo facilita la gestión de la base de datos **MySQL**, sino que también permitió escribir código más limpio y mantener una separación clara entre la lógica de negocio y la base de datos. Se usó **XAMPP** para configurar el entorno de servidor y base de datos, lo que dio un control total sobre los servicios de backend. Para la gestión de archivos multimedia, se utilizó **Multer**, que permite un manejo eficiente y seguro de la carga de archivos, una necesidad crítica para el proyecto. Además, **JQuery** y **Ajax** fueron integrados para mejorar la interactividad del frontend, permitiendo una experiencia de usuario más dinámica y fluida.
Dotenv fue clave para manejar las configuraciones sensibles del proyecto, como las credenciales de la base de datos, asegurando que esta información se mantuviera segura y separada del código fuente. La gestión del control de versiones a través de **GitHub** fue esencial para mantener un historial detallado de cambios y colaborar eficazmente. **Visual Studio Code** fue el editor de elección, proporcionando un entorno de desarrollo ágil y eficiente.
La elección de la metodología **Scrum** se basó en su capacidad para adaptarse a los cambios y mejorar continuamente, características esenciales dado que el desarrollo de software suele implicar ajustes frecuentes. Esta metodología permitió organizar el trabajo en sprints cortos, evaluar el progreso regularmente y realizar ajustes rápidos en función de los desafíos y las oportunidades que surgían durante el desarrollo.

1.4 Estructura de la memoria

La memoria ha sido desarrollada para guiar al lector a lo largo de las diferentes etapas del desarrollo del proyecto, asegurando una comprensión clara y detallada. A continuación, se desglosan cada una de ellas.

Primer capítulo: Introducción: en esta primera parte, se presentan los aspectos clave del proyecto, empezando por la motivación que impulsó su desarrollo. Aquí se explica por qué se eligió este tema y cuáles son los objetivos que se pretenden alcanzar. Además, se describe la metodología empleada para llevar a cabo el trabajo, junto con las tecnologías utilizadas. Por último, se ofrece una visión general de cómo está organizada la memoria, de modo que el lector sepa qué esperar en cada capítulo.

Segundo capitulo: Análisis del sistema: este capítulo se dedica a analizar el sistema que se quiere desarrollar. Empieza con un estudio de la competencia, identificando qué soluciones similares existen y en qué se diferencian del proyecto propuesto. Luego, se elabora un catálogo de requisitos, donde se recogen las funcionalidades que el sistema debe tener, tanto a nivel técnico como de usuario. También se incluyen casos de uso que muestran cómo los usuarios interactuarán con el sistema en situaciones concretas. El objetivo aquí es entender bien qué se necesita y cómo se va a cubrir esas necesidades.

Tercer capítulo: Diseño: se describe cómo se va a estructurar y cómo va a funcionar el sistema. Se incluyen diagramas de secuencia que ilustran el flujo de interacciones entre los diferentes componentes. También se detalla el modelo de almacenamiento, explicando cómo se van a organizar los datos dentro del sistema. A esto se suma el modelo físico, que describe la infraestructura técnica necesaria para que todo funcione. Además, se presenta el diseño de la interfaz de usuario, mostrando cómo será la interacción entre el sistema y las personas que lo utilicen. Todo esto sirve como base para la siguiente fase, que es la implementación.

Cuarto capítulo: Implementación: este capítulo se centra en la implementación del sistema. Aquí se explica cómo se ha construido el proyecto, detallando la arquitectura del software y las bibliotecas que se han utilizado. También se describe cómo está organizada la estructura del proyecto, incluyendo la división entre frontend y backend, y cómo se conectan estos componentes con la base de datos. Se discuten las medidas de seguridad que se han implementado para proteger el sistema. Además, se incluyen ejemplos prácticos que ilustran algunos de los retos que surgieron durante la implementación y cómo se resolvieron.

Quinto capítulo: Conclusiones: se analizan los resultados obtenidos y se evalúa si se han cumplido los objetivos planteados al principio. Además, se ofrece una reflexión personal sobre lo aprendido y cómo ha sido la experiencia de trabajar en el proyecto. También se discuten posibles mejoras y nuevas funcionalidades que podrían añadirse en el futuro para seguir desarrollando la plataforma.

Bibliografía: contendrá todas las fuentes consultadas durante el desarrollo de la aplicación, proporcionando una referencia completa de los recursos.

Primer apéndice: Manual de instalación: se incluye un manual que explica paso a paso cómo instalar y desplegar el sistema en un entorno local. Se detallan los requisitos previos, la configuración necesaria y todos los pasos a seguir para poner en marcha el proyecto.

Segundo apéndice: Manual de usuario: este segundo apéndice contiene un manual de usuario dirigido a las personas que vayan a utilizar el sistema. Se explica cómo usar cada una de las funcionalidades disponibles, con ejemplos prácticos que ayudan a entender mejor cómo interactuar con la plataforma.

2

Análisis del sistema

2.1 Análisis de la competencia y diferencias

A lo largo del desarrollo del proyecto se han evaluado aplicaciones existentes para la gestión de eventos deportivos. Analizándolas se han podido identificar qué aspectos se pueden integrar y mejorar dentro la aplicación.

2.1.1 EventBrite

Es una de las plataformas más reconocidas y utilizadas a nivel mundial para la gestión de eventos, permitiendo a los usuarios crear, promover y administrar una amplia variedad de eventos, desde conferencias y conciertos hasta talleres y seminarios. Su popularidad se debe, en gran parte, a su interfaz intuitiva y a su capacidad para manejar eventos de diferentes tamaños , lo que la convierte en una opción versátil para organizadores en diversas industrias.

Sin embargo, a pesar de su gran reputación y su éxito en la gestión de eventos en general, EventBrite no está optimizada específicamente para eventos deportivos, lo que puede limitar su eficacia cuando se trata de este nicho en particular. Los eventos deportivos suelen requerir características y funcionalidades especializadas, como la gestión de equipos, la programación de torneos o la venta de entradas basada en asientos específicos en estadios.

A diferencia de EventBrite, el proyecto desarrollado se ha diseñado exclusivamente para cubrir las necesidades de los eventos deportivos. Esto significa que ofrece herramientas y funcionalidades específicas que mejoran significativamente la experiencia del usuario tanto para los organizadores como para los asistentes a los eventos. La plataforma del proyecto se centra en optimizar la gestión de eventos deportivos, proporcionando soluciones adaptadas a las particularidades de este tipo de eventos, desde la promoción y venta de entradas hasta

la organización y seguimiento de actividades deportivas. Esto no solo facilita la gestión, sino que también permite una experiencia de usuario más personalizada y eficiente, posicionándose como una solución más adecuada para el sector.

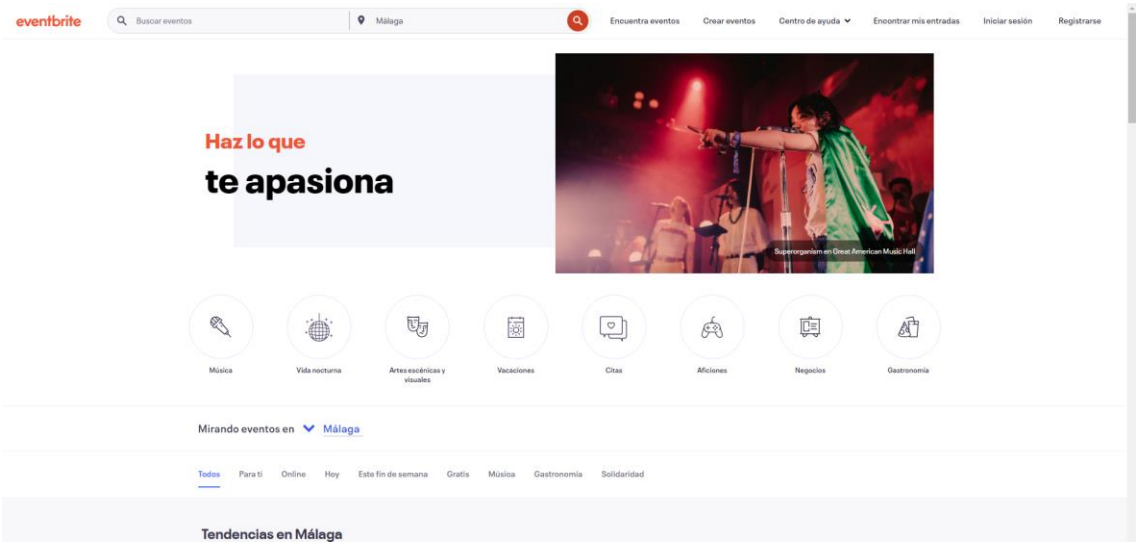

Figura 1 EventBrite página principal, Fuente: https://www.eventbrite.es/l/gestion-eventos-online/

2.1.2 Meetup

En la Figura 2 se observa como Meetup es una plataforma intuitiva y accesible que se destaca por facilitar la creación y el fomento de comunidades en torno a intereses comunes. Su enfoque principal es ayudar a las personas a organizar y unirse a encuentros y actividades sociales, desde grupos de lectura hasta clases de yoga, permitiendo a los usuarios conectarse con otros que comparten sus pasiones y hobbies.
Sin embargo, a pesar de su eficacia en la creación de comunidades, Meetup no está orientada específicamente hacia la gestión de eventos deportivos. La plataforma carece de un sistema dedicado a la gestión y venta de entradas, lo que puede ser una limitación significativa cuando se trata de organizar eventos deportivos que requieren una logística más compleja, como la administración de asistentes, la venta de entradas para diferentes categorías o la integración de herramientas específicas para deportes.
El proyecto desarrollado se centra en abordar esta brecha, proporcionando un sistema robusto y especializado que va más allá de lo que ofrece Meetup. La plataforma no solo facilita la creación y promoción de eventos deportivos, sino que también incluye un sistema completo de gestión de entradas, permitiendo a los organizadores vender entradas de manera eficiente y segura.

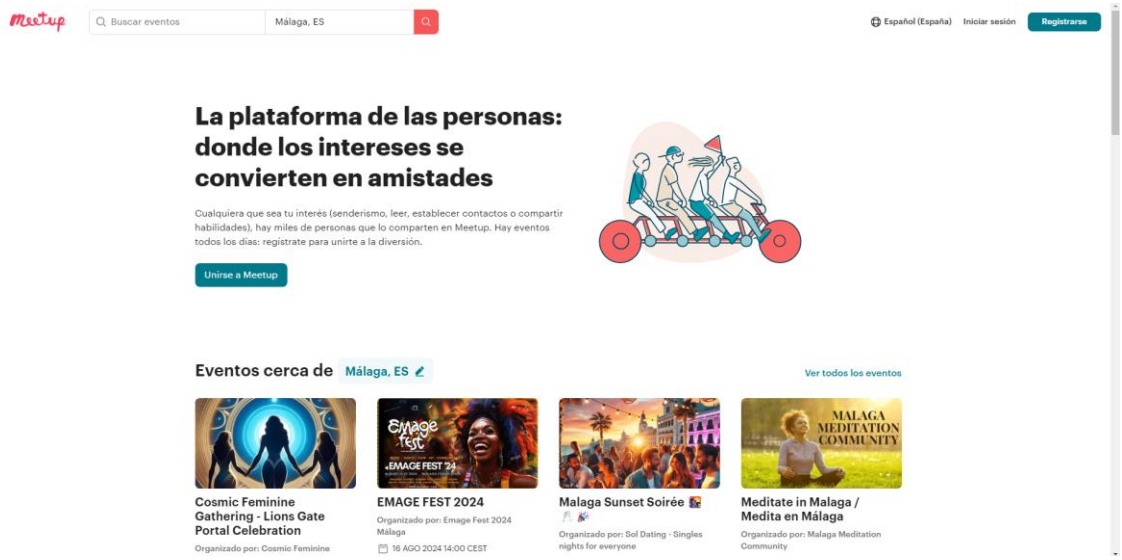

Figura 2 Meetup página principal, Fuente: https://www.meetup.com/es-ES/

2.1.3 Active Network

En la Figura 3 se observa como Active Network es una herramienta enfocada en la gestión de inscripciones y el seguimiento de participantes en eventos deportivos. Ofrece funcionalidades avanzadas como la administración detallada de registros, la capacidad de gestionar grandes volúmenes de datos, y opciones para el análisis de rendimiento de los participantes. Sin embargo, su nivel de complejidad puede ser una barrera para los organizadores de eventos de menor escala o aquellos que buscan una solución más sencilla e intuitiva. La curva de aprendizaje para utilizar todas sus funciones puede resultar desafiante, especialmente para usuarios menos experimentados.
A diferencia de la aplicación desarrollad que, si busca encontrar un equilibrio entre la funcionalidad avanzada y la facilidad de uso, proporcionando una solución que sea lo suficientemente robusta para gestionar eventos deportivos de diferentes tamaños, pero sin la complejidad que podría alejar a los usuarios menos técnicos. La plataforma está diseñada para ser escalable, lo que permite su adaptación tanto a eventos pequeños como a eventos de mayor envergadura, ofreciendo una experiencia de usuario más accesible y directa sin sacrificar las capacidades necesarias para una gestión efectiva.

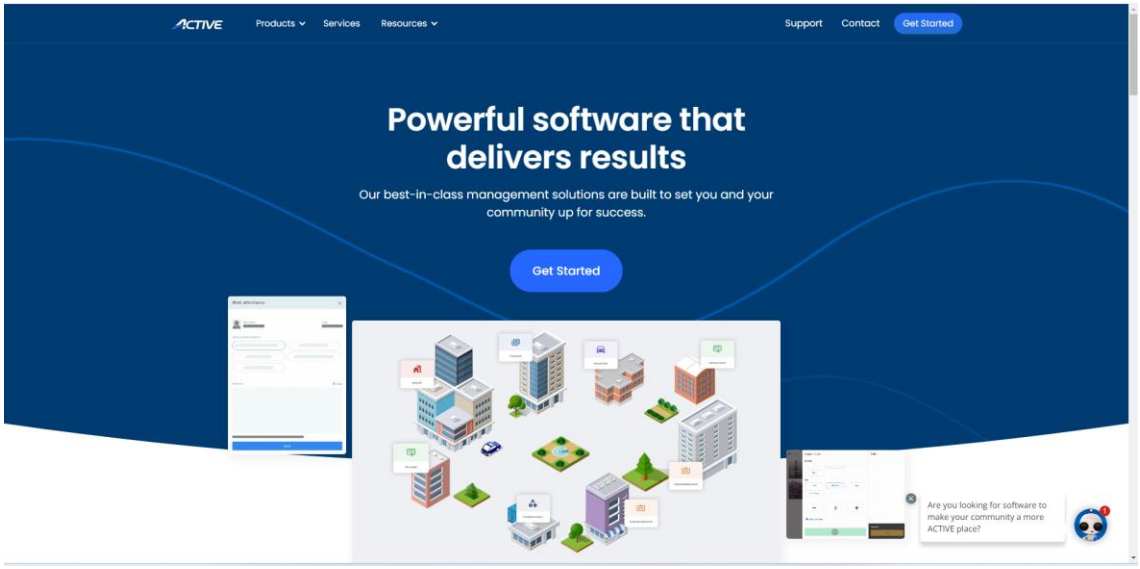

Figura 3 Active Network página principal, Fuente: https://www.activenetwork.com/

2.1.4 Facebook Events

En la Figura 4 se observa como Facebook Events, integrado en la plataforma de Meta, es una herramienta ampliamente utilizada para la creación y gestión de eventos, aprovechando la vasta red social de Facebook para alcanzar a una audiencia potencialmente grande. Sin embargo, aunque tiene un gran alcance, esta herramienta presenta ciertas limitaciones cuando se trata de eventos deportivos. No ofrece filtros específicos para este tipo de eventos, lo que puede dificultar la búsqueda y categorización de eventos deportivos entre otros tipos de eventos más generales. Además, carece de un calendario visual integrado que facilite la organización y visualización de múltiples eventos deportivos a lo largo del tiempo. Otra limitación es que Facebook Events requiere que los usuarios tengan un perfil en la red social para poder acceder y participar en los eventos, lo que puede excluir a quienes no desean tener una cuenta en Facebook.
El proyecto desarrollado supera estas limitaciones al ofrecer un sistema de filtrado especializado que permite a los usuarios buscar y categorizar eventos deportivos de manera mucho más eficiente. Además, incluye un calendario integrado que facilita la visualización y planificación de eventos deportivos, mejorando significativamente la usabilidad y accesibilidad para los usuarios interesados en este tipo de eventos. Al no estar vinculado a una red social en particular, la plataforma permite que cualquier usuario, independientemente de su presencia en redes sociales, pueda acceder y beneficiarse de todas sus funcionalidades.

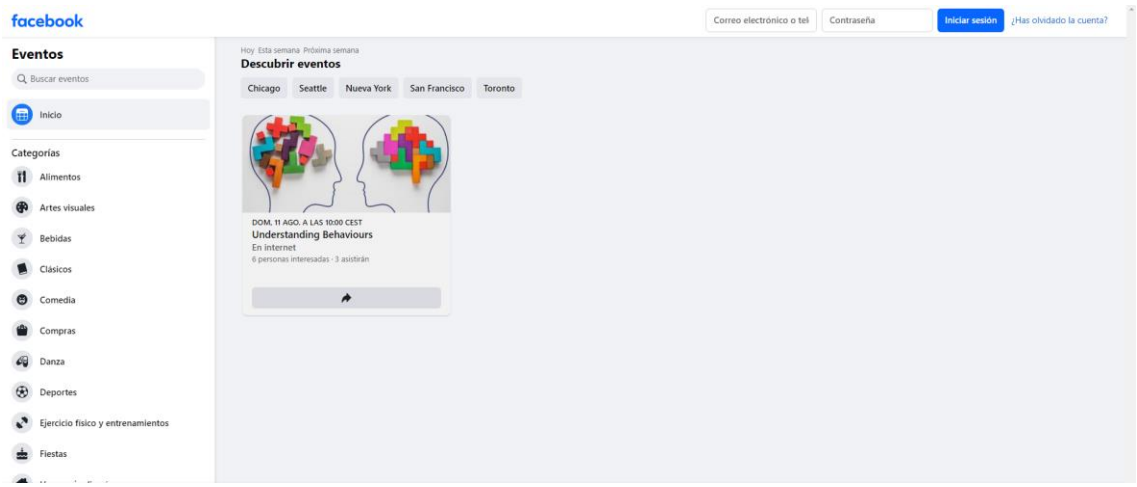

Figura 4 Facebook Events página principal, Fuente: https://www.facebook.com/events/

2.2 Catálogo de requisitos

Se han tomado en cuenta aplicaciones como EventBrite, Meetup, Active Network y Facebook Events a lo largo del desarrollo de la aplicación, teniendo en cuenta sus puntos fuertes y buscando mejorar en áreas clave. La aplicación se especializa en los eventos deportivos, ofreciendo una interfaz accesible, un sistema de venta de entradas, y herramientas específicas para la gestión y promoción de eventos, superando así las limitaciones encontradas en las plataformas de la competencia.

2.2.1 Requisitos funcionales

Los requisitos funcionales declaran como debe comportarse el sistema para satisfacer las necesidades del usuario.

Código	Titulo	Descripción
RF-01	CRUD Usuarios	La aplicación permite crear usuarios con sus datos diferenciando entre persona y empresa. Se deberá enviar un correo electrónico para cualquier cosa relacionada con ellos, tanto la creación, compra de entrada o actualización de datos.
RF-02	Autenticación de usuarios	Los usuarios deben validar su usuario con el correo de confirmación.
RF-03	Registro	Todo el mundo podrá registrarse en la aplicación diferenciando el tipo de usuario que desea ser.

RF-04	Olvidar contraseña	Dentro del inicio de sesión se podrá acceder a un enlace donde el usuario puede restablecer su contraseña.
RF-05	Cerrar Sesión	Cualquier usuario podrá cerrar su sesión y acceder a la página principal desde el menú de navegación.
RF-06	CRUD Eventos	Las empresas podrán crear sus propios eventos y hacer todas las modificaciones posibles sobre ellos en cualquier momento. Los usuarios solo podrán acceder a la información de ellos. Los administradores del sistema podrán hacer lo mismo que las empresas, pero para cualquier evento.
RF-07	Panel de administración	Los administradores podrán hacer cualquier tipo de modificaciones sobre los eventos.
RF-08	Gestión de evento destacado	El administrador podrá seleccionar entre todos los eventos que se encuentren almacenados cuál es el que tiene mayor repercusión.
RF-09	Visión eventos	Cualquier usuario podrá acceder a los eventos y ver la información detallada e imágenes asociadas.
RF-10	Subir archivos	Los administradores y empresas podrán modificar los eventos y añadir las imágenes que consideren necesarias. Las empresas solo podrán hacerlo de los eventos que tengan asociados y los administradores tendrán acceso a todos.
RF-11	Visión usuario	El administrador podrá ver la web desde el punto de vista de un usuario para poder comprobar posibles fallos.
RF-12	Visión empresa	El administrador podrá ver la web desde el punto de vista de una empresa para poder comprobar posibles fallos.
RF-13	Venta de entradas	Los usuarios podrán comprar entradas para los eventos. La aplicación deberá generar el comprobante de compra y enviar la entrada al correo del usuario. Todo esto es configurado con un proxy que simula la pasarela de pago.
RF-14	Generación de entrada	Una vez realizada la compra, se enviará un correo con la entrada para el evento. Deberá ser única y personalizada para cada evento.
RF-15	Gestión de inscripciones	Se deberá controlar la venta de entradas teniendo en cuenta la cantidad que ponga la empresa promotora.
RF-16	Inicio	Todo usuario que entre por primera vez podrá acceder a una página donde se puede hacer el

		registro, iniciar sesión o ver información sobre la empresa.
RF-17	Calendario de eventos	La aplicación contendrá un calendario donde aparecerán los eventos marcados de cada día para que el usuario pueda acceder a la información de ellos.
RF-18	Buscador de eventos	Todo usuario registrado como "persona" podrá filtrar todos los eventos en función de la categoría.
RF-19	Ordenación de eventos	Las vistas de eventos tendrán un filtro para hacer más intuitiva la experiencia del usuario.
RF-20	Política de privacidad	Antes de confirmar el registro se deberá aceptar la política de privacidad, pudiendo acceder a ella en todo momento.
RF-21	Cookies	El usuario al entrar podrá aceptar o denegar las cookies

2.2.2 Requisitos no funcionales

Los requisitos no funcionales son aquellos que definen la calidad y las restricciones bajo las cuales el sistema debe operar.

Código	Titulo	Descripción
RNF-01	Cifrado de contraseña	Las contraseñas de los usuarios registrados deberán almacenarse cifradas en la base de datos
RNF-02	Encriptación	Las contraseñas deberán ser cifradas con bcrypt, usando un hashing unidireccional. No permitiendo que una contraseña una vez cifrada pueda ser desencriptada.
RNF-03	Almacenamiento de datos	La base de datos deberá tener la capacidad de almacenar todos los datos.
RNF-04	Inicio de sesión	Solo podrán realizar compras, creación de eventos o modificaciones los usuarios que hayan validado su correo electrónico.
RNF-05	Legalidad	Se deberá cumplir en todo momento con la Ley Orgánica de Protección de Datos y la garantía de derechos digitales.
RNF-06	Software	Aplicación diseñada en un principio para web.
RNF-07	Base de datos	La base de datos será relacional (MySQL)
RNF-08	Navegadores	La aplicación será accesible desde todos los navegadores.

RNF-09	Reactividad	La aplicación web se recargará de manera automática ante cualquier tipo de modificación.
RNF-10	Lenguajes	El desarrollo será principalmente con JavaScript que permite el enlace entre el frontend y el backend que es en Nodejs con el servidor en ExpressJS.
RNF-11	Usabilidad	La interfaz debe ser intuitiva y fácil de usar.
RNF-12	Disponibilidad	La aplicación deberá estar disponible en todo momento.
RNF-13	Retroalimentación	Los usuarios recibirán validaciones inmediatas en la web al interactuar con la aplicación, mientras que en el backend se generarán respuestas JSON para asegurar una comunicación eficiente y coherente.
RFN-14	Seguridad	La seguridad de la sesión del usuario se gestionará mediante el módulo JWT de Node.js, que almacena la sesión en un token, garantizando un acceso seguro y autenticado a la aplicación.
RFN-15	Actualizaciones en la base de datos	Los cambios realizados tanto en los eventos como en los usuarios deberán verse reflejados de manera automática en las tablas correspondientes.

2.2.3 Requisitos de información

Los requisitos de información se refieren a los datos que el sistema necesita manejar para funcionar correctamente.

Código	Titulo	Descripción
RI-01	Usuario	ID: Integer, Primary Key, auto incremental user: String pass: String email: String verified: Boolean, Valor por defecto: false resetToken: String role: Enum ('user', 'company', 'admin') token: String
RI-02	Persona	ID: Integer, Primary Key, auto incremental usuario_id: Integer, Foreign Key hacia Usuario (ID) nombre: String apellido: String email: String

RI-03	Admin	ID: Integer, Primary Key, auto incremental usuario_id: Integer, Foreign Key hacia Usuario (ID)
RI-04	Compañía	ID: Integer, Primary Key, auto incremental usuario_id: Integer, Foreign Key hacia Usuario (ID) email: String nif: String contacto: String
RI-05	Evento	ID: Integer, Primary Key, auto incremental título: String (255) descripción: Text numero_entradas: Integer localización: String (255) precio: Decimal (10, 2) fecha_inicio: Date fecha_fin: Date deporte: Text id_compania: Integer, Foreign Key hacia Compania (ID)
RI-06	Evento Clase	evento id: Integer, Primary Key, Foreign Key hacia Evento (ID) instructor: String duración: String nivel: String
RI-07	Evento Partido	evento_id: Integer, Primary Key, Foreign Key hacia Evento (ID) equipo_local: String equipo_visitante: String liga: String
RI-08	Evento Campus	evento_id: Integer, Primary Key, Foreign Key hacia Evento (ID) programa: String
RI-09	Evento Ocasión	evento_id: Integer, Primary Key, Foreign Key hacia Evento (ID) tipo_ocasion: String

2.3 Casos de uso

Los casos de uso son descripciones de las interacciones entre un usuario y un sistema para lograr un objetivo específico. Detallan los pasos que un usuario sigue para realizar una tarea en el sistema, ayudando a identificar las funcionalidades que debe tener el software y cómo deben responder a las acciones del usuario. A continuación, se muestran los casos de uso para cada perfil de usuario.

2.3.1 Administrador

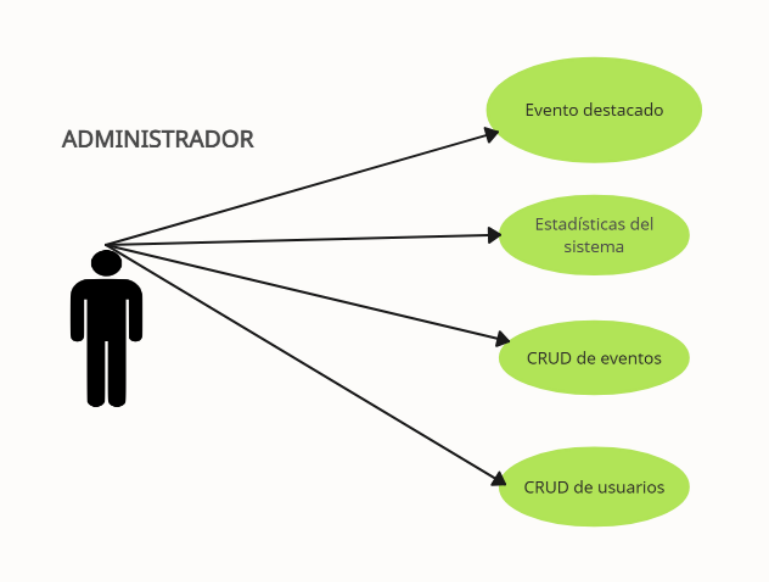

Figura 5. Caso de uso para el administrador

El administrador (Figura 5) tiene acceso a todas las funcionalidades de la web, incluyendo la capacidad de marcar eventos como destacados para obtener mayor visibilidad dentro de la plataforma. También puede acceder a estadísticas del sistema, permitiéndole visualizar la cantidad de entradas restantes para cada evento o consultar el calendario de eventos. Además, el administrador puede crear, editar, eliminar y gestionar la información de todos los eventos en la plataforma, asegurando un control completo sobre la gestión y promoción de los eventos deportivos. Por último, podrá tener una lista con todos los usuarios del sistema para darles de baja y poder marcar los eventos como destacados.

2.3.2 Empresa

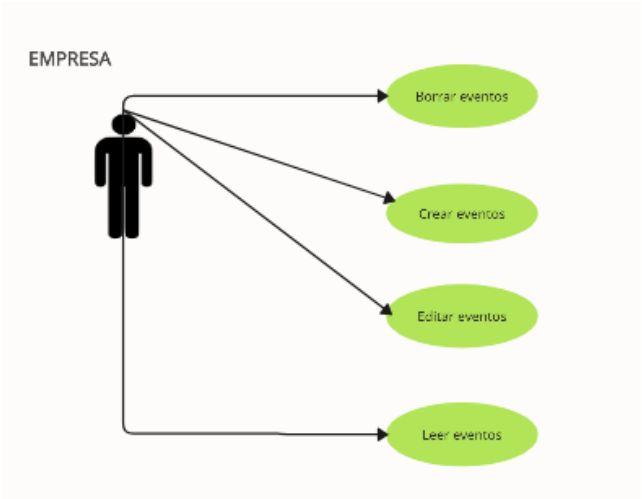

Figura 6. Caso de uso para las empresas

La empresa (Figura 6) puede interactuar con la plataforma a través de varias funcionalidades clave. Puede crear eventos, agregando nuevos eventos a la plataforma con sus respectivas imágenes. También puede editarlos, permitiéndole modificar detalles de eventos existentes. La empresa tiene la capacidad de borrar sus eventos en caso de que lo consideren necesario.

2.3.3 Usuario

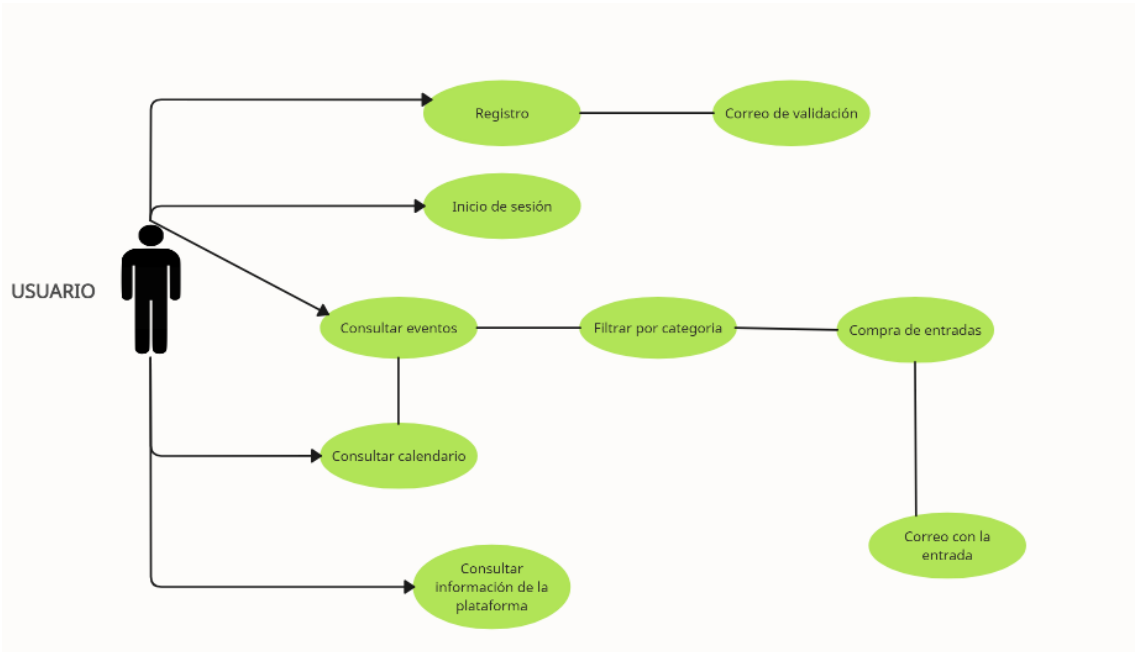

Figura 7. Caso de uso para los usuarios

En la Figura 7 se observa como el usuario puede interactuar con la plataforma a través de varias funcionalidades clave. Primero, puede registrarse y recibir un correo de validación. Una vez registrado, puede iniciar sesión para acceder a todas las funcionalidades.
El usuario puede consultar y filtrar eventos por categoría para encontrar aquel que desee. También puede comprar entradas y recibirá un correo con los detalles de estas.
Además, puede consultar el calendario de eventos y acceder a la información de la plataforma para obtener más detalles sobre su funcionamiento y características.

3

Diseño

3.1 Diagramas de secuencia

Los diagramas de secuencia son una herramienta visual utilizada en el desarrollo de software para representar cómo interactúan los diferentes componentes de un sistema a lo largo del tiempo.
En estos diagramas, los actores y objetos se ubican en la parte superior, con líneas de vida que descienden desde ellos, representando el tiempo. Las interacciones se muestran mediante flechas que conectan las líneas de vida, indicando el envío de mensajes o la invocación de métodos.

3.1.1 Inicio de sesión

El diagrama de secuencia presentado describe el proceso de inicio de sesión en la aplicación. Comienza con el usuario introduciendo sus credenciales (usuario y contraseña) en la interfaz de inicio de sesión. Una vez que los datos son ingresados, el sistema realiza una validación inicial para verificar si los campos requeridos han sido completados. Si faltan datos, se muestra un mensaje de error al usuario.
Si los datos son correctos, se envían al controlador de usuario (UserController), que se encarga de gestionar la lógica del inicio de sesión. El controlador solicita al modelo de usuarios (Usuarios) que verifique la existencia de las credenciales en la base de datos. El modelo consulta la base de datos y devuelve los resultados al controlador. Finalmente, según los resultados de la búsqueda, el sistema permite el acceso al usuario o muestra un mensaje de error si las credenciales no son válidas.

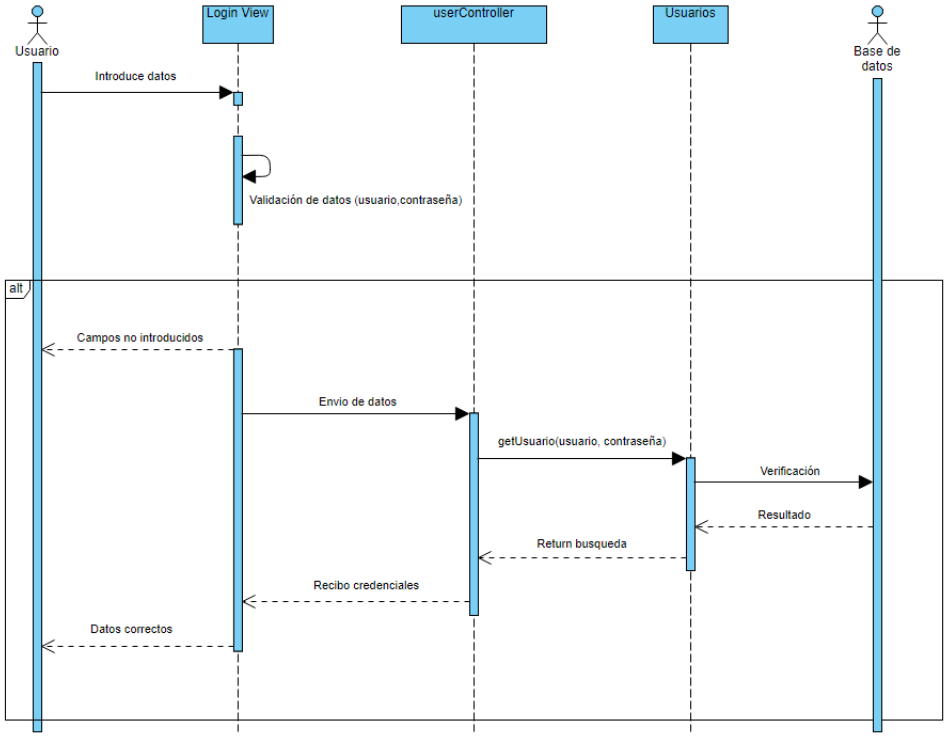

Figura 8. Diagrama de secuencia de inicio de sesión

3.1.2 Búsqueda en la tabla por filtro

El diagrama de secuencia (Figura 9) describe el proceso mediante el cual un usuario accede a la vista de eventos de la aplicación y utiliza filtros para ordenar la lista de eventos disponibles. El flujo comienza cuando el usuario accede a la vista de eventos, lo que desencadena la recuperación de la lista de eventos desde la base de datos a través de una serie de interacciones entre la vista, el controlador y el modelo. Una vez que los eventos se muestran, el usuario puede aplicar filtros de ordenamiento, lo que inicia un nuevo ciclo de consultas a la base de datos para obtener y mostrar la lista de eventos ordenada según los criterios seleccionados. Este proceso asegura que el usuario tenga acceso a una lista de eventos organizada de manera que satisfaga sus preferencias.

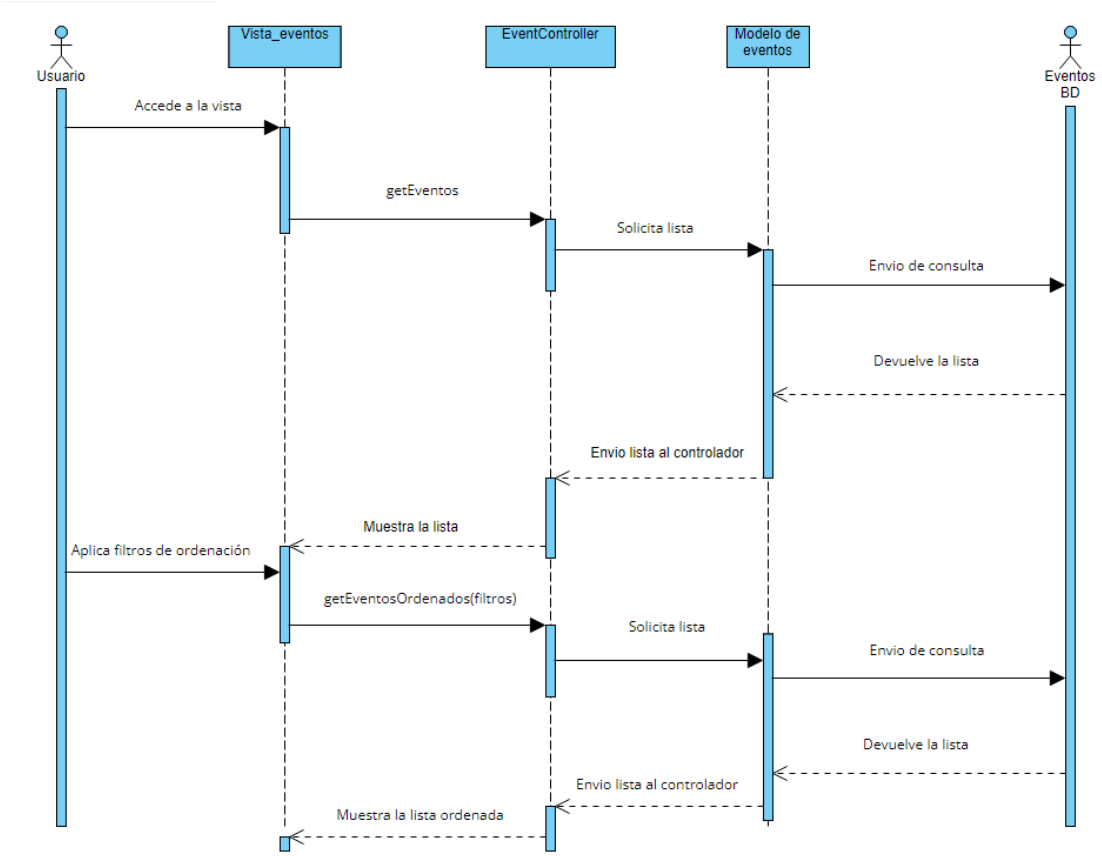

Figura 9. Diagrama de secuencia de ordenar los eventos

3.2 Almacenamiento

Uno de los principales objetivos dentro del desarrollo de la aplicación es poder manejar la subida de imágenes de los eventos de manera eficiente.
La primera idea que surgió fue **almacenar las imágenes en la base de datos**.
Esta idea no es recomendable por los siguientes motivos:

- **Rendimiento:** las bases de datos no están optimizadas para manejar grandes volúmenes de datos.
- **Escalabilidad:** en caso de necesitar operaciones respaldo o recuperación al incrementar el tamaño de la base de datos dificulta el acceso.
- **Costos:** tratándolo como un sistema real, el almacenamiento y mantenimiento se agranda.

Otra de las ideas fue **almacenar las imágenes en la nube**, esta idea también fue descartada debido:

- **Implementación:** las imágenes se almacenan en servicios de almacenamiento en la nube y por tanto se acceden a ellas mediante Url.

- **Complejidad:** requiere configuraciones de SDK y gestión de credenciales de acceso.
- **Dependencia con el proveedor:** depende de la disponibilidad y confiabilidad del proveedor de servicios.

Tras la evaluación de estas opciones se decidió por el uso de **Multer** como almacenamiento en el disco local ya que proporciona una solución rápida y efectiva para el desarrollo del proyecto. Los puntos fuertes son:

- **Simplicidad y rapidez:** permite una configuración rápida durante la fase de desarrollo y resulta más sencillo almacenar los archivos en el servidor local.
- **Control directo:** hay un acceso inmediato y sin latencias de red al estar en el servidor local, también proporciona un control total sobre los archivos.
- **Costos:** esta idea no tiene costes para el desarrollo.

3.3 Modelo físico

El modelo físico es una representación precisa de cómo se organizan y almacenan los datos dentro del sistema de la base de datos. Como se puede apreciar en la Figura 10, en este modelo, se definen las tablas junto con sus nombres y estructuras, las columnas y sus tipos de datos, así como las claves primarias y foráneas que establecen las relaciones entre las tablas. Además, se incluyen los índices y las restricciones, como 'not null' o 'unique' . El objetivo principal del modelo físico es optimizar el rendimiento, mejorando la eficiencia del almacenamiento, la seguridad y la consistencia de los datos .

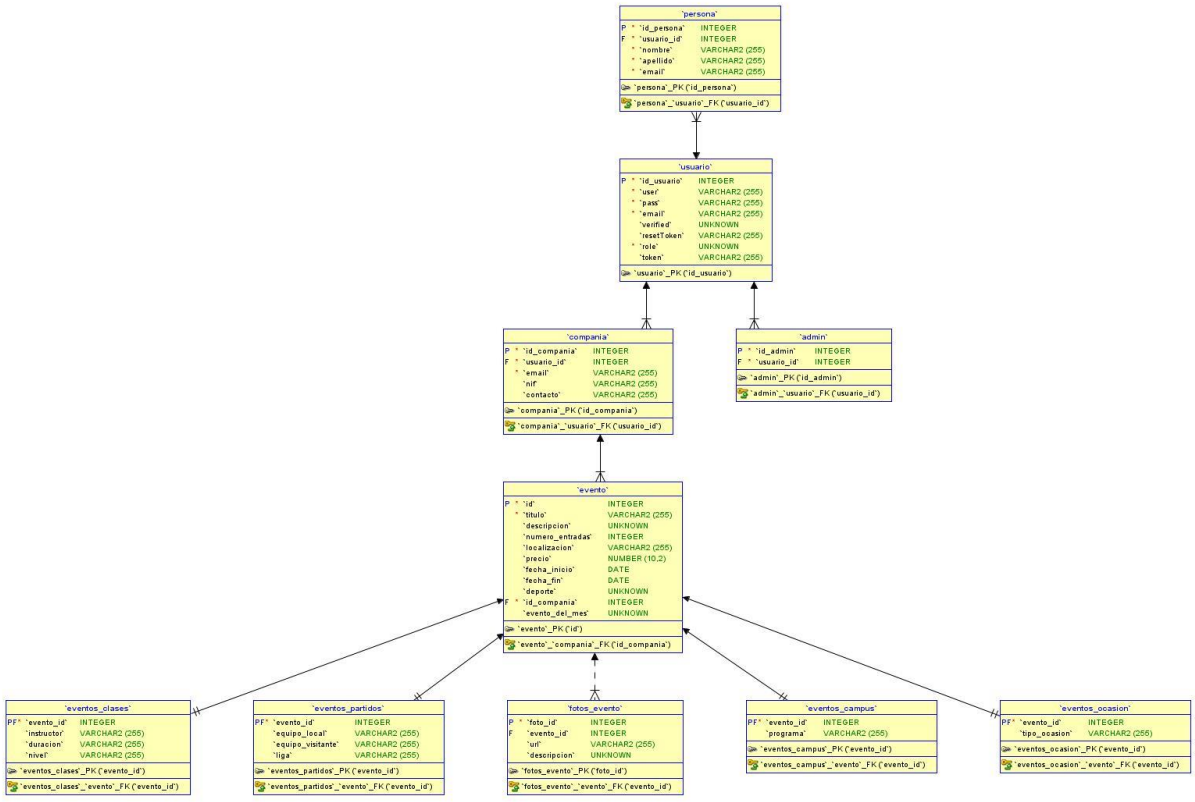

Figura 10. Modelo físico

3.4 Interfaz de usuario

Para el desarrollo de la interfaz se optó por desarrollar la interfaz de usuario directamente sin pasar por un maquetado previo, las ventajas que se tuvieron en cuenta para la toma de la decisión fueron:

- **Iteración rápida:** al desarrollar directamente se obtiene un ciclo de retroalimentación más rápido, permitiendo hacer los cambios relacionados con las funcionalidades más rápido que trabajando primero sobre la maqueta.
- **Integración más temprana de la funcionalidad:** en un maquetado, es difícil visualizar cómo interactuarán los usuarios con los elementos dinámicos o interactivos, mientras que en una interfaz funcional estos aspectos se pueden probar y ajustar continuamente.
- **Mejora continua:** facilita cambios rápidos en el diseño sin la necesidad de rehacer maquetados.
- **Alineación con el usuario:** permite centrarse desde el principio en las necesidades del usuario final, mejorando la usabilidad del producto.

A continuación, se muestran algunas de las principales vistas de la aplicación.

3.4.1 Inicio

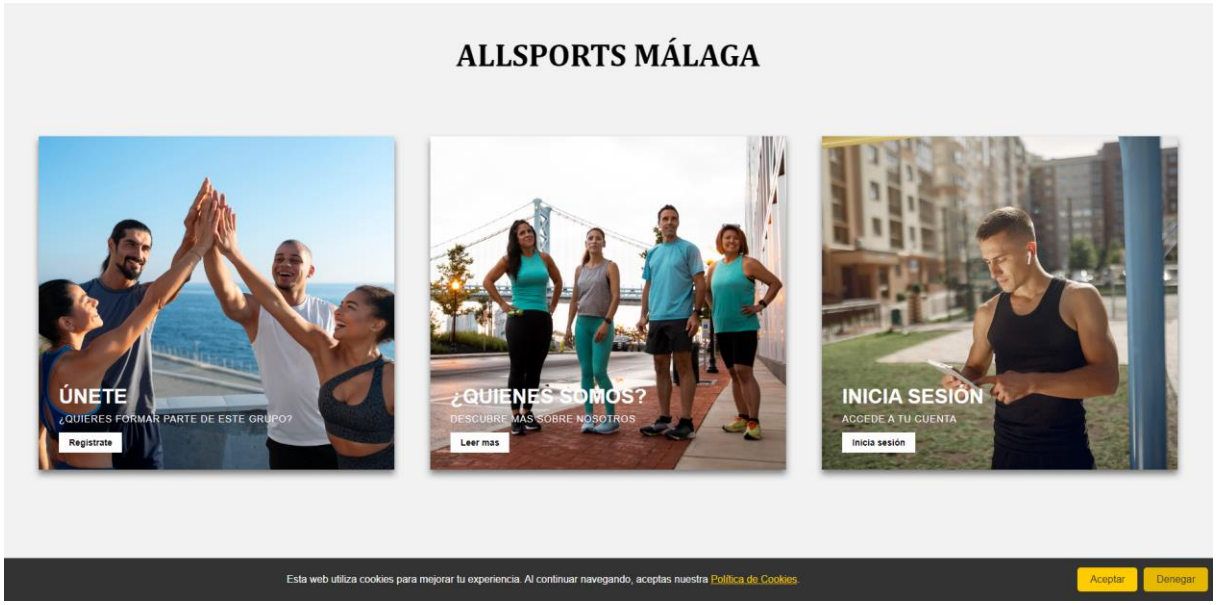

Figura 11. Página inicial

3.4.2 Registro

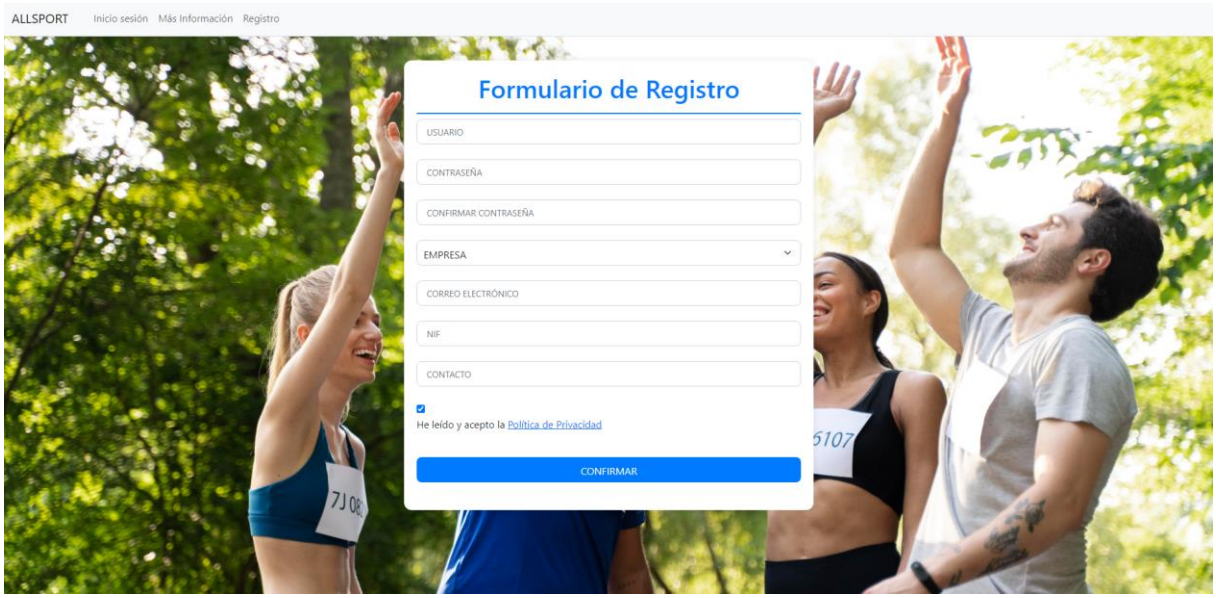

Figura 12. Página de registro

3.4.3 Inicio de sesión

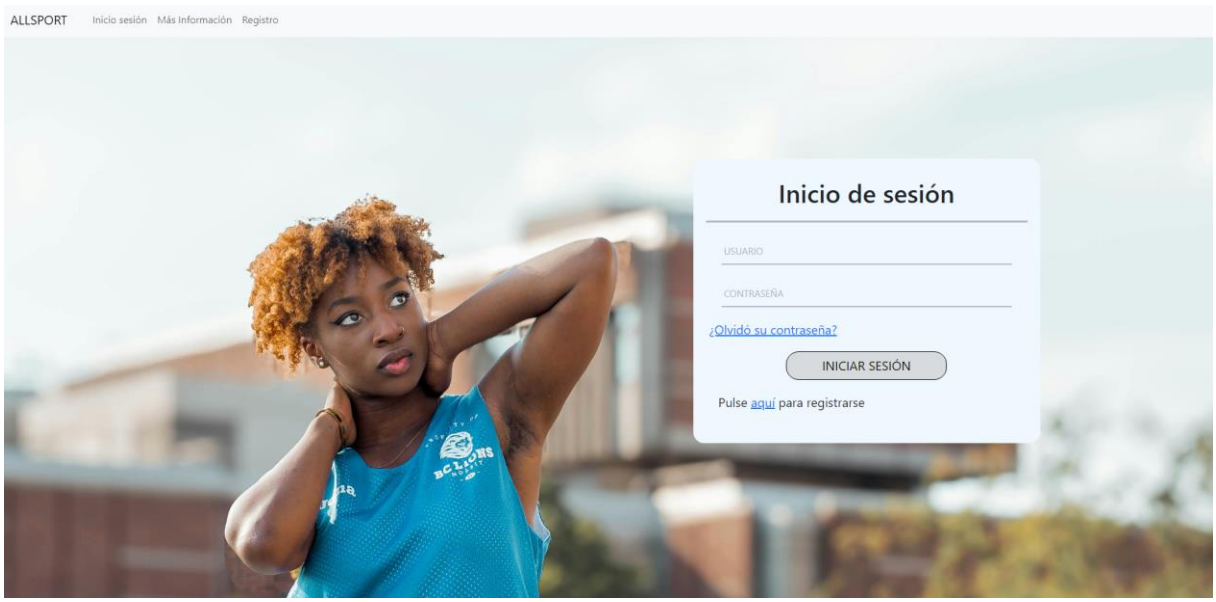

Figura 13. Página de inicio de sesión

3.4.4 Espacio usuario inicial

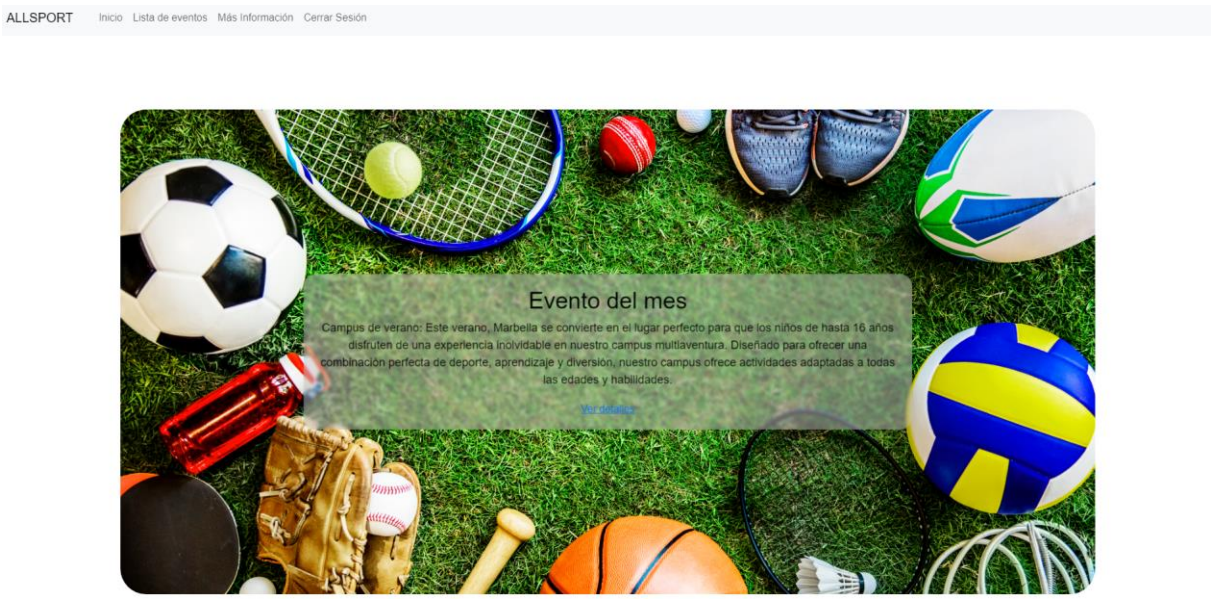

Figura 14. Página de usuario, vista de evento destacado

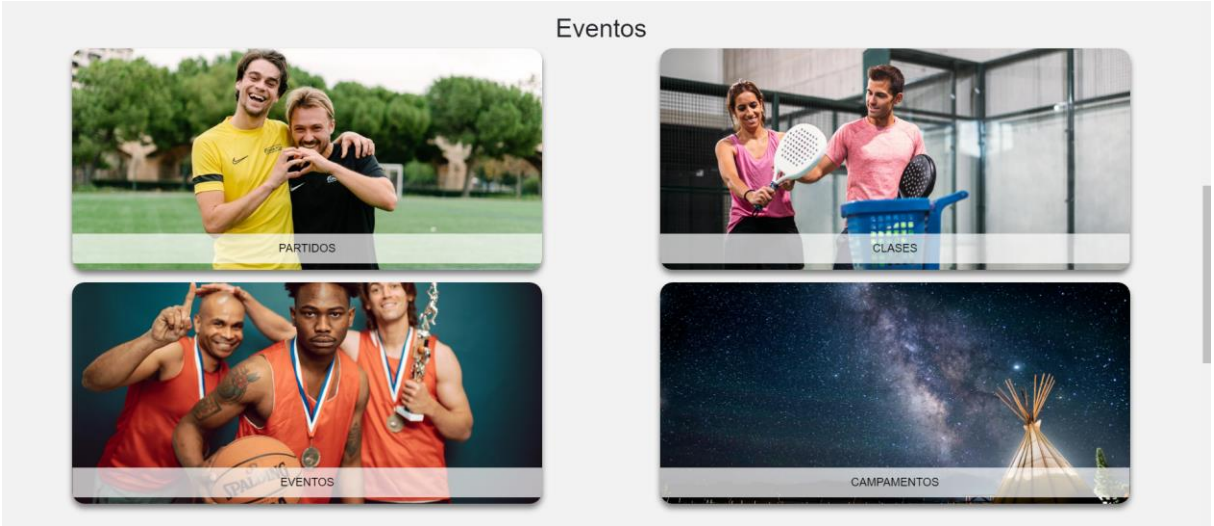

Figura 15. Página de usuario, vista de eventos por categoría

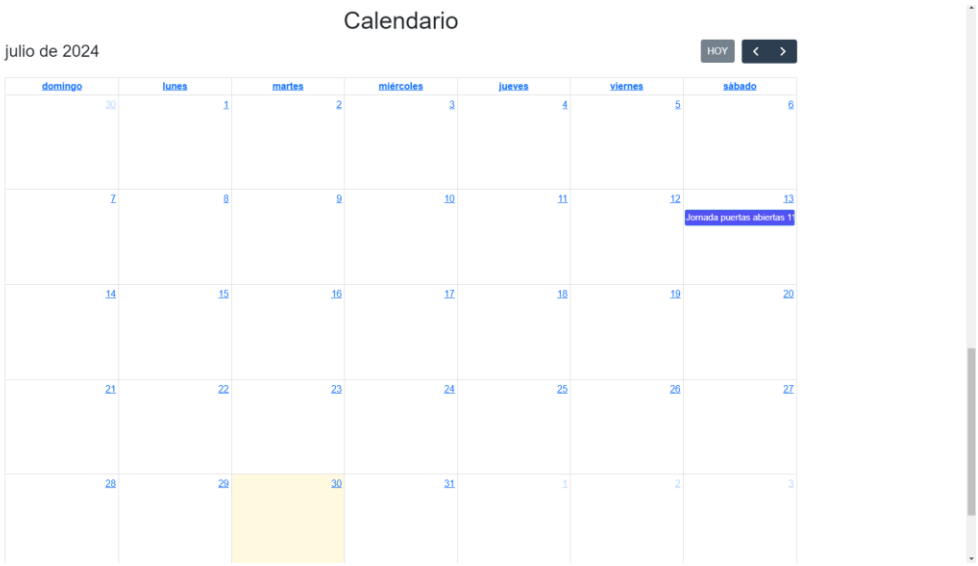

Figura 16. Página de usuario, vista de calendario

3.4.5 Eventos

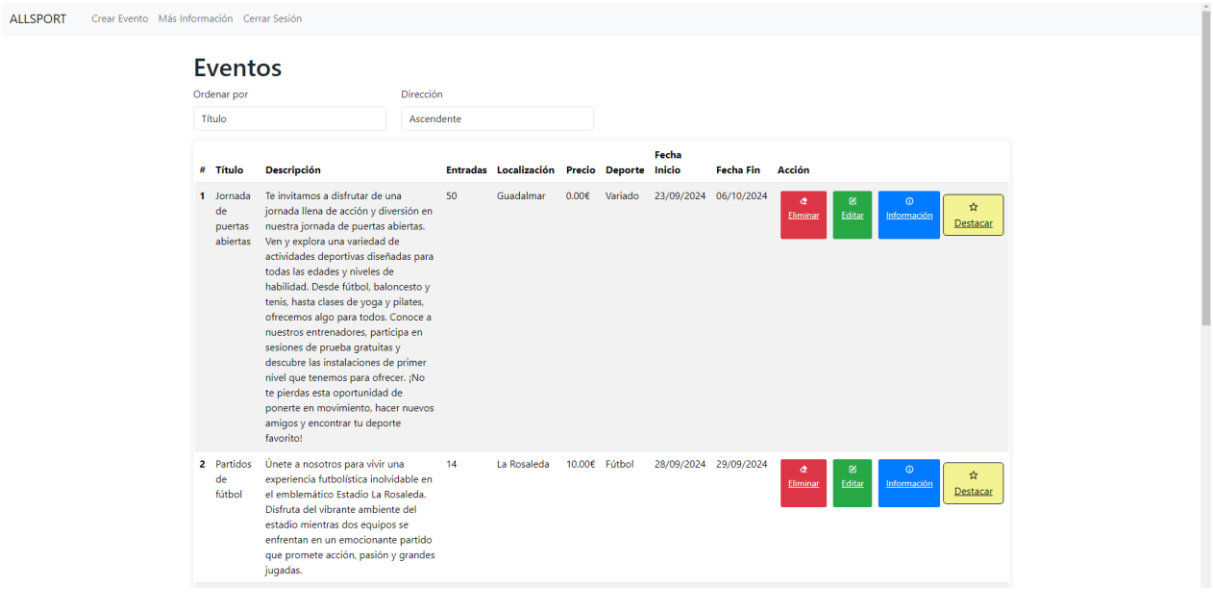

Figura 17. Página de eventos

3.4.6 Información de evento

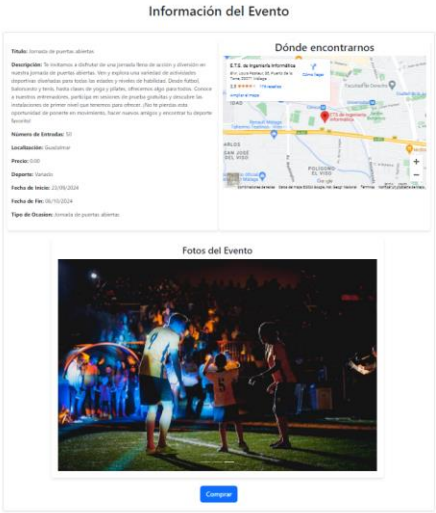

Figura 18. Página de información de eventos

3.4.7 Compra entrada

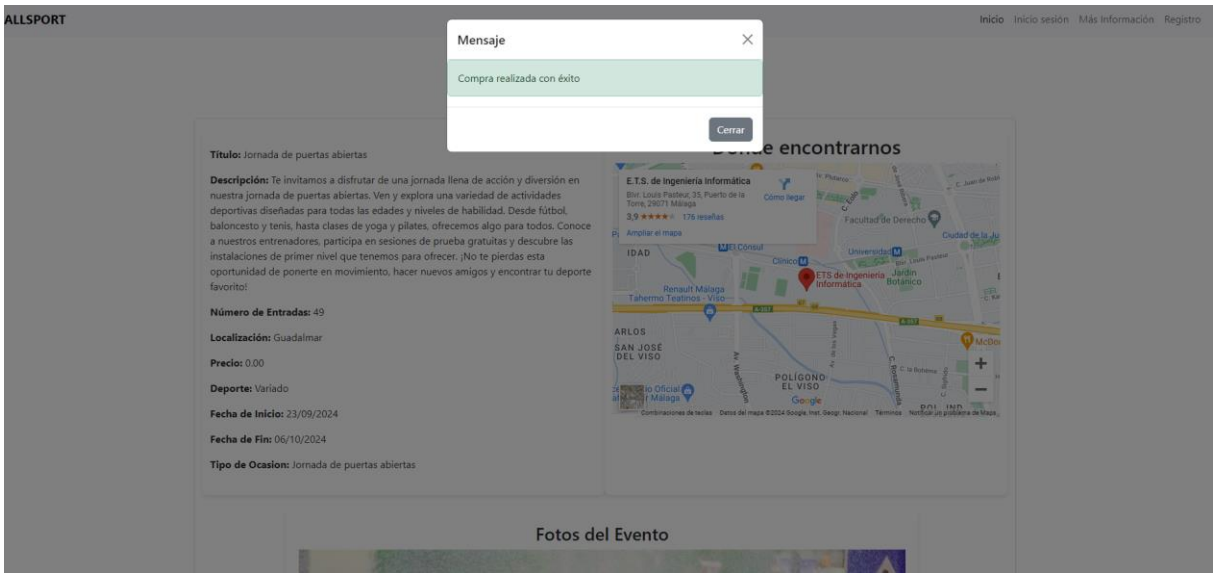

Figura 19. Ventana emergente de compra de entradas

3.4.8 Crear eventos

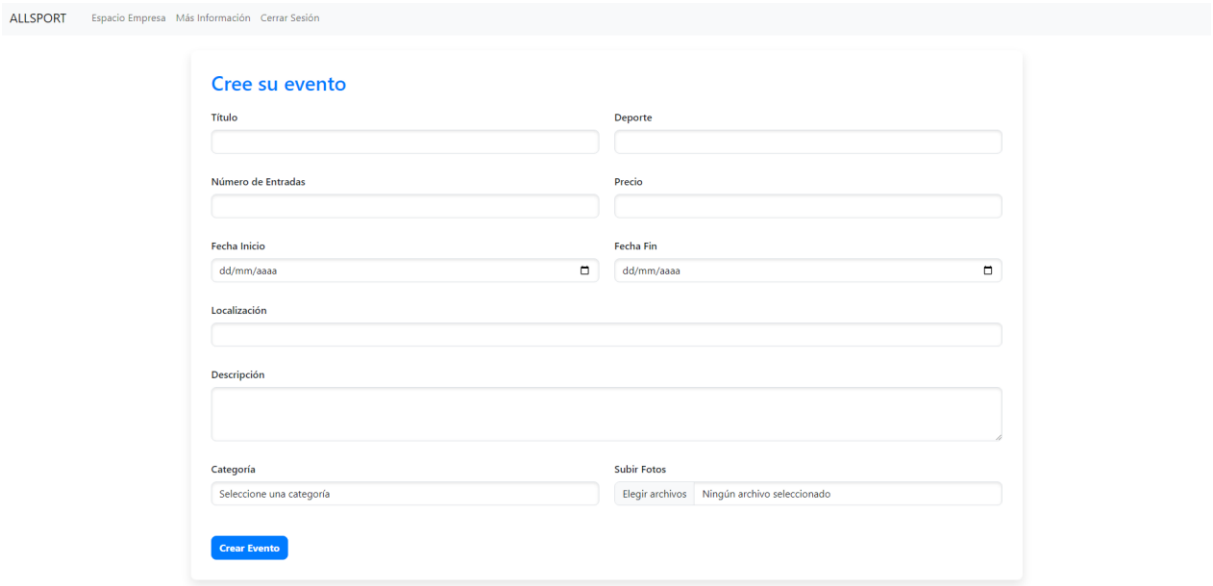

Figura 20. Página de crear eventos

3.4.9 Ver eventos

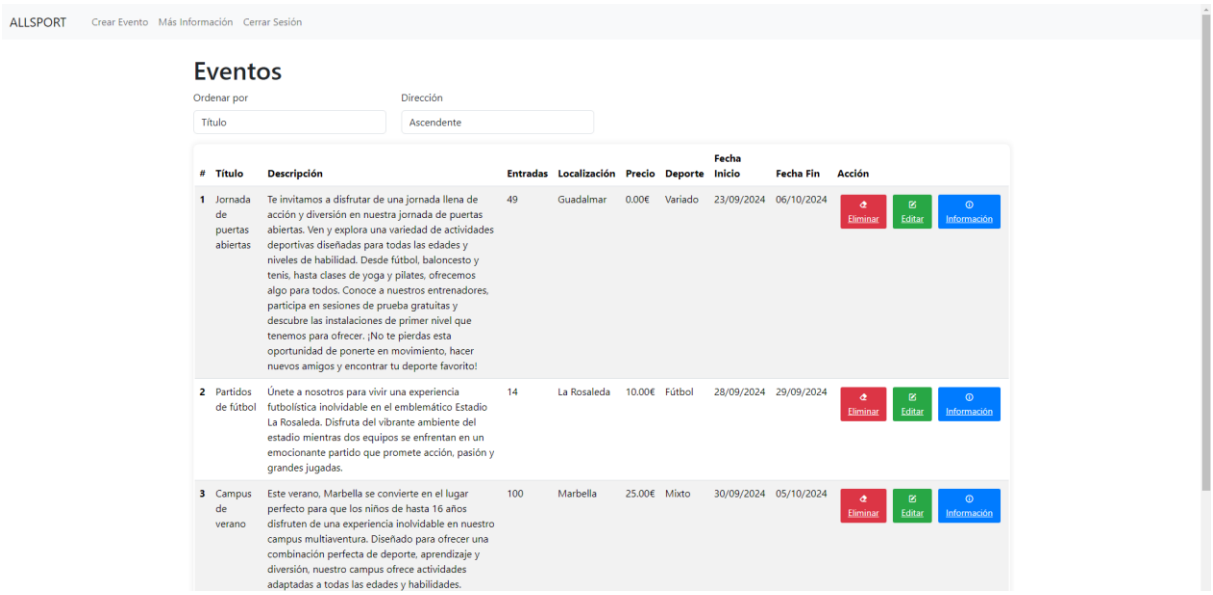

Figura 21. Página de ver eventos de las empresas

3.4.10 Editar evento

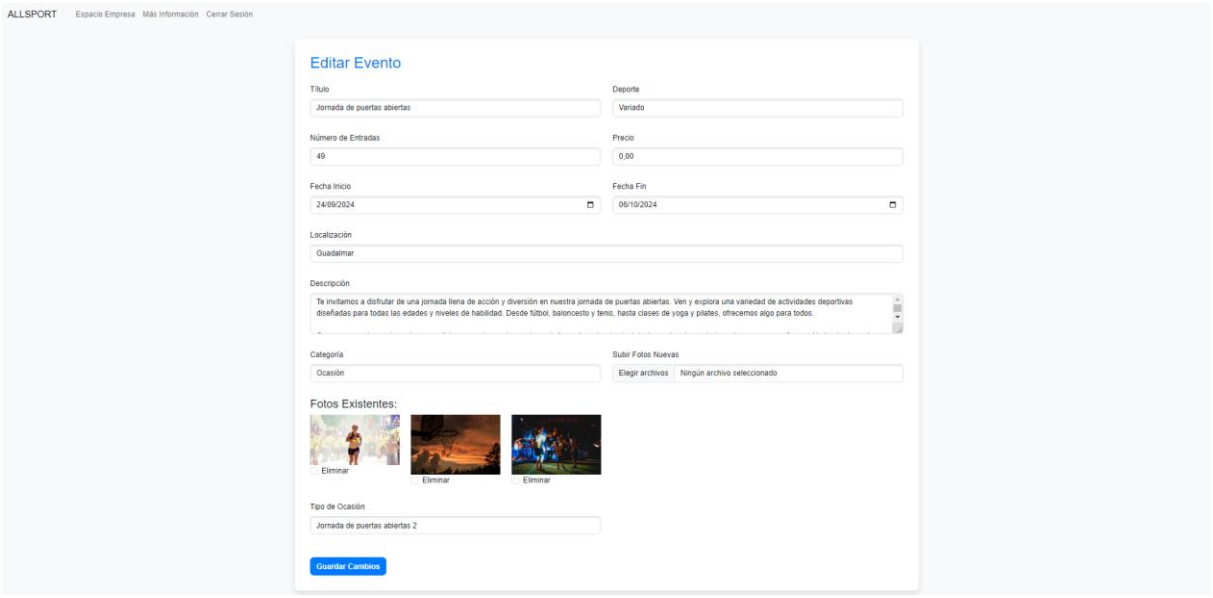

Figura 22. Página de edición de eventos

4

Implementación

4.1 Arquitectura Software

La implementación de este proyecto se estructuró en varias fases clave, cada una diseñada para garantizar la funcionalidad y seguridad de la aplicación de gestión de eventos deportivos. A continuación, se describen las principales componentes de la arquitectura del software, incluyendo las bibliotecas, controladores, middleware y la estructura del proyecto.

4.1.1 Bibliotecas

Las bibliotecas son conjuntos de códigos que los desarrolladores pueden reutilizar para realizar sus tareas. Durante el desarrollo de la aplicación se han usado las siguientes bibliotecas de JavaScript y Nodejs. Todas ellas han sido implementadas en:

```
1  const express = require('express');
2  const app = express();
3  const session = require('express-session');
4  const dotenv = require('dotenv');
5  const cookieParser = require('cookie-parser');
6  const bodyParser = require('body-parser');
7  const multer = require('multer');
8  const path = require('path');
```

Figura 23. Instalación de bibliotecas

- **Express**

Express es un framework minimalista y flexible de Node.js que proporciona un conjunto robusto de características para aplicaciones web y móviles. Simplifica la gestión de rutas, el manejo de solicitudes y respuestas HTTP, permitiendo crear aplicaciones de manera rápida y eficiente.

- **Sequelize**

Sequelize es un ORM (Object-Relational Mapping) para Node.js que soporta varios dialectos de SQL, incluyendo MySQL, PostgreSQL y SQLite. Este ORM simplifica la interacción con la base de datos, permitiendo definir modelos y relaciones entre tablas de manera sencilla y eficiente.

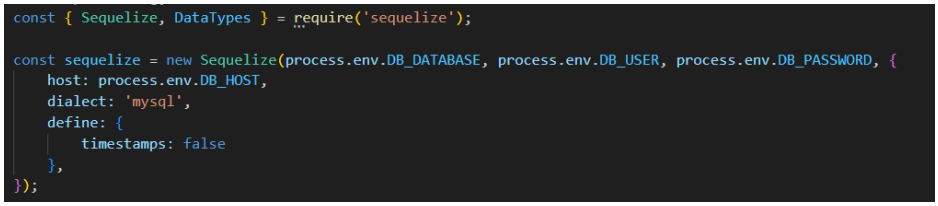

Figura 24. Instalación de Sequelize

- **Jsonwebtoken**

Jsonwebtoken es una biblioteca para crear y verificar JSON Web Tokens (JWT). Estos tokens se usan en la gestión de autenticación y autorización en la aplicación, proporcionando una manera segura de transmitir información entre el cliente y el servidor.

- **Bcrypt**

Bcrypt se utiliza para el hashing de contraseñas, añadiendo una capa adicional de seguridad a la gestión de credenciales de usuario. Esto asegura que las contraseñas almacenadas en la base de datos estén cifradas.

- **Express-session**

Express-session permite almacenar información de sesión en el servidor y usar cookies para rastrear sesiones en el cliente. Esto es esencial para gestionar la autenticación y mantener el estado de la sesión del usuario a lo largo de su interacción con la aplicación.

- **Dotenv**

Facilita la configuración de aplicaciones al permitir el uso de variables de entorno definidas en un archivo, manteniendo la configuración y las credenciales fuera del código fuente.

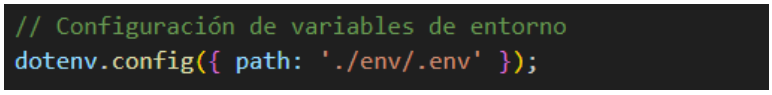

Figura 25. Instalación de bibliotecas

- **CookieParser**

CookieParser permite acceder a cookies en las solicitudes, facilitando la gestión de autenticación y personalización de la experiencia del usuario. Esta biblioteca es esencial para leer y escribir cookies de manera sencilla.

- **BodyParser**

BodyParser analiza cuerpos de solicitudes HTTP entrantes en diferentes formatos (JSON, URL-encoded) y los hace accesibles a través de **req. body**. Esto simplifica la manipulación de datos enviados por el cliente en las solicitudes HTTP.

- **Multer**

Facilita la subida de archivos al servidor, permitiendo especificar opciones de almacenamiento y nombres de archivo.

Figura 26. Instalación de bibliotecas

Como se puede apreciar en la figura se utiliza como almacenamiento **diskStorage**, el cual permite almacenar los archivos en el sistema de ficheros local. Este viene dado en destination y es la carpeta **uploads**, en cuanto al **filename** tiene como objetivo renombrar los archivos subidos para así evitar colisiones de nombres repetidos, el nombre será la union de la fecha con el nombre del archivo.
Por último, de define la constante **upload** para poder usarla como middleware.

- **Path**

Path proporciona utilidades para manipular y resolver rutas de archivos, asegurando que las rutas sean compatibles en diferentes sistemas operativos. Esto es crucial para gestionar rutas de manera segura y eficiente en un entorno multiplataforma.

- **Nodemailer**

Nodemailer (Figura 27) es una biblioteca de Node.js que simplifica el envío de correos electrónicos mediante SMTP. Compatible con cualquier servicio de correo electrónico que soporte SMTP, ofrece autenticación y conexiones seguras a través de TLS/SSL, siendo ideal para confirmaciones, restablecimientos de contraseñas y notificaciones en aplicaciones web.

```
const nodemailer = require('nodemailer');
const path = require('path');

const transporter = nodemailer.createTransport({
    service: 'gmail',
    host: process.env.EMAIL_HOST,
    port: 465,
    secure: true,
    auth: {
        user: process.env.EMAIL_USER,
        pass: process.env.EMAIL_PASSWORD
    }
});
```

Figura 27. Controladores

La configuración se hace mediante el transporte que no es más que una definición de cómo se enviaran los correos, indicando el servicio, host, puerto, secure y auth. Este último deberá contener el usuario y contraseña del email emisor.

4.1.2 Controladores

Son los componentes en una aplicación que gestionan la lógica de negocio, actuando como intermediarios entre la vista (interfaz de usuario) y el modelo (datos), procesando las solicitudes del usuario y actualizando la interfaz con los resultados.
Para la correcta funcionalidad y seguridad de la aplicación, se han implementado diversos controladores y middleware. A continuación, se detalla cada uno de ellos y su propósito.

```
// Importación de modelos y controladores
const { sequelize, Usuario, Evento, EventoPartido, EventoClase, EventoCampus, EventoOcasion, FotoEvento } = require('./database/sequelize-config');
const authController = require('./public/controladores/authController');
const userController = require('./public/controladores/userController');
const companyController = require('./public/controladores/companyController');
const verificacionToken_jwt = require('./public/controladores/jwtMiddleware');
const eventoController = require('./public/controladores/eventoController');
```

Figura 28. Controladores

- **AuthController**

  Este módulo gestiona la autenticación de usuarios, verificando credenciales y generando tokens JWT durante el proceso de inicio de sesión. Utiliza bcrypt para comparar contraseñas encriptadas y asegura que el usuario esté verificado antes de acceder. Dependiendo del rol del usuario (empresa, persona física o administrador), redirige a la interfaz correspondiente. Además, guarda el token en la sesión para poder mantener la autenticación en todo momento.

- **UserController y CompanyController**

Este módulo gestiona la autenticación de usuarios, tanto para empresas como para personas físicas, abarcando procesos como el registro y recuperación de contraseñas. Sus funciones principales incluyen: **registrar** nuevos usuarios y empresas en la plataforma; **enviar correos** electrónicos con enlaces para restablecer contraseñas a través de la función de recuperación de contraseñas **y validación de la cuenta**; y mostrar la página específica para restablecer la contraseña.

- **EventoController**

  Se gestiona las operaciones relacionadas con eventos, permitiendo su creación, edición, eliminación y consulta. Sus funciones principales incluyen: **GuardarEvento**, que almacena un nuevo evento en la base de datos; **ObtenerEventos**, que recupera todos los eventos para mostrarlos en la interfaz de usuario; **EliminarEvento**, que borra un evento específico; **ActualizarEvento**, que modifica los detalles de un evento existente; **SubirFoto**, que maneja la subida de fotos asociadas a un evento; ComprarEntrada, que permite a los usuarios adquirir entradas; y **ObtenerEventoPorId**, que recupera la información de un evento específico usando su ID.

4.1.3 Middleware

Son los programas o funciones que actúan como intermediarios en el flujo de datos entre el servidor y la aplicación, gestionando tareas como la autenticación, la autorización, el manejo de errores o la gestión de sesiones antes de que la solicitud llegue a los controladores.

- **JwtMiddleware:** Verifica los tokens JWT para autenticación y autorización.
- **RoleMiddleware:** Verifica el rol del usuario para otorgar acceso a diferentes partes de la aplicación.

4.1.4 Estructura del código fuente

La estructura del proyecto se ha diseñado para mantener el código organizado, limpio y fácil de entender. Como se puede apreciar en la figura 29, cada directorio contiene sus archivos y funcionalidades.

- **Database**: este directorio contiene archivos relacionados con la configuración y la conexión a la base de datos. En este caso es Sequelize-config el cual incluye todos los detalles para la conexión con la Base de Datos, al igual que todas sus tablas y relaciones.

- **Documentación**: este directorio se reserva para almacenar documentación del proyecto.

- **Env**: contiene el archivo env, el cual incluye variables de entorno, tales como credenciales de la base de datos y claves secretas, para mantener esta información segura y fuera del código fuente.

- **Node Modules**: directorio generado automáticamente por NPM, que contiene todas las dependencias y módulos necesarios para que la aplicación funcione correctamente.

- **Public**: Este directorio contiene archivos públicos accesibles desde el navegador. A su vez incluye las carpetas de controladores con todos los archivos JavaScript que manejan la ruta y lógica de la aplicación. CSS, todos los archivos para el diseño de la aplicación. JavaScript, esta carpeta almacena los archivos usados en el frontend.

- **Uploads**: directorio destinado al almacenamiento de las imágenes de los eventos.

- **App.js**: Archivo principal de la aplicación que inicializa el servidor, configura middleware y define las rutas principales.

- **Package-lock.json**: Archivo generado automáticamente que describe la estructura exacta de las dependencias de NPM instaladas.

- **Package.json**: Archivo de configuración de NPM que contiene información sobre el proyecto, como el nombre, la versión, las dependencias y los scripts de NPM.

- **README.md**: Archivo de texto que proporciona una descripción general del proyecto, instrucciones de instalación, uso y cualquier otra información relevante para desarrolladores y usuarios.

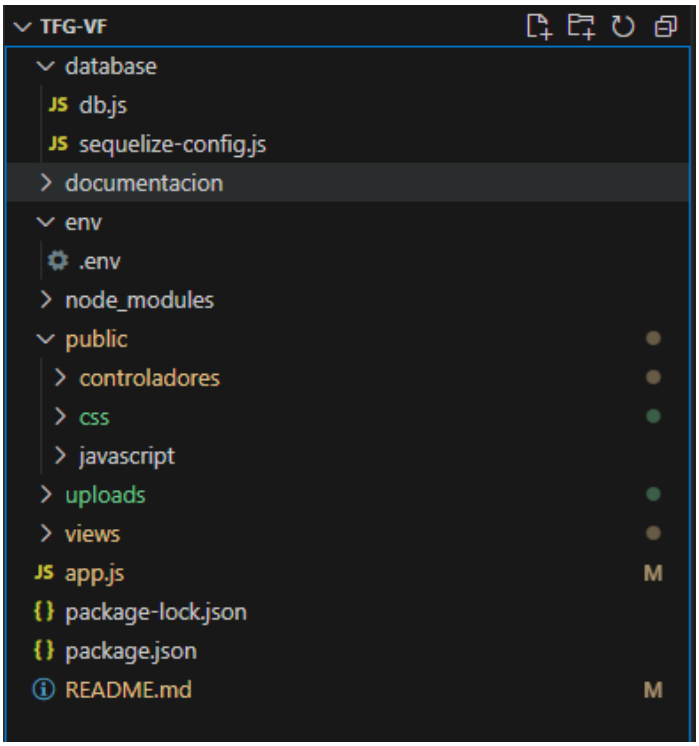

Figura 29. Estructura del código fuente

4.2 Seguridad

La seguridad es un aspecto crucial en el desarrollo de cualquier aplicación, especialmente cuando se manejan datos sensibles como la información de usuarios y las transacciones. Para garantizar un entorno seguro, se implementan diversas estrategias y herramientas que protegen tanto la integridad de los datos como la autenticidad de los usuarios. Entre estas, el uso de **JSON Web Tokens (JWT)** es fundamental para la autenticación y autorización, permitiendo a los usuarios acceder de manera segura a las funcionalidades del sistema sin exponer sus credenciales. Además, la seguridad en la interacción con bases de datos se fortalece mediante prácticas como el uso de **consultas SQL parametrizadas**, que previenen inyecciones SQL, un tipo de ataque común en aplicaciones web. Finalmente, la gestión segura de configuraciones y credenciales del sistema se realiza a través de **variables de entorno**, almacenadas en archivos .env, lo que evita la exposición directa de información crítica en el código fuente. Estas medidas en conjunto forman una capa de protección integral, asegurando que la aplicación opere de manera segura y confiable.

4.2.1 Variables de entorno

Permiten mejorar la configuración de la aplicación y seguridad, manteniendo la configuración sensible y credenciales fuera del código fuente, evitando que la información crítica se exponga.
Se ha implementado la biblioteca **dotenv** se usa para cargar las variables de entorno desde un archivo**. env** en el entorno de ejecución de Node.js. Este archivo contiene claves de configuración como contraseñas de bases de datos, claves secretas para JWT, y otros parámetros de configuración sensibles.
Los beneficios que aporta son:

- Mantenimiento de credenciales y configuraciones sensibles fuera del código.
- Posibilidad de cambiar la modificación sin modificar el código.
- Facilita la configuración en diferentes entornos (desarrollo, prueba, producción).

4.2.2 SQL

Se ha optado por el uso de Sequelize, un ORM de Node.js, que proporciona una capa de abstracción sobre SQL, permitiendo interactuar con la base de datos de manera más segura y eficiente.

Se ha creado un archivo de conexión con la base de datos donde se han llamado campos como DB_DATABASE, DB_USER, DB_PASSWORD, los cuales se encuentran definidos en el archivo ENV.

```
const { Sequelize, DataTypes } = require('sequelize');

const sequelize = new Sequelize(process.env.DB_DATABASE, process.env.DB_USER, process.env.DB_PASSWORD, {
    host: process.env.DB_HOST,
    dialect: 'mysql',
    define: {
        timestamps: false
    },
});
```

Figura 30. Sequelize

Para cada entidad hay que definir una constante que esté enlazada a la tabla de la base de datos. Como se puede apreciar en la imagen, se replica la tabla con sus relaciones poniendo todos sus campos y tipo de datos. Ver Figura 31.

```
const Persona = sequelize.define('Persona', {
    id_persona: {
        type: DataTypes.INTEGER,
        primaryKey: true,
        autoIncrement: true,
    },
    usuario_id: {
        type: DataTypes.INTEGER,
        allowNull: false,
        references: {
            model: 'Usuario',
            key: 'id_usuario',
        }
    },
    nombre: {
        type: DataTypes.STRING,
        allowNull: false,
    },
    apellido: {
        type: DataTypes.STRING,
        allowNull: false,
    },
    email: {
        type: DataTypes.STRING,
        allowNull: false,
    },
}, {
    tableName: 'persona'
});
```

Figura 31. Estructura de Sequelize para las tablas

Finalizando con la configuración del archivo hay que tener en cuenta que se deben replicar las relaciones entre las tablas.

```
Usuario.hasOne(Persona, { foreignKey: 'usuario_id' });
Usuario.hasOne(Admin, { foreignKey: 'usuario_id' });
Usuario.hasOne(Compania, { foreignKey: 'usuario_id' });

Persona.belongsTo(Usuario, { foreignKey: 'usuario_id' });
Admin.belongsTo(Usuario, { foreignKey: 'usuario_id' });
Compania.belongsTo(Usuario, { foreignKey: 'usuario_id' });

Compania.hasMany(Evento, { foreignKey: 'id_compania' });
Evento.belongsTo(Compania, { foreignKey: 'id_compania' });
```

Figura 32. Estructura de Sequelize con las relaciones

4.2.3 JWT

Los JSON Web Tokens (JWT) se utilizan para la gestión de autenticación y autorización en la aplicación. Los JWT son un método seguro y estandarizado para transmitir información entre el cliente y el servidor.

Sus beneficios son:

- Los tokens firmados garantizan que los datos no han sido manipulados.
- Los tokens son autocontenidos, lo que facilita la gestión de sesiones en aplicaciones distribuidas.
- Reduce la carga en el servidor al no tener que mantener sesiones en la memoria del servidor.

Su estructura es:

- **Header**: consta de dos partes, el algoritmo de firma y tipo de token.
- **Payload**: contiene las reclamaciones o el objeto JSON.
- **Signature**: una cadena que se genera a través de un algoritmo criptográfico que se usa para verificar la integridad de la carga útil del JSON.

4.3 Interfaz con la base de datos

En el desarrollo de la aplicación, se optó por utilizar MySQL como base de datos relacional y Sequelize como ORM (Object-Relational Mapping) para la interacción con esta. Los factores que se tuvieron en cuenta fueron:

- **Robustez y fiabilidad de MySQL**
  MySQL es una base de datos relacional confiable y ampliamente utilizada, ideal para gestionar datos de manera segura y eficiente en una aplicación de gestión de eventos.
- **Escalabilidad**
  MySQL puede manejar un aumento en la cantidad de datos y usuarios sin afectar el rendimiento, lo que es crucial para el crecimiento de la aplicación.
- **Migraciones y Gestión de Esquemas**
  Sequelize ofrece un sistema de migraciones que permite modificar la estructura de la base de datos de manera controlada, facilitando el trabajo colaborativo.
- **Seguridad y Autenticación**
  La combinación de Sequelize y MySQL facilita la implementación de mecanismos de seguridad, como la autenticación de usuarios y el cifrado de datos, mejorando la protección de la aplicación.

4.4 Estructura del Front-end y Back-end

La arquitectura de la aplicación está dividida en dos partes, el frontend y backend. El **frontend** se encarga de la interfaz del usuario para hacer la experiencia interactiva y atractiva para los usuarios. Están diseñadas para permitir la navegación fluida a través de las distintas funcionalidades como son el registro, creación de eventos y compra de entradas. Cada vista se encuentra conectada al backend mediante solicitudes HTTP, lo que facilita la comunicación con la base de datos .
En cambio, el **backend** es el encargado de la lógica, como son las operaciones necesarias con la base de datos o la autenticación de los usuarios.
A continuación, se muestran los ejemplos más representativos.

4.4.1 Registro y verificación de cuenta

En el HTML los datos son enviados al backend mediante la solicitud POST al end point registró nuevo. Una vez relleno los campos del formulario se envía al backend para su procesamiento.

```
<div class="registro">
    <form action="registro_nuevo" method="POST" id="registroForm">
        <input type="text" name="user" required id="usuario_input">
        <label for="user" id="usuario_label">Usuario</label>
    </div>

    <div class="div">
        <input type="password" name="pass" required id="password_input">
        <label for="pass" id="password_label">Contraseña</label>
    </div>

    <div class="div">
        <input type="password" name="pass2" required id="password_confirm_input">
        <label for="pass2" id="password_confirm_label">Confirmar Contraseña</label>
    </div>

    <div class="div">
        <select class="form-select" name="role" id="rol_select" required onchange="mostrarCampos()">
            <option value="">TIPO DE USUARIO</option>
            <option value="user">Usuario individual</option>
            <option value="company">Empresa</option>
        </select>
    </div>
```

Figura 33. HTML de registro

En el backend llega a la función registro_usuario, la cual contiene los mismos valores que en el HTML. Se realizan las comprobaciones necesarias para poder efectuar el registro, como son las contraseñas introducidas o que el correo no estuviera registrado. Una vez terminado se procede a hacer un hash de la contraseña para almacenarla en la base de datos. Al comienzo de cada archivo se llama mediante contante a las librerías necesarias, bases de datos o métodos de otras clases.

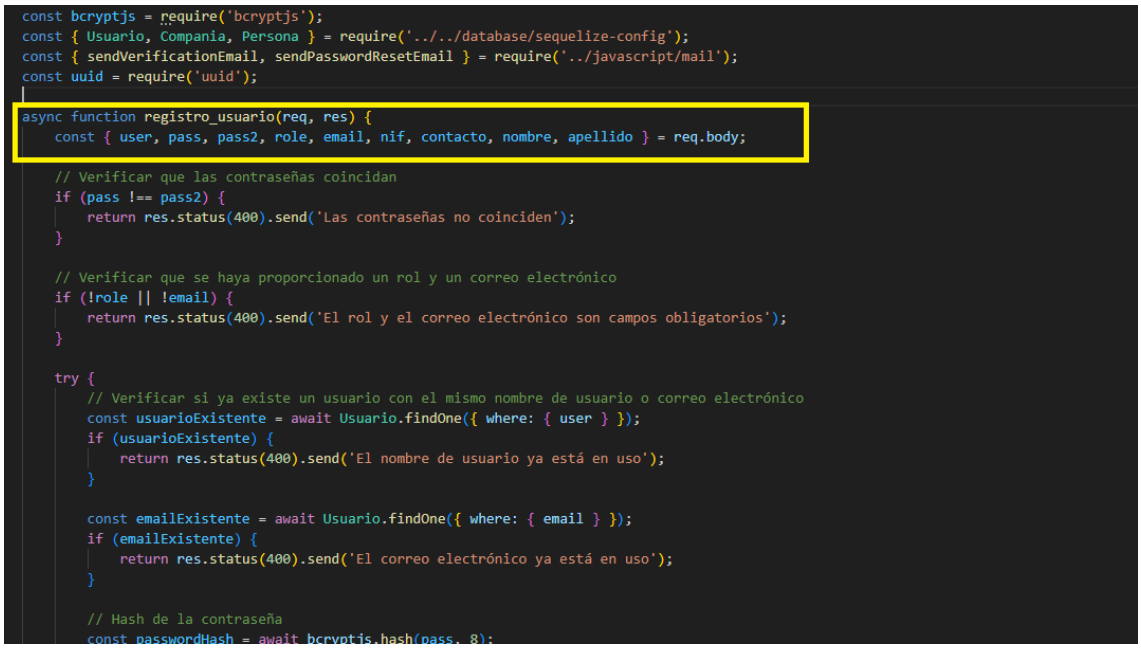

Figura 34. UserController para registro

También, en el frontend el usuario pude indicar el rol, por lo que en el backend se deberá tratar. En este caso tenemos que en función de la opción que tome los valores se almacenaran en la tabla correspondiente y finalmente llegará un correo de verificación de cuenta al usuario. Esta última función se ha tenido que declarar previamente en la cabecera del archivo.

Figura 35. UserController para registro, parte del rol y envío de correo

Una llegado al envío del correo como se indica al comienzo del archivo, la lógica del backend hará una parada para llamar al método de verificación. Este está compuesto por el email del usuario y el token de este para que sea único. La función del HTML será cargar una plantilla con toda la información requerida para validar el registro.

```
const nodemailer = require('nodemailer');
const path = require('path');

const transporter = nodemailer.createTransport({
    service: 'gmail',
    host: process.env.EMAIL_HOST,
    port: 465,
    secure: true,
    auth: {
        user: process.env.EMAIL_USER,
        pass: process.env.EMAIL_PASSWORD
    }
});

const sendVerificationEmail = async (email, token) => {
    try {
        return await transporter.sendMail({
            from: process.env.EMAIL_USER,
            to: email,
            subject: "Activación de cuenta en ALLSPORT",
            html: getTemplate(token),
        });
    } catch (error) {
        console.log('ERROR CON EMAIL:', error);
    }
};
```

Figura 36. Envío de correo

Para finalizar el proceso se procederá dentro del backend a hacer la verificación de la activación de la cuenta. Se extrae el token de la URL y se busca en la base de datos al usuario que tenga asociado el token recibido. Se usa **Usuario.findOne** para hacerlo dentro de la tabla usuario.

```
async function verificacionCuenta(req, res) {
    const { token } = req.params;
    try {
        const usuario = await Usuario.findOne({ where: { token } });
        if (!usuario) {
            return res.status(400).send('Token de verificación inválido');
        }
        await Usuario.update({ verified: true, token: null }, { where: { id_usuario: usuario.id_usuario } });
        res.send('Usuario verificado correctamente');
    } catch (error) {
        console.error(error);
        res.status(500).send('Error interno del servidor');
    }
}
```

Figura 37. Verificación de usuario

4.4.2 Creación de eventos

En el HTML se rellena el formulario de crear eventos con todos los campos que son solicitados.

Figura 38. Formulario de creación de eventos

- **FormData**: Este objeto captura todos los datos del formulario, incluyendo los archivos seleccionados.
- **Fetch API**: Envía una solicitud POST al servidor con los datos del formulario en el cuerpo de la solicitud.
- **Modal de Confirmación**: Si la creación del evento es exitosa, se muestra un modal para notificar al usuario.

Figura 39. Formulario de creación de eventos

El controlador correspondiente guardarEvento se encarga de procesar los datos y crear el evento en la base de datos

```
116  app.post('/api/eventos/crear', verificacionToken_jwt(['admin', 'company']), upload.array('fotos', 10), eventoController.guardarEvento);
```

Figura 40. Ruta de controlador guardar evento

```
public > controladores > JS eventoControllers.js > guardarEvento
1   const { sequelize, Usuario, Evento, EventoClase, EventoPartido, EventoCampus, EventoOcasion, FotoEvento } = require('../../database/sequelize-config');
2   const path = require('path');
3   const fs = require('fs');
4   const { sendEntradas } = require('../javascript/mail');
5
6   async function guardarEvento(req, res) {
7       const { titulo, descripcion, numero_entradas, localizacion, precio, categoria, deporte, fecha_inicio, fecha_fin, ...detallesCategoria } = req.body;
8       const id_compania = req.session.user.companiaId;
9
10      console.log('Datos recibidos:', req.body);
11
12      if (!id_compania) {
13          return res.status(400).send('El id de la compañía es obligatorio');
14      }
15
16      try {
17          const evento = await Evento.create({
18              titulo,
19              descripcion,
20              numero_entradas,
21              localizacion,
22              precio,
23              deporte,
24              fecha_inicio,
25              fecha_fin,
26              id_compania
27          });
```

Figura 41. Controlador de guardar evento

- **Validación**: se aseguran de que todos los campos requeridos estén presentes.
- **Interacción con la Base de Datos**: se crea un nuevo registro de evento en la base de datos utilizando Sequelize.
- **Respuesta al Front-End**: se responde con un mensaje de éxito o error según el resultado del proceso.

4.4.3 Compra de entradas

Como se puede ver en la figura 42, una vez creado el evento el usuario puede acceder a la vista donde ver la información del evento. Esta se encuentra compuesta por un apartado donde aparece la información común a todos y en función de la categoría los datos extras.

```
<div class="container mt-4">
    <h1>Información del Evento</h1>
    <div class="evento-info">
        <div class="evento-detalles">
            <p><strong>Título:</strong> <%= evento.titulo %></p>
            <p><strong>Descripción:</strong> <%= evento.descripcion %></p>
            <p><strong>Número de Entradas:</strong> <span id="numeroEntradas"><%= evento.numero_entradas %></span></p>
            <p><strong>Localización:</strong> <%= evento.localizacion %></p>
            <p><strong>Precio:</strong> <%= evento.precio %></p>
            <p><strong>Deporte:</strong> <%= evento.deporte %></p>
            <p><strong>Fecha de Inicio:</strong> <%= evento.fecha_inicio %></p>
            <p><strong>Fecha de Fin:</strong> <%= evento.fecha_fin %></p>
            <% if (categoria === 'clase') { %>
                <p><strong>Instructor:</strong> <%= evento.EventoClase ? evento.EventoClase.instructor : '' %></p>
                <p><strong>Duración:</strong> <%= evento.EventoClase ? evento.EventoClase.duracion : '' %></p>
                <p><strong>Nivel:</strong> <%= evento.EventoClase ? evento.EventoClase.nivel : '' %></p>
            <% } else if (categoria === 'partido') { %>
                <p><strong>Equipo Local:</strong> <%= evento.EventoPartido ? evento.EventoPartido.equipo_local : '' %></p>
                <p><strong>Equipo Visitante:</strong> <%= evento.EventoPartido ? evento.EventoPartido.equipo_visitante : '' %></p>
                <p><strong>Liga:</strong> <%= evento.EventoPartido ? evento.EventoPartido.liga : '' %></p>
            <% } else if (categoria === 'campus') { %>
                <p><strong>Programa:</strong> <%= evento.EventoCampus ? evento.EventoCampus.programa : '' %></p>
            <% } else if (categoria === 'ocasion') { %>
                <p><strong>Tipo de Ocasion:</strong> <%= evento.EventoOcasion ? evento.EventoOcasion.tipo_ocasion : '' %></p>
            <% } %>
            <div class="comprar-boton">
                <input type="hidden" id="eventoId" value="<%= evento.id %>">
                <button id="comprarButton" class="btn btn-primary btn-lg">Comprar</button>
            </div>
        </div>
```

Figura 42. Vista de eventos

Una vez pulsado el botón de comprar se realiza una solicitud Ajax donde se envía una solicitud 'POST' al servidor para la URL con el id del evento que se quiere comprar. Finalmente, con las funciones de **success** y **error** se analizará si la compra ha sido relazada con éxito o no.

```
<script>
    $(document).ready(function() {
        $('#comprarButton').click(function() {
            const eventId = $('#eventoId').val();
            $.ajax({
                url: `/api/comprar_entrada/${eventId}`,
                type: 'POST',
                success: function(response) {
                    const messageModal = new bootstrap.Modal(document.getElementById('messageModal'));
                    const messageContent = document.getElementById('messageContent');
                    messageContent.textContent = response.message;
                    messageContent.className = 'alert alert-success';
                    messageModal.show();

                    $('#numeroEntradas').text(function(i, oldVal) {
                        return oldVal - 1;
                    });
                },
                error: function(xhr) {
                    const errorResponse = xhr.responseJSON ? xhr.responseJSON.error : 'Error al comprar la entrada';
                    const messageModal = new bootstrap.Modal(document.getElementById('messageModal'));
                    const messageContent = document.getElementById('messageContent');
                    messageContent.textContent = errorResponse;
                    messageContent.className = 'alert alert-danger';
                    messageModal.show();
                }
            });
        });
    });
</script>
```

Figura 43. Envío de formulario para la compra

Todo este proceso tiene también el proceso de comprar entrada en el controlador, este realiza las comprobaciones necesarias como ver si el evento existe o si quedan entradas disponibles. Además del id del evento también se tiene en cuenta el email del usuario que este haciendo la compra para poder enviarle la entrada. Ver figura 45.

```
234  async function comprarEntrada(eventId, email) {
235      try {
236          const evento = await Evento.findByPk(eventId);
237          if (!evento) {
238              console.log('Evento no encontrado');
239              return { error: 'Evento no encontrado', status: 404 };
240          }
241          if (evento.numero_entradas > 0) {
242              evento.numero_entradas -= 1;
243              await evento.save();
244              console.log('Compra realizada con éxito');
245              await sendEntradas(email);
246              return { message: 'Compra realizada con éxito', status: 200 };
247          } else {
248              console.log('No hay entradas disponibles');
249              return { error: 'No hay entradas disponibles', status: 400 };
250          }
251      } catch (error) {
252          console.error('Error al comprar entrada:', error);
253          return { error: 'Error al comprar entrada', status: 500 };
254      }
255  }
```

Figura 44. Controlador de compra de entradas

5

Conclusiones

5.1 Conclusiones

Desde el punto de vista personal, el desarrollo de este proyecto ha sido una experiencia enriquecedora, permitiendo aplicar y consolidar los conocimientos teóricos y prácticos adquiridos a lo largo de la carrera. La creación de una aplicación para la gestión de eventos deportivos no solo representó un desafío técnico significativo, al involucrar tecnologías que no habían sido exploradas previamente en mi formación académica, sino que también se presentó como una gran oportunidad de mejorar mis habilidades en programación, diseño, gestión de proyectos y pensamiento lógico. A lo largo del proyecto me enfrenté diversos obstáculos, lo que fortaleció mi capacidad para resolver problemas de manera creativa y eficiente, adaptándome a nuevas circunstancias y adoptando un enfoque proactivo ante los desafíos.

Profesionalmente, este proyecto ha proporcionado un impulso hacia un manejo más avanzado de tecnologías modernas y metodologías ágiles de desarrollo, como Scrum. El uso de herramientas y frameworks como Node.js, Sequelize, JWT, y Bootstrap fue esencial para la creación de una aplicación robusta, segura y eficiente. Este proceso no solo me permitió profundizar en estas tecnologías, sino que también me proporcionó una experiencia práctica valiosa en la gestión de bases de datos y la implementación de medidas de seguridad avanzadas, competencias que serán de gran utilidad en futuros proyectos y en mi desarrollo profesional en el campo de la informática. La experiencia adquirida ha sido fundamental para encontrar trabajo, y me ha proporcionado una base sólida que será de gran utilidad en futuros proyectos y en mi desarrollo profesional en el campo de la informática.

Desde la perspectiva del proyecto, se han alcanzado los objetivos principales, como la centralización de la gestión de eventos deportivos en una plataforma única, escalable y fácil de usar. La aplicación desarrollada no solo facilita la creación, promoción y administración

de eventos, sino que también ofrece funcionalidades específicas para el sector deportivo, abordando y superando las limitaciones de otras plataformas disponibles en el mercado. Este enfoque especializado asegura no solo una mejor experiencia para los usuarios, sino también una mayor eficiencia en la gestión de eventos deportivos, contribuyendo al éxito general del proyecto.

Finalmente, gracias a la sólida base de conocimientos adquiridos durante el grado, he logrado enfrentar con éxito los desafíos inherentes al desarrollo de este proyecto. Esta experiencia ha profundizado mi comprensión de las complejidades que implica la gestión de eventos deportivos, y me ha permitido obtener una visión más amplia y crítica sobre cómo se pueden implementar soluciones tecnológicas efectivas en este ámbito. El aprendizaje continuo y la aplicación práctica de este conocimiento han sido fundamentales para alcanzar los resultados deseados y sentar una base sólida para futuros desarrollos en este campo.

5.2 Líneas futuras

La aplicación actual cumple con los requisitos establecidos y proporciona las funcionalidades necesarias para la gestión de eventos deportivos. Sin embargo, existe un amplio margen de mejora que se podría potenciar aún más la utilidad y la experiencia de usuario de la plataforma.

Una de las mejoras clave sería la implementación de un foro específico para cada evento. Este foro permitiría a los usuarios compartir sus opiniones, experiencias y sugerencias, fomentando una comunidad más activa y participativa. La retroalimentación obtenida a través de estos foros podría ser valiosa para los organizadores de eventos, ayudándoles a realizar mejoras en futuros eventos y a personalizar la oferta según las preferencias del público.

Otra línea de mejora sería la integración de sistemas de análisis de datos. Incorporar herramientas de análisis avanzadas permitiría obtener información detallada sobre el rendimiento de los eventos y las preferencias de los usuarios. Esta información podría utilizarse para optimizar la organización de futuros eventos, mejorar la segmentación de mercado y personalizar la experiencia del usuario.

Además, desarrollar una aplicación móvil complementaria sería un paso importante para ampliar el alcance de la plataforma. Una aplicación móvil facilitaría a los usuarios el acceso a la información y la gestión de eventos en cualquier momento y lugar, mejorando la conveniencia y aumentando la tasa de participación.

Bibliografía

- *BezKoder. (s.f.). "Node.js Rest APIs example with Express, Sequelize & MySQL." Recuperado el 2023, de https://www.bezkoder.com/node-js-express-sequelize-mysql*
- *Bootstrap Documentation. (s.f.). "Bootstrap 5." Recuperado el 2023, de https://getbootstrap.com/*
- *Dotenv GitHub Repository. (s.f.). "Dotenv - Loads environment variables from .env for Node.js projects." Recuperado el 2023, de https://github.com/motdotla/dotenv*
- *EJS Official Website. (s.f.). "EJS: Embedded JavaScript Templates." Recuperado el 2023, de https://ejs.co/*
- *GitHub. (s.f.). "REST API using Node.js, Express, Sequelize and MySQL + JWT Authentication and Authorization." Recuperado el 2023, de https://github.com/indraarianggi/nodejs-sequelize-mysql-api*
- *jQuery Official Website. (s.f.). "jQuery Documentation." Recuperado el 2023, de https://jquery.com/*
- *Multer GitHub Repository. (s.f.). "Express/Multer - Middleware for Handling multipart/form-data." Recuperado el 2023, de https://github.com/expressjs/multer*
- *MySQL Documentation. (s.f.). "MySQL 8.0 Reference Manual." Recuperado el 2023, de https://dev.mysql.com/doc/refman/8.0/en/*
- *bcrypt - npm*. (s.f.). Recuperado el 2022, de https://www.npmjs.com/package/bcrypt
- Ajax documentation de https://api.jquery.com/category/ajax/
- W3Schools. (s.f.). "JavaScript Reference." Recuperado el 2023, de https://www.w3schools.com/jsref/default.asp
- W3Schools. (s.f.). "SQL Tutorial." Recuperado el 2023, de https://www.w3schools.com/sql/
- Sequelize. (s.f.). "Getting Started with Sequelize." Recuperado el 2023, de https://sequelize.org/master/manual/getting-started.html
- Sequelize GitHub. (s.f.). "Sequelize Documentation." Recuperado el 2023, de https://github.com/sequelize/sequelize

Apéndice A

Manual de Instalación

Requisitos

- PC con conexión a internet sin importar el sistema operativo.
- Tener instalado XAMPP https://www.apachefriends.org/download.html
- Node https://nodejs.org/en/download/package-manager
- Se recomienda el uso de navegadores Google Chrome o Edge
- Uso de Visual Studio Code https://code.visualstudio.com/download

Base de datos

- Descargar desde GitHub la carpeta de XAMPP y copiarla en la que se ha creado durante la instalación.

https://github.com/alvarosc2000/TFG-VF

- Desde la interfaz de XAMPP con pulsar el botón de start y el de admin redirigirá automáticamente al gestor de la base de datos en el que se encontraran todas las tablas .

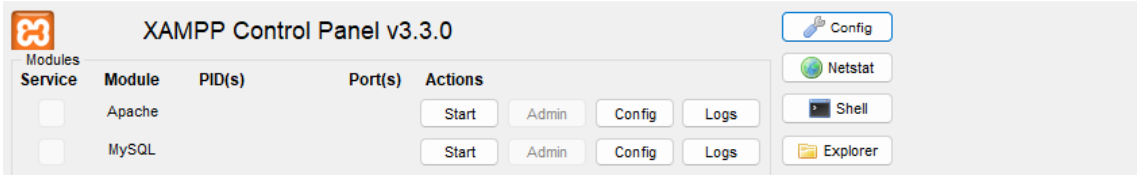

Figura 45. Interfaz de XAMPP

Servidor

- De la misma manera que se inicializa la base de datos en el modulo de Apache con iniciarlo es suficiente.

Cliente

- Una vez descargado el proyecto y cargado en Visual Studio Code con poner el comando en terminal " **node app**" se ejecutará la aplicación en la siguiente dirección: http://localhost:4000/
- Para comprobar las funcionales puede acceder a cada rol con las siguientes credenciales.
    - admin (usuario) admin (contraseña)
    - us1 (usuario) us1(contraseña)
    - empresa1 (usuario) empresa1 (contraseña)

Apéndice B

Manual de Usuario

Inicio

La primera vista de un usuario al entrar en la página la compondrán los apartados de:

- **Únete**: en el cual un usuario podrá registrarse.
- **¿Quiénes somos?:** incorpora una descripción sobre la empresa y su localización.
- **Inicio de sesión:** si el usuario se encuentra registrado podrá acceder con sus credenciales.

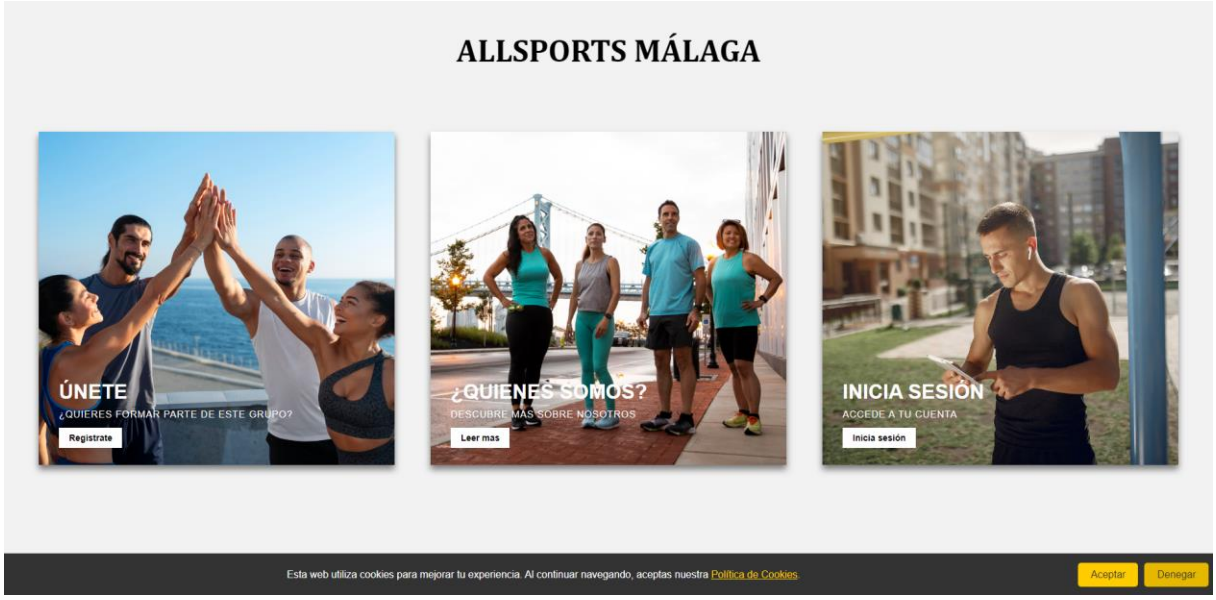

Figura 46. Página inicial

Política Cookies

Si accede a la política de cookies le redirigirá a la siguiente vista donde se explican que son y como ponerse en contacto en caso de dudas.

Figura 47. Página de explicación de cookies

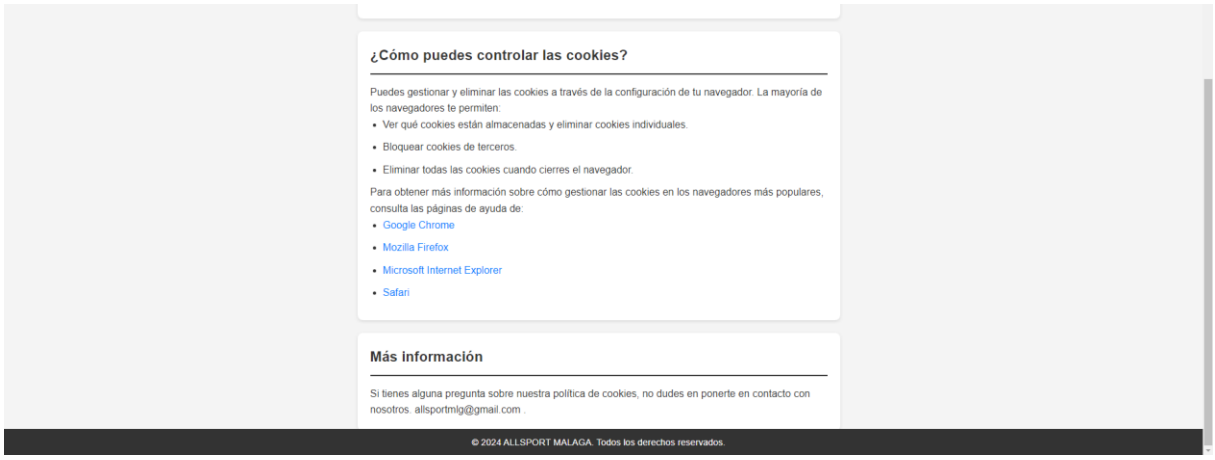

Figura 48. Página de explicación de cookies

Registro

Si el usuario no está registrado podrá hacerlo rellenando el formulario y aceptando la **política de privacidad**, la cual podrá consultar en cualquier momento accediendo al enlace.

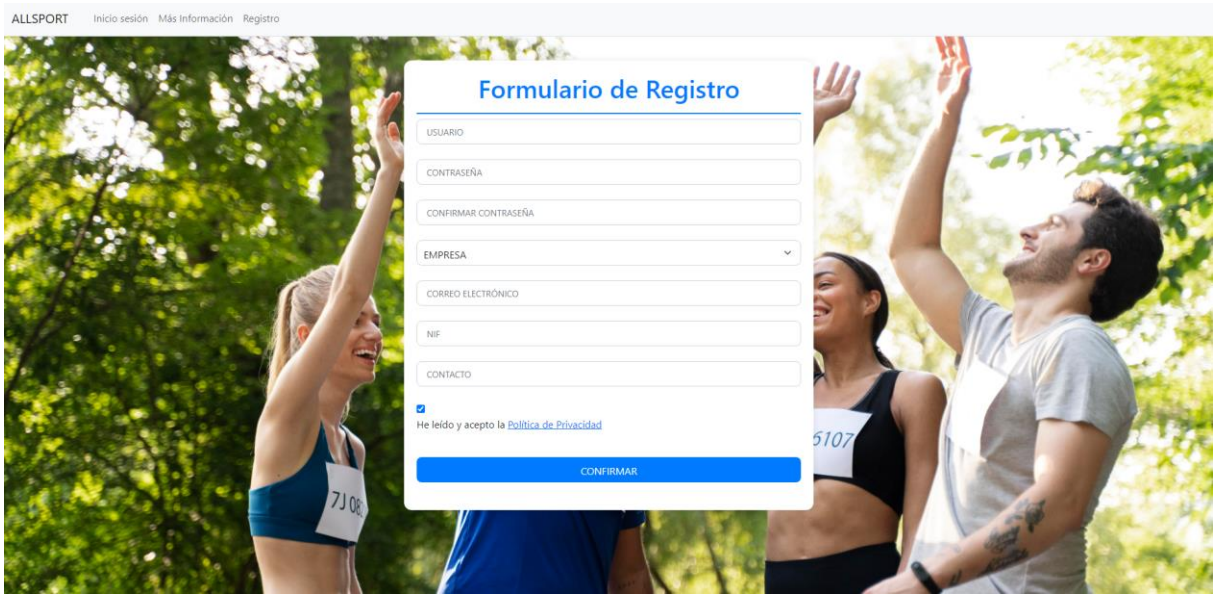

Figura 49. Página de registro

Política de privacidad

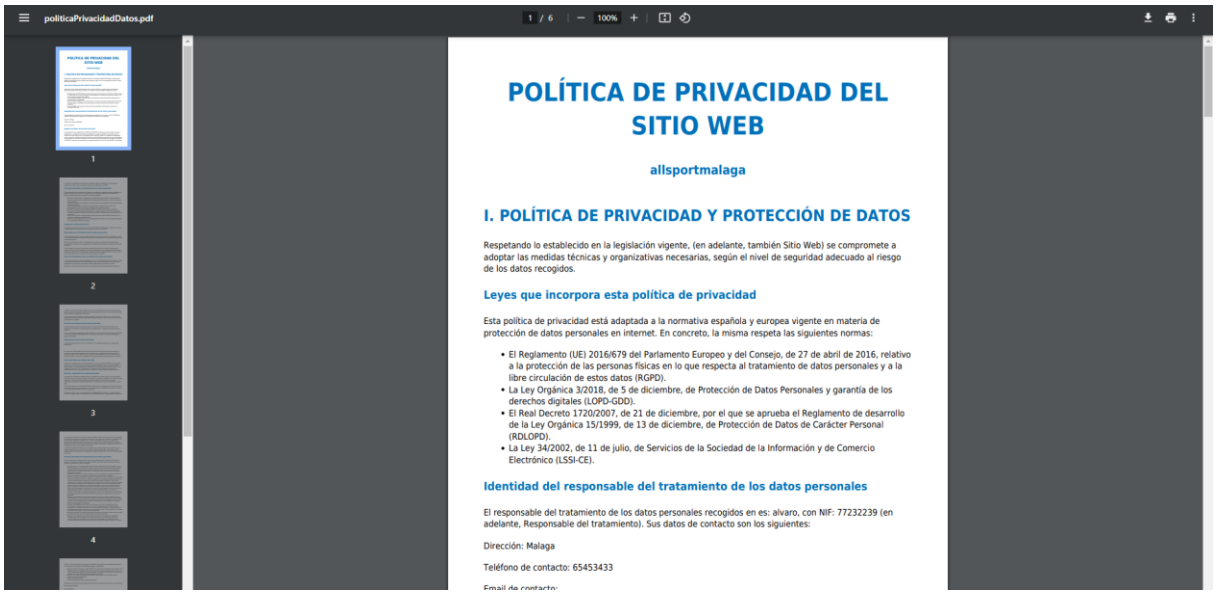

Figura 50. Página de política de privacidad

¿Quiénes somos?

En esta vista los usuarios podrán acceder sin importar si están registrados o no y ver toda la información sobre la empresa, desde las instalaciones, equipos hasta la localización.

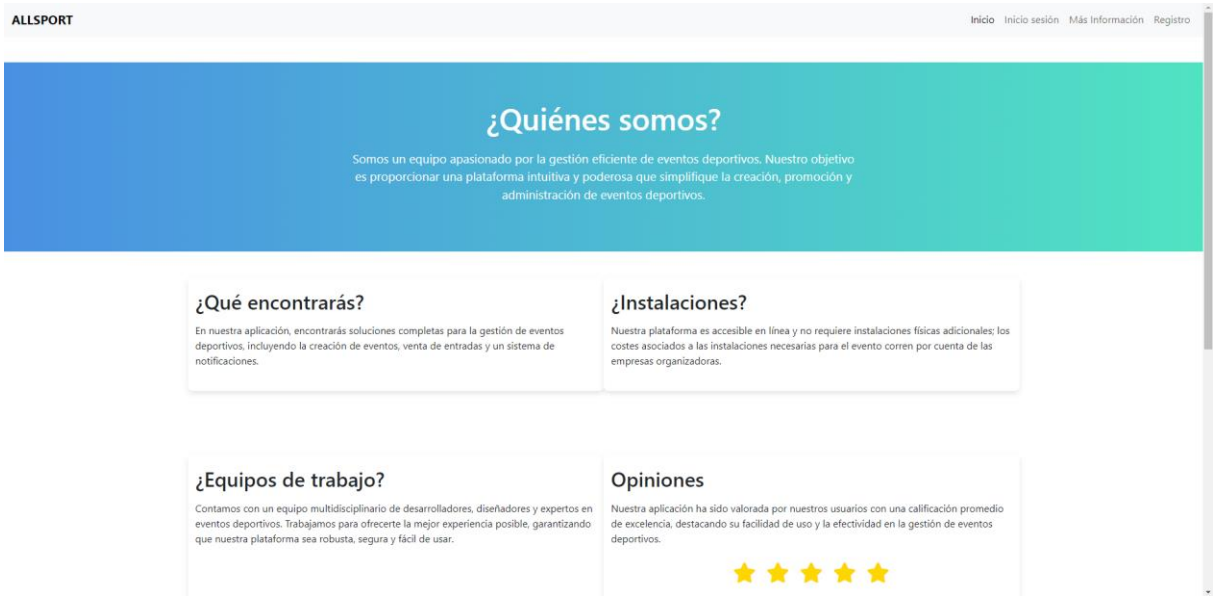

Figura 51. Página de información

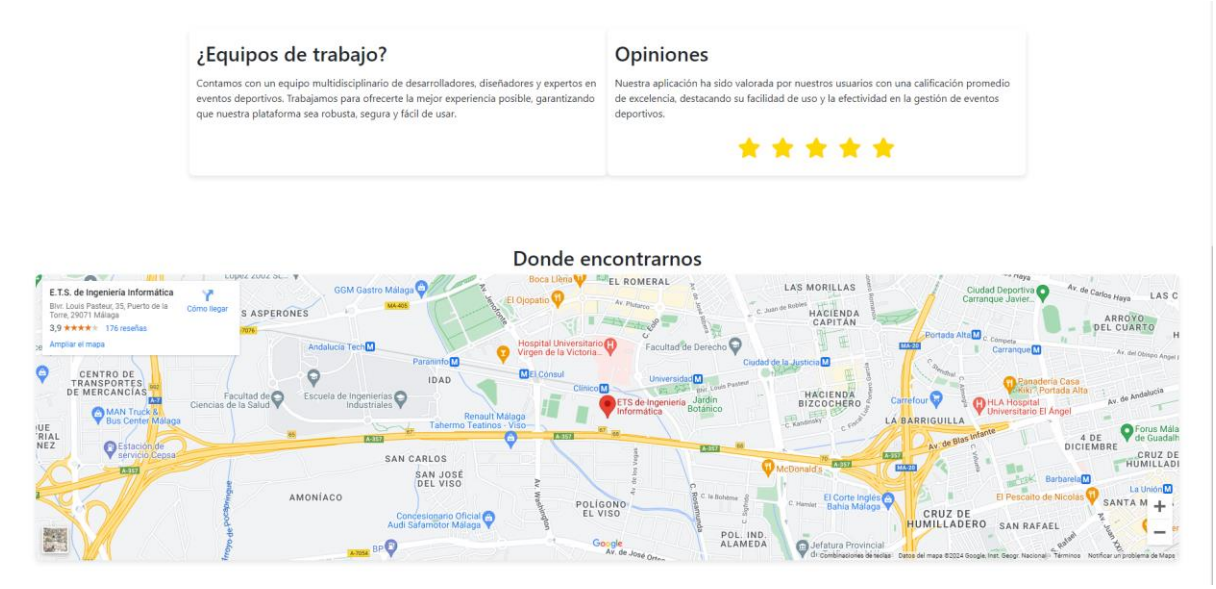

Figura 52. Página de información

Inicio de sesión

El usuario registrado podrá iniciar sesión con sus credenciales, registrarse en caso de que no lo este o incluso recuperar su contraseña si se le olvidó.

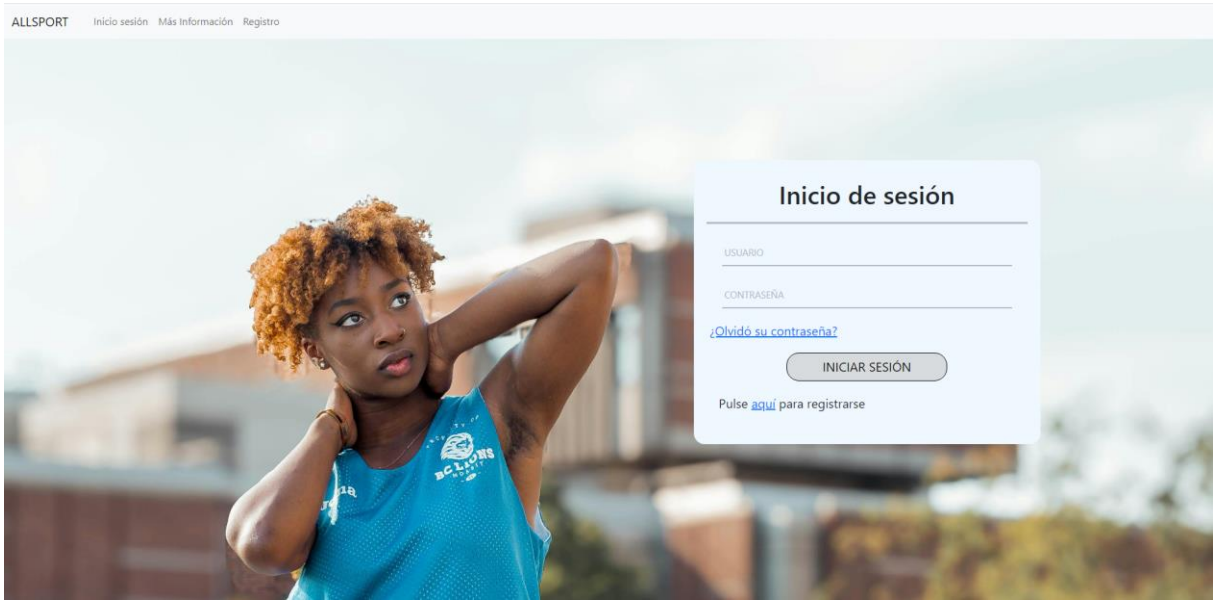

Figura 53. Página de inicio de sesión

Olvidó contraseña

Desde el inicio de sesión el usuario podrá acceder al enlace de recuperar contraseña, el cual le redirigirá a la siguiente vista donde con poner el correo con el que se hizo el registro este recibirá un enlace donde se restablecerá.

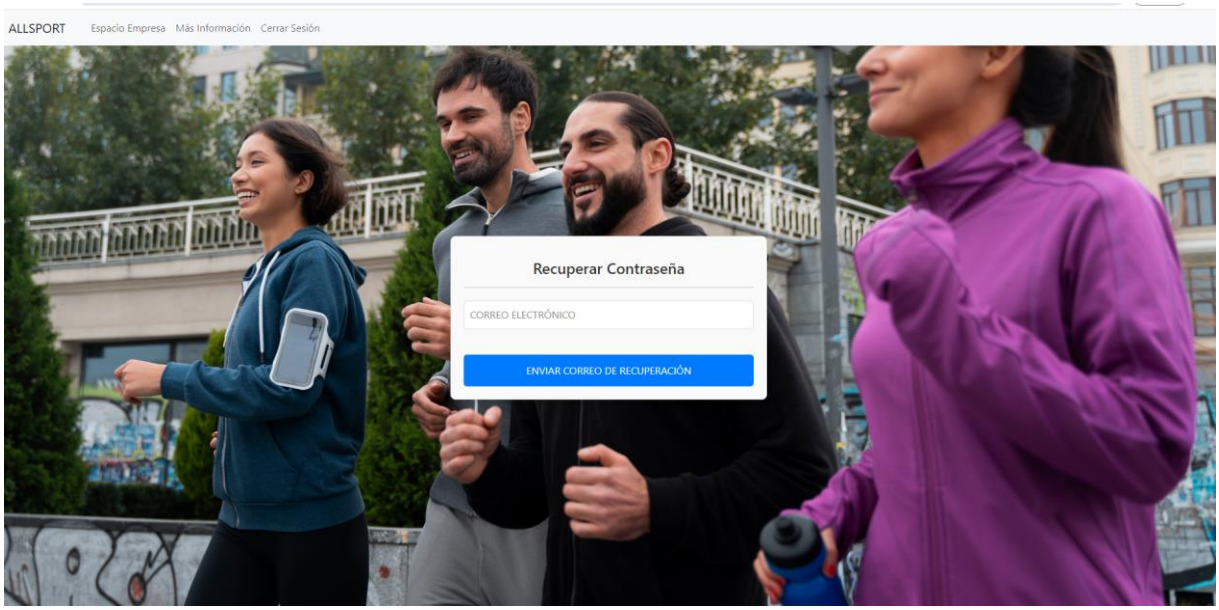

Figura 54. Página de olvidar contraseña

Espacio usuario inicial

Una vez iniciado sesión si el usuario está registrado como "persona física" accederá a la siguiente vista donde encontrará:

- **Barra de navegación:** donde se podrá ir a todas las vistas anteriores, cerrar sesión o ver la lista de eventos.
- **Evento destacado:** será el evento con mayor repercusión del mes.
- **Eventos por categoría:** podrá ver un listado de eventos ya filtrados.
- **Calendario:** podrá ver un calendario con vista mes a mes con los eventos marcados para mayor facilidad del usuario.

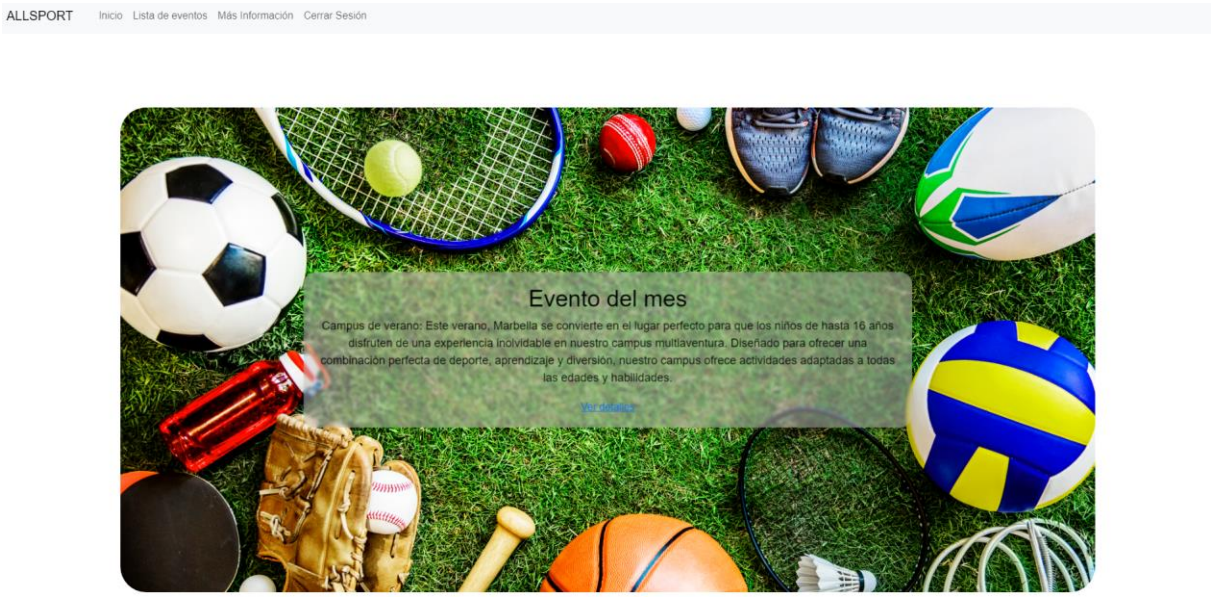

Figura 55. Página de usuario, vista de evento destacado

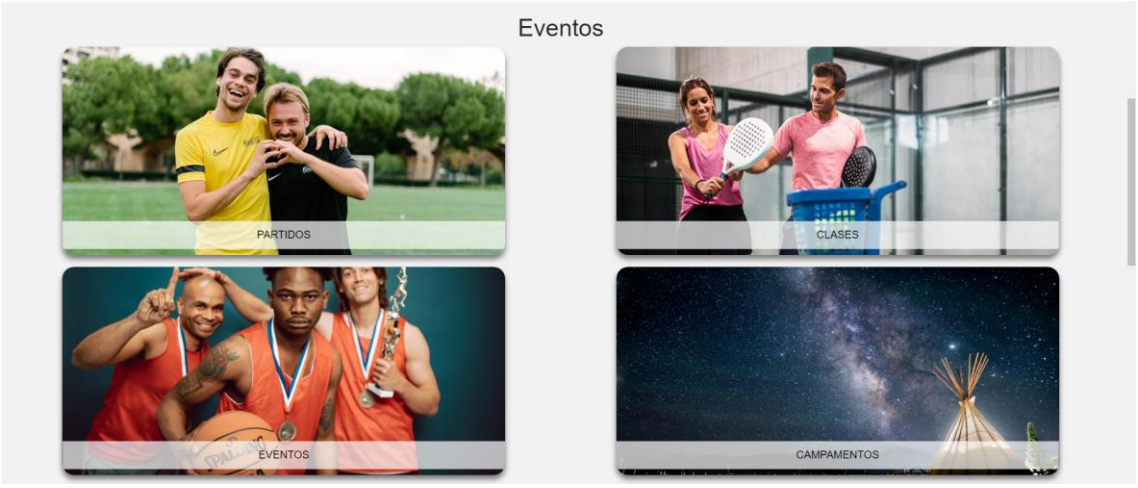

Figura 56. Página de usuario, vista de eventos por categoría

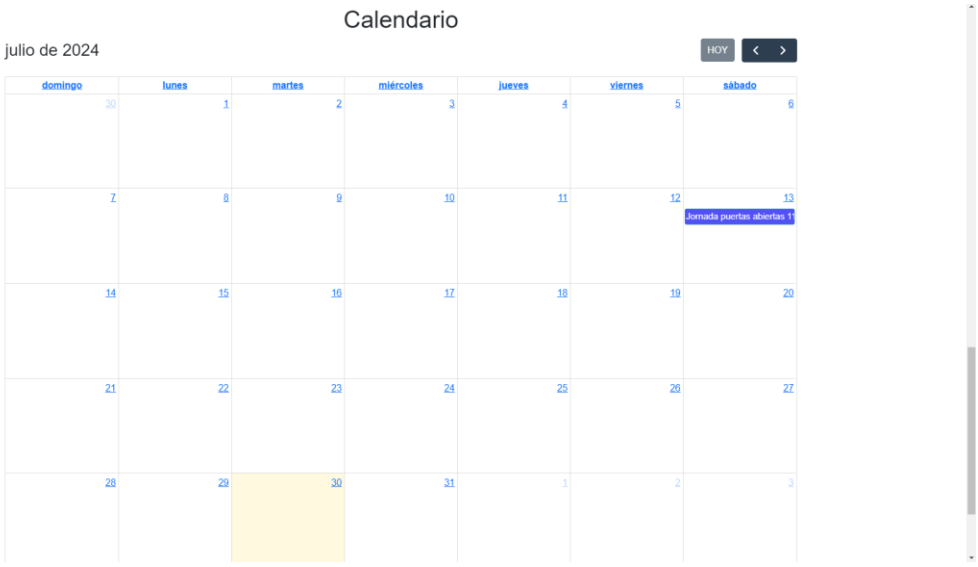

Figura 57. Página de usuario, vista de calendario

Eventos

Si el usuario en la barra de navegación accede a eventos, le redirigirá a esta vista donde aparecerán todos los eventos. Estos pueden ser filtrados por cualquiera de los campos en orden ascendente o descendente y en ver toda la información pulsando el botón de acción.

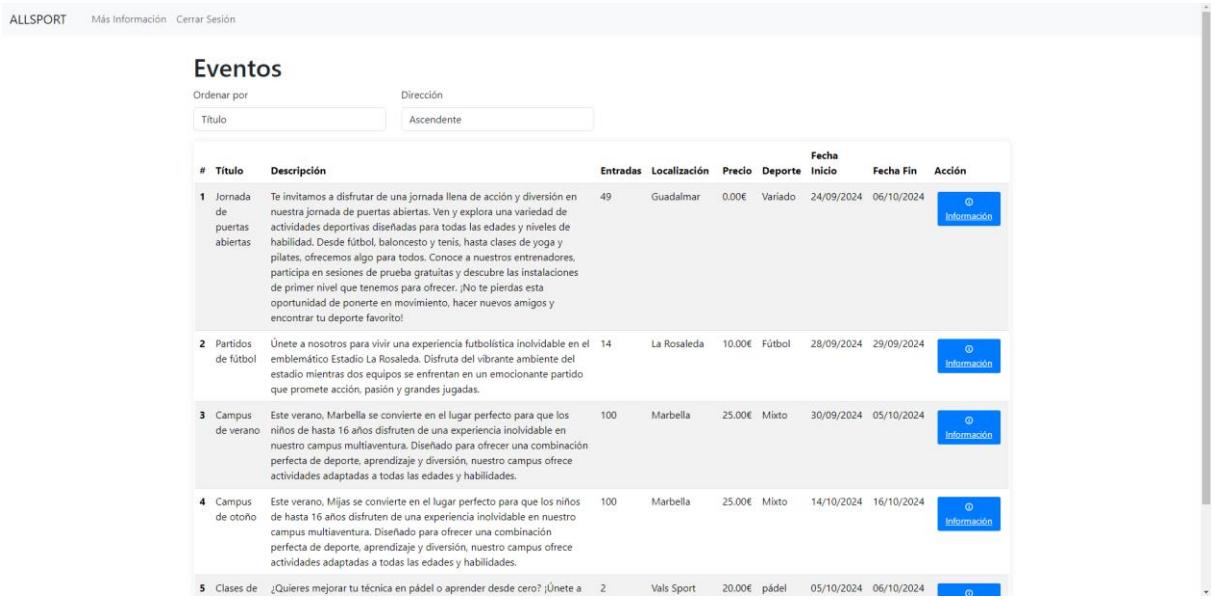

Figura 58. Página de eventos

Información de evento

En esta vista el usuario puede ver toda la información del evento además de un carrusel de las imágenes y un botón de compra.

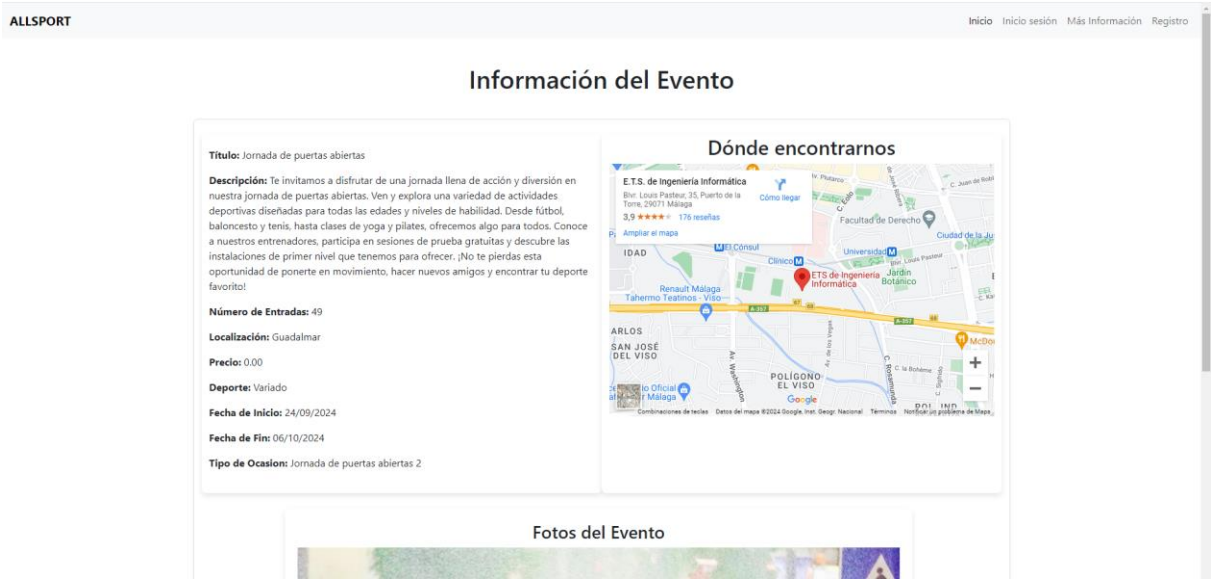

Figura 59. Página de información de eventos

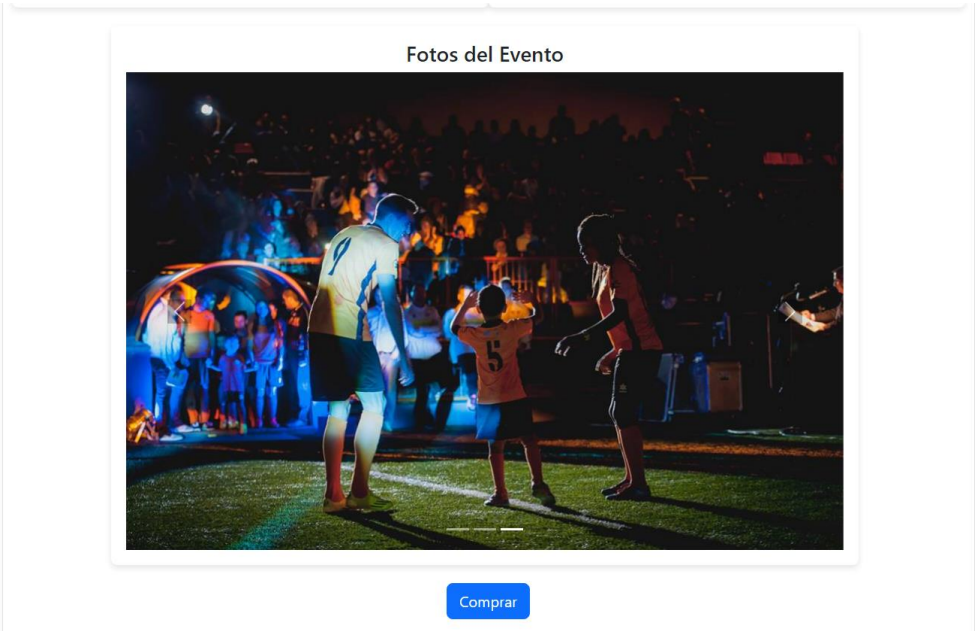

Figura 60. Página de información de eventos

Compra entrada

Si pulsa el botón de comprar aparecerá una ventana emergente con el mensaje correspondiente de que se ha realizado la compra si hay entradas o de error en el caso de que no estén disponibles.

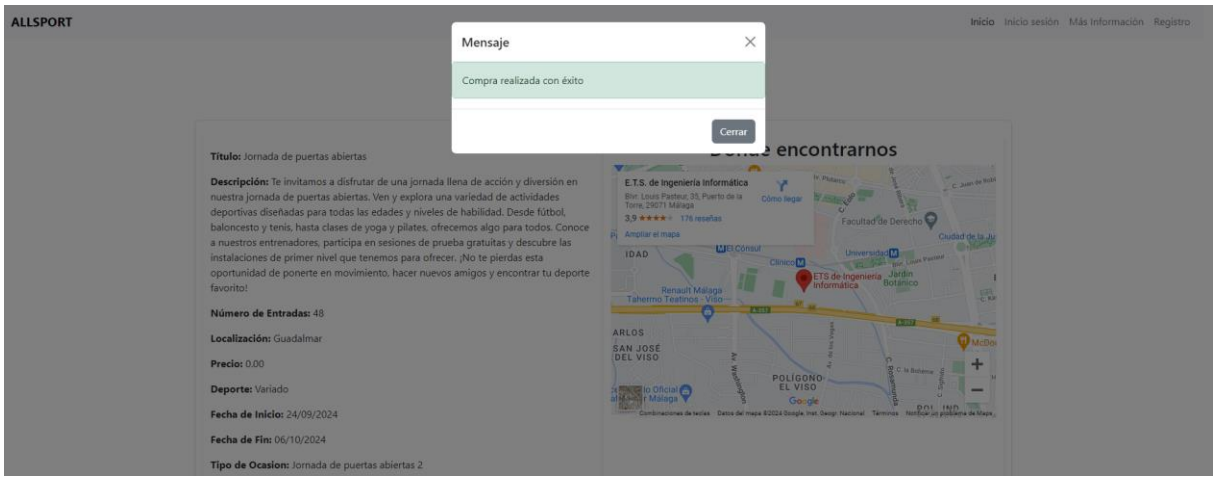

Figura 61. Ventana emergente de compra de entradas

Recibo de entrada

Una vez se ha realizado la compra el usuario recibe en su correo la entrada correspondiente.

Figura 62. Correo de recibo de entrada

Espacio empresa

Si el usuario registrado era una empresa tendrá un cuadro de administración donde se podrán crear eventos o modificar los suyos.

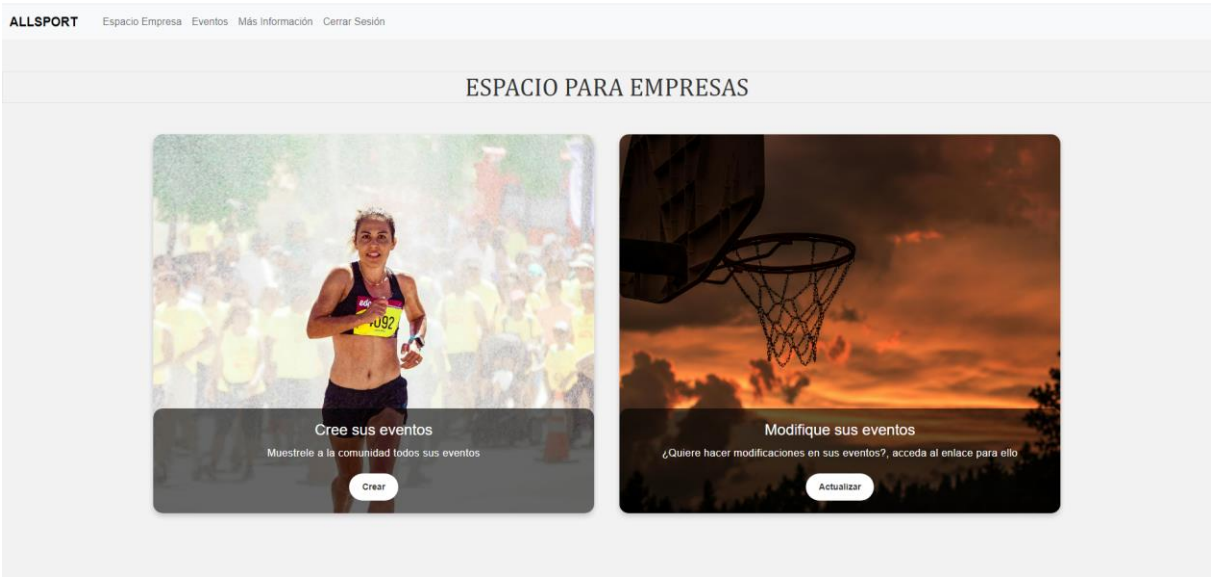

Figura 63. Página de administración para las empresas

Crear eventos

Si el usuario quiere crear eventos deberá completar todos los campos requeridos y añadir las imágenes correspondientes.

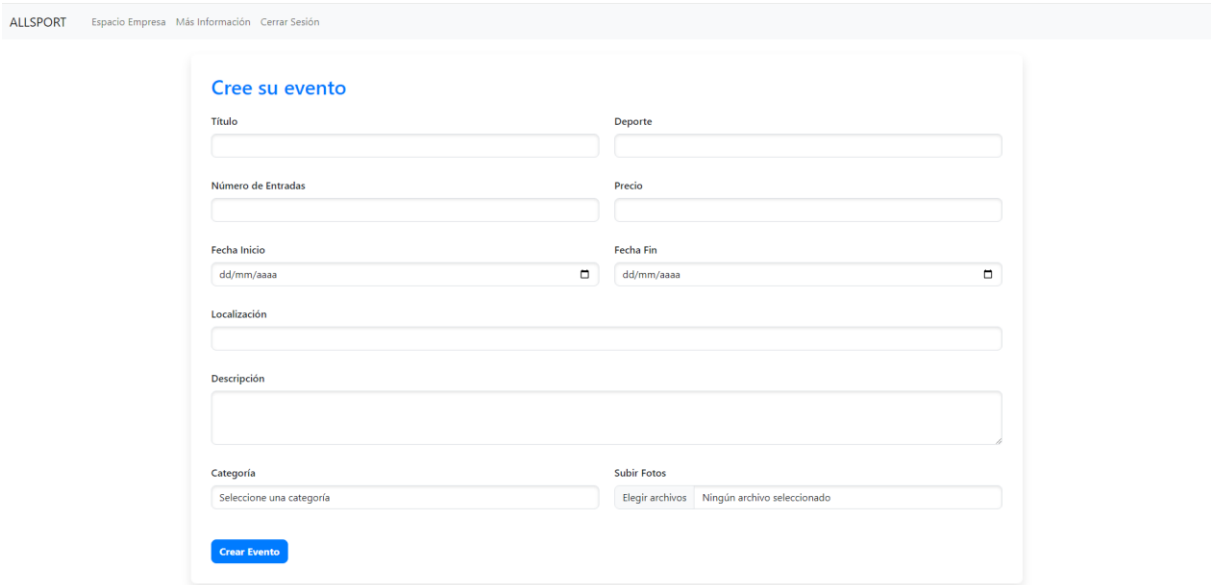

Figura 64. Página de crear evento

Ver eventos

Esta vista de eventos tiene como diferencia de la de un usuario registrado como persona que aquí la empresa puede eliminar, modificar o acceder a la información de sus eventos.

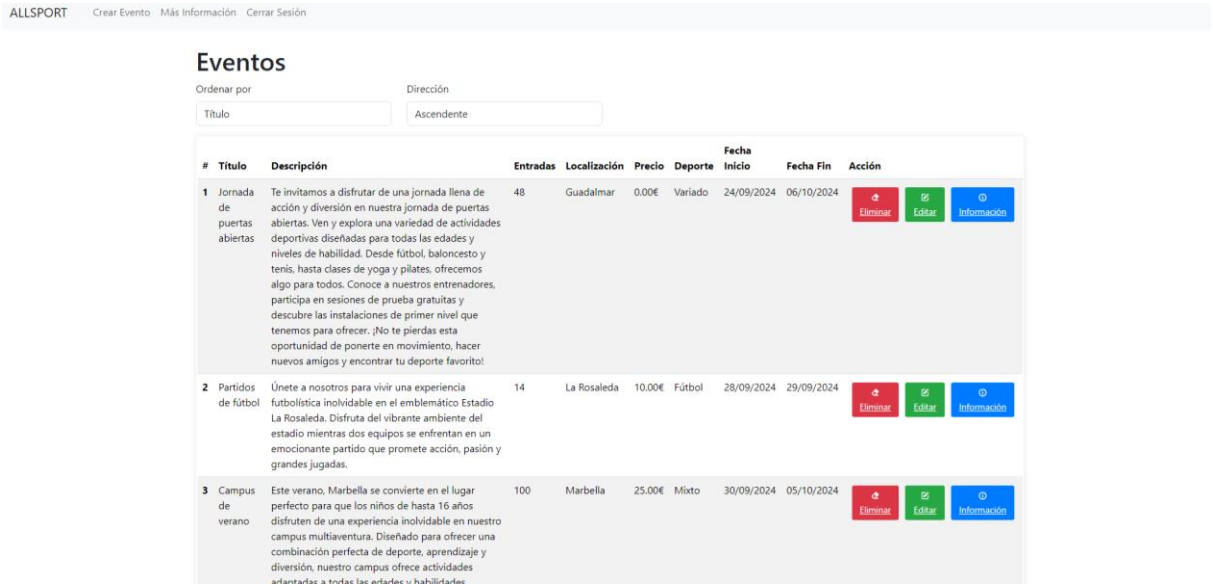

Figura 65. Página de ver eventos de las empresas

Editar evento

La vista de editar tendrá toda la información del evento con todos sus campos rellenos tal y como se guardaron en el momento de la creación. Cualquier modificación que se desee hacer en los campos de texto, añadir imágenes o eliminarlas para hacerlas efectivas se deberá pulsar el botón de guardar cambios.

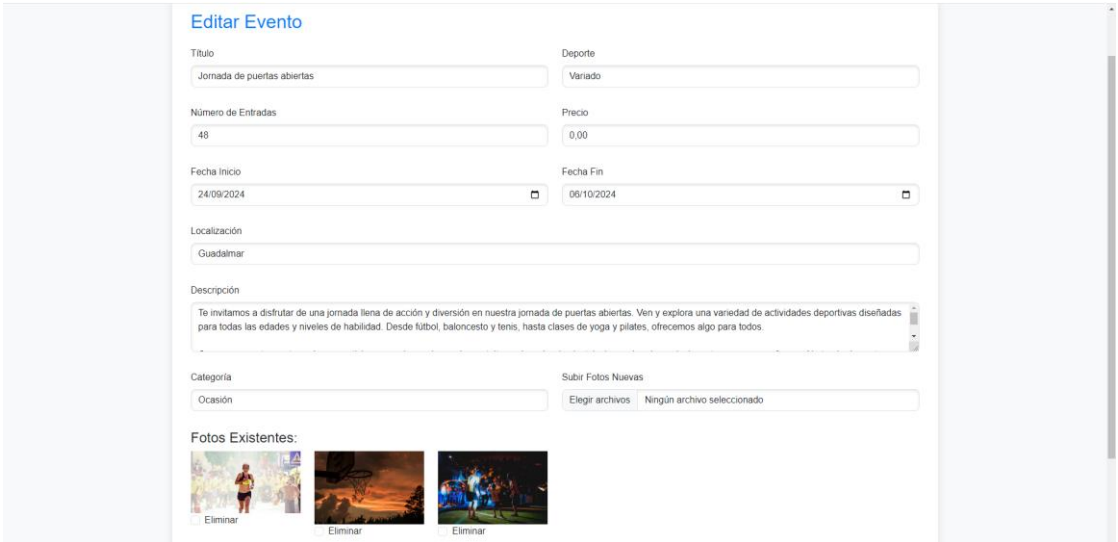

Figura 62. Página de edición de eventos

Espacio administrador

La vista del administrador la compondrá:

- **Vista del usuario**: se podrá ver la aplicación como si fuese un usuario registrado como persona.
- **Vista de empresa**: :se podrá ver la aplicación como si fuese un usuario registrado como empresa.
- **Modificación y creación de eventos**: a diferencia de las empresas que solo acceden a los suyos esta vista si estará compuesta por todos los eventos que se encuentran almacenados.
- **Lista de usuarios:** podrá ver y dar de baja a los usuarios que se encuentran en el sistema.

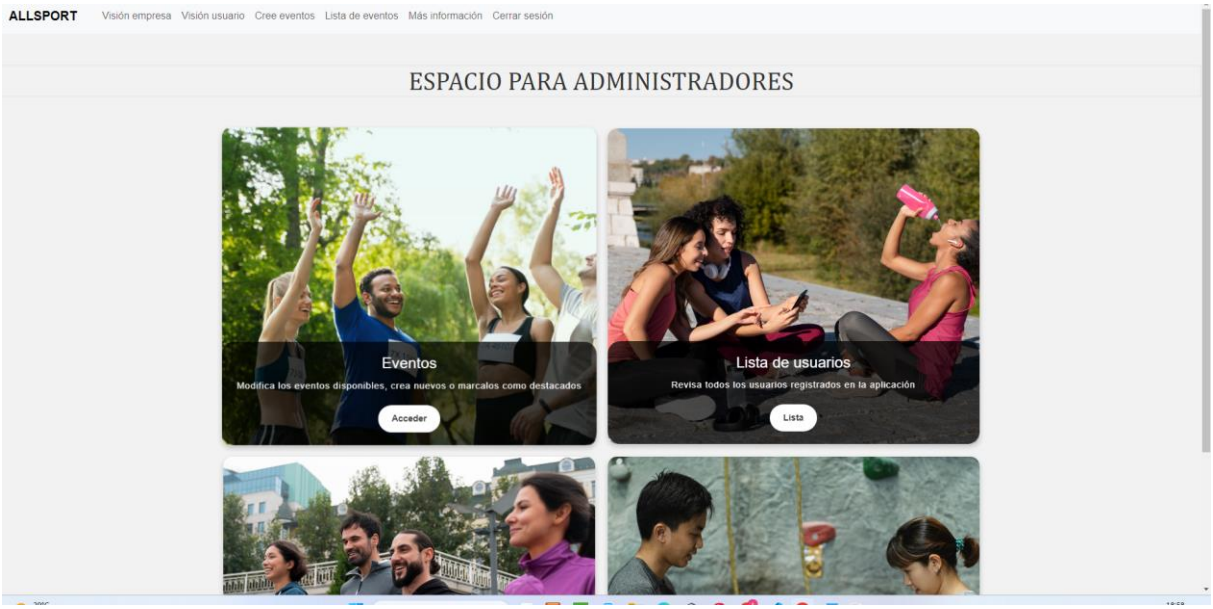

Figura 63. Página de administrador

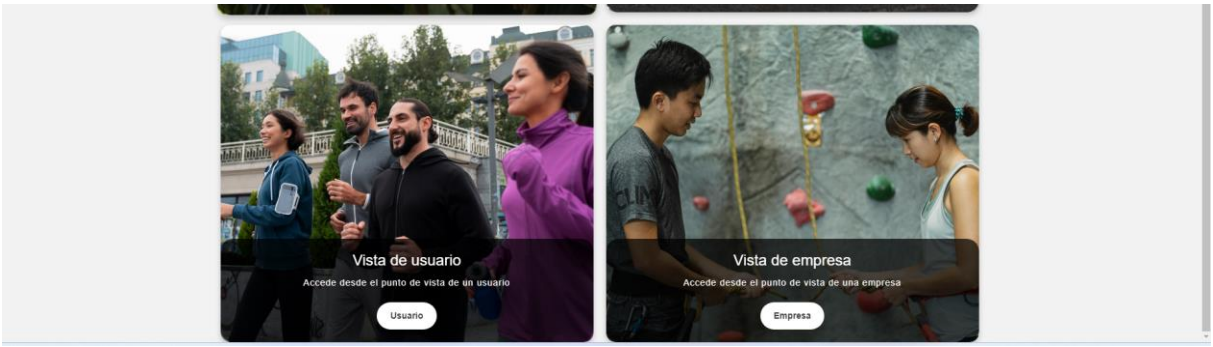

Figura 64. Página de administrador

Eventos desde administrador

En la lista de eventos a diferencia del usuario y la empresa tendrá un botón para marcar como destacado un evento

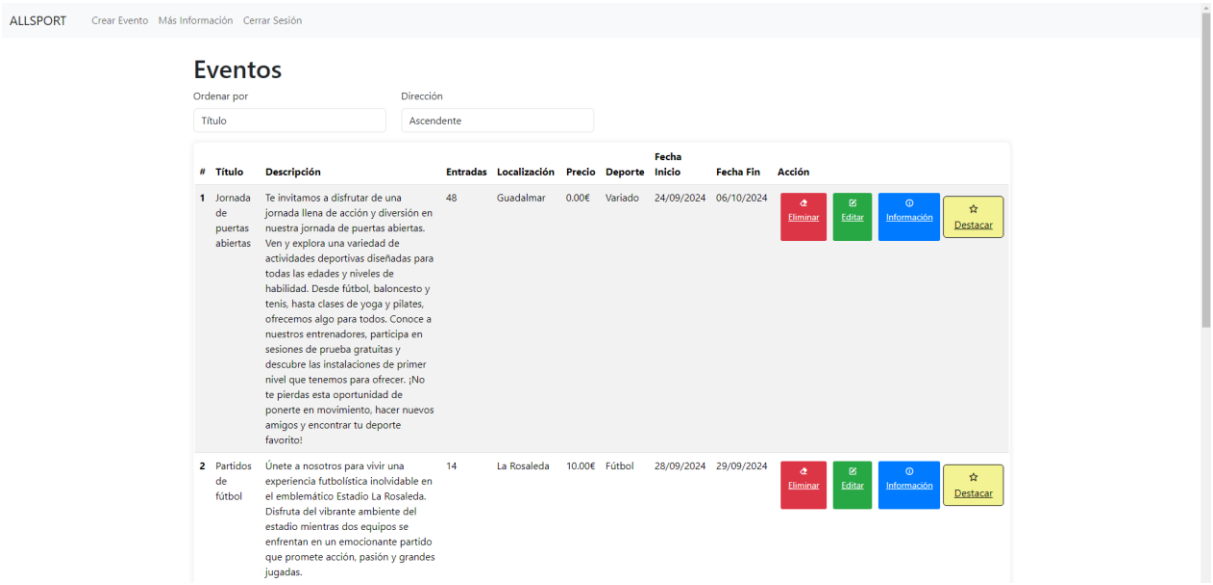

Figura 65. Página de administrador

Lista usuarios

Si accede a la lista de usuarios verá una lista con todos y la opción de dar de baja.

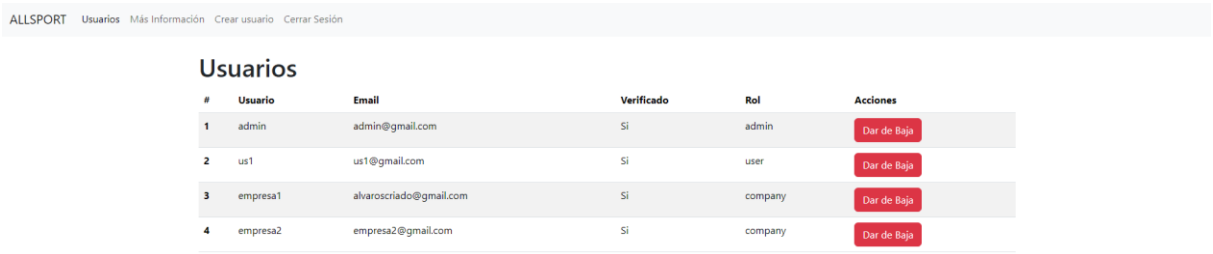

Figura 66. Página de administrador

UNIVERSIDAD DE MÁLAGA | uma.es

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga

E.T.S. DE INGENIERÍA INFORMÁTICA