UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Crea tu liga. Diseño e implementación de un gestor deportivo

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Vela de la Iglesia, Amadeo Pablo

Tutor/a: Sáez Barona, Sergio

CURSO ACADÉMICO: 2022/2023

Resumen

Desarrollo de una aplicación web para la gestión de ligas deportivas. Su principal objetivo es proporcionar una plataforma simple donde los usuarios pueda unificar las ligas en las que están suscritos en los diferentes ámbitos de su vida con un único inicio de sesión. De modo que puedan visualizar dentro de la misma plataforma todos los deportes en los que participan y gestionar sus equipos, ligas, jugadores, calendarios, etc.

Palabras clave: aplicación web, MySQL, PHP, gestión deportiva, ligas, equipos, deportes

Resumen

Desenvolupament d'una aplicació web per a la gestió de lligues esportives. El seu principal objectiu és proporcionar una plataforma simple on els usuaris puguen unificar les lligues en les quals estan subscrits en els diferents àmbits de la seua vida amb una única sessió d'inici. De manera que puguen visualitzar dins de la mateixa plataforma tots els esports en els quals participen i gestionar els seus equips, lligues, jugadors, calendaris, etc.

Paraules clau: aplicació web, MySQL, PHP, gestió esportiva, lligues, equips, esports.

Summary

Development of a web application for the management of sports leagues. Its main objective is to provide a simple platform where users can unify the leagues they are subscribed to in different areas of their life with a single login. So that they can view within the same platform all the sports in which they participate and manage their teams, leagues, players, calendars, etc.

Keywords: web application, MySQL, PHP, sports management, leagues, teams, sports.

Tabla de contenidos

Tabla de figuras

1. Introducción

En el entorno del deporte, la gestión eficiente de las ligas deportivas es esencial para brindar a los equipos, jugadores y administradores una experiencia deportiva fluida y gratificante. En este contexto, se ha desarrollado una innovadora plataforma web de gestión de ligas que tiene como objetivo unificar todas las competiciones deportivas en una única aplicación en línea. Esta plataforma aborda la necesidad de simplificar y mejorar la gestión de ligas, ofreciendo una solución integral para administradores, equipos y jugadores.

En esta memoria del proyecto, exploraremos la motivación detrás de esta plataforma, los objetivos que busca alcanzar, el impacto esperado en la comunidad deportiva, la metodología de desarrollo y la estructura fundamental de la plataforma. A través de esta memoria, se pretende proporcionar una visión completa de esta solución tecnológica que espera facilitar la gestión de ligas deportivas en un entorno copado por competiciones de cualquier tipo de deporte en un momento donde el culto al deporte y a la vida saludable está presente en todos nosotros.

1.1. MOTIVACIÓN

La motivación detrás de la creación de esta plataforma de gestión de ligas es abordar la necesidad de simplificar y mejorar la experiencia de todos los involucrados en el mundo del deporte, desde administradores de ligas hasta equipos y jugadores. Tradicionalmente, la gestión de ligas deportivas involucra múltiples herramientas y procesos dispersos, lo que puede resultar en una falta de eficiencia y colaboración. La motivación de este proyecto radica en proporcionar una solución integral que aúne todas las ligas deportivas en una misma plataforma web, simplificando así la administración, programación de partidos, seguimiento de resultados y estadísticas deportivas. Se desea fomentar una mayor participación en el deporte al ofrecer una experiencia de usuario sencilla, más eficiente y gratificante.

1.2. OBJETIVOS

Los objetivos principales de esta plataforma son:

- Unificar todas las ligas deportivas en una única aplicación web para facilitar su gestión.
- Ofrecer a los administradores de ligas una herramienta eficiente para programar partidos y hacer un seguimiento de resultados y estadísticas.

Mejorar la experiencia de los equipos y jugadores al proporcionar acceso fácil a la información de sus competiciones.

- Fomentar la colaboración y la participación en ligas deportivas.
- Reducir la complejidad y los obstáculos en la administración de competiciones deportivas.

1.3. IMPACTO ESPERADO

Esperamos que esta plataforma tenga un impacto significativo en varios aspectos:

- Mayor participación en ligas deportivas debido a la facilidad de acceso y gestión
- Eficiencia mejorada en la administración de ligas y programación de partidos
- Colaboración mejorada entre equipos y jugadores
- Mayor seguimiento de estadísticas y resultados deportivos
- Reducción de la complejidad en la gestión de competiciones deportivas
- Mayor adopción digital en entornos con pocos recursos.

1.4. METODOLOGÍA

La metodología utilizada para desarrollar esta plataforma implica la investigación de las necesidades de los usuarios, el diseño centrado en el usuario y la implementación de tecnología web moderna. Se han llevado a cabo encuestas y entrevistas con administradores de ligas y usuarios finales para comprender sus requisitos y desafíos. El desarrollo se basa en las mejores prácticas de diseño de experiencia de usuario (UX/UI) y en la implementación de tecnologías web escalables y seguras.

1.5. COLABORACIONES

Desde https://gestorligas.com/ han colaborado a la resolución de dudas de manera altruista para la organización de tablas en la base de datos, y cómo agilizar estructuras para algunas consultas referentes a los partidos y jornadas.

2. Estado del Arte

En la actualidad, la gestión de ligas y eventos deportivos ha experimentado una transformación significativa gracias al desarrollo de aplicaciones web especializadas. Estas herramientas están diseñadas para simplificar y mejorar las tareas administrativas y operativas de los gestores deportivos, brindando una serie de ventajas y características innovadoras. A continuación, se presenta un resumen del estado del arte en aplicaciones web para la gestión deportiva.

2.2. PLATAFORMAS

Todo en Uno:

SportsPress[1]: Esta aplicación se integra perfectamente con sitios web de WordPress y permite a los gestores de ligas crear, administrar y mostrar información sobre ligas y equipos en línea. Proporciona estadísticas en tiempo real y herramientas de programación.

TeamSnap[2]: Ofrece una variedad de funciones que incluyen la programación de eventos, la comunicación con jugadores y padres, y la administración de listas y estadísticas de jugadores. Es ampliamente utilizado en deportes de equipo como el fútbol y el baloncesto.

Programación y Gestión de Eventos:

LeagueApps[3]: Esta plataforma se centra en simplificar la programación de eventos deportivos, la inscripción de jugadores y la gestión de pagos. Ofrece una interfaz intuitiva y personalizable para los usuarios.

Teamer[4]: Es una aplicación de gestión de eventos deportivos ampliamente utilizada para la recaudación de fondos, la programación de eventos y la comunicación con los miembros del equipo.

Plataformas de Comunicación y Redes Sociales:

Teamer[5]: Además de su funcionalidad de programación, Teamer también actúa como una plataforma de comunicación para equipos deportivos y ofrece herramientas para recaudar fondos y gestionar los recursos financieros del equipo.

[1] https://es.wordpress.org/plugins/sportspress/
[2] https://www.teamsnap.com/
[3] https://leagueapps.com/
[4] https://www.teamer.net

SportsYou[6]: Esta aplicación permite a entrenadores, jugadores y padres mantenerse en contacto y compartir información importante sobre los eventos y entrenamientos deportivos.

Plataformas de Estadísticas y Análisis:

Hudl [7] : Es una herramienta de análisis de video deportivo que permite a los entrenadores y jugadores revisar grabaciones de juegos para mejorar el rendimiento y la estrategia.

iSportsAnalysis[8]: Ofrece herramientas avanzadas de análisis de video para deportes como el fútbol, el rugby y el hockey. Ayuda a los equipos a comprender mejor su rendimiento y a tomar decisiones informadas. Centrada sobre todo en el público norte-americano.

Las aplicaciones web para la gestión deportiva se caracteriza por la diversidad de soluciones disponibles. Estas aplicaciones están diseñadas para abordar diversas necesidades, desde la programación de eventos hasta la comunicación y el análisis de estadísticas. La tecnología continúa desempeñando un papel crucial en la simplificación y mejora de la gestión deportiva, brindando eficiencia y comodidad a todos los involucrados en el mundo del deporte.

Existen muchos ejemplos más llevados a cabo por empresas con intenciones recaudatorias. Sin embargo, también aparecen multitud de aplicaciones web realizadas por usuarios sin ánimo de lucro y que su fin, al igual que puede darse en el presente caso, sirven con fines académicos centrados en el aprendizaje y desarrollo como programador de su/sus creadores. En los que, en cierto momento, la plataforma ha podido tomar relevancia e incluso llegar a su comercialización.

Entre estos ejemplos podemos encontrar webs como

- https://gestorligas.com/
- https://ligaprivada.es/
- https://www.xporty.com/gestor-de-torneos
- https://www.enjore.com/

[5] https://www.teamer.net/
[6] https://www.sportsyou.com/
[7] https://www.hudl.com/
[8] https://www.isportsanalysis.com/

2.3. PROPUESTA

Esta aplicación de gestión de ligas intenta tener un diferencial fundamental: aúna todas las ligas en una misma plataforma bajo el mismo usuario. Esta característica única ofrece una serie de ventajas y beneficios significativos para los usuarios. Aquí se presentan diferentes puntos como propuesta de valor distintiva:

1. Centralización y simplificación: La principal ventaja es su capacidad para centralizar todas las ligas deportivas en un solo lugar. Los usuarios, ya sean gestores, entrenadores, jugadores o aficionados, pueden acceder a múltiples ligas desde una única plataforma. Esto elimina la necesidad de utilizar varias aplicaciones o sitios web para seguir diferentes competiciones.
2. Acceso a una amplia variedad de ligas: la aplicación brinda la posibilidad de adaptar cualquier tipo de competición y deporte a la plataforma. Los usuarios pueden explorar y unirse a ligas locales, regionales e incluso internacionales, lo que les brinda una experiencia deportiva más enriquecedora.
3. Comunidad Deportiva Unida: la aplicación fomenta la creación de una comunidad deportiva unida. Los usuarios pueden conectarse con personas que comparten su pasión por el deporte.
4. Personalización y flexibilidad: se entiende que cada liga deportiva tiene sus propias necesidades y reglas. Por lo tanto, la aplicación permite la personalización ya que es posible, mediante ligeras implementaciones, adaptar la plataforma a las reglas específicas de cada liga y deporte, lo que garantiza una experiencia adecuada para cada comunidad deportiva.
5. Información integral y actualizada: los usuarios tienen acceso a información integral y actualizada sobre sus ligas y equipos favoritos. Esto incluye estadísticas detalladas, horarios de partidos y resultados.
6. Ahorro de tiempo y recursos: al unificar todas las ligas en una sola plataforma, la aplicación ahorra tiempo y recursos tanto a gestores como a aficionados. Ya no es necesario buscar información dispersa en diferentes sitios web o aplicaciones.
7. Facilita la organización de eventos multideporte: para aquellos que deseen organizar eventos multideporte, la aplicación brinda la capacidad de coordinar múltiples competiciones simultáneamente, lo que simplifica en gran medida la logística. Esto es debido a que en un equipo pueden inscribirse multitud de jugadores, y cada equipo puede participar con estos jugadores en diferentes ligas de distintos deportes de manera simultánea.

3. Análisis del problema

La aplicación de gestión de ligas deportivas es una solución diseñada para facilitar la organización y el seguimiento de competiciones deportivas a diferentes niveles, desde ligas locales hasta eventos de mayor envergadura. Este proyecto se centra en un análisis funcional detallado de la aplicación, desde un punto de vista operativo, con el objetivo de definir cómo la plataforma abordará los requisitos funcionales planteados.

En la primera parte de este análisis, se abordaron los módulos principales de la aplicación, destacando las funcionalidades clave que se esperan ofrecer a los usuarios. Esto incluye desde la creación y gestión de ligas hasta el registro de resultados de partidos y la generación de clasificaciones. Además, se exploraron aspectos esenciales como la comunicación entre usuarios y la seguridad de los datos.

En esta segunda parte, se profundizará aún más en el análisis funcional detallado. Se describirán los casos de uso específicos que los usuarios experimentarán al interactuar con la aplicación, así como las entidades o clases propuestas que respaldarán estas funcionalidades. Además, se considerarán aspectos críticos como la seguridad, la eficiencia algorítmica y el marco legal y ético.

Este análisis funcional proporciona una base sólida para el diseño y la implementación de la aplicación de gestión de ligas deportivas, asegurando que cada requisito funcional sea abordado de manera efectiva y que los usuarios puedan aprovechar al máximo esta plataforma integral para la gestión de competiciones deportivas.

3.1. ANÁLISIS FUNDAMENTAL DETALLADO

El análisis funcional detallado proporciona una visión más completa de cómo funcionará la aplicación de gestión de ligas deportivas y qué funcionalidades se deben implementar para satisfacer las necesidades de los usuarios.

3.1.1. Módulo de Usuarios

Registro de Usuarios: Los usuarios pueden registrarse en la plataforma proporcionando su dirección de correo electrónico y contraseña. Esto crea una cuenta de usuario en la base de datos. Más tarde pueden agregar nombre y apellidos en la edición del perfil.

Inicio de Sesión: Los usuarios registrados pueden iniciar sesión en la aplicación utilizando su dirección de correo electrónico y contraseña.

Gestión de Perfil: Los usuarios pueden ver y editar su perfil, incluyendo detalles como su nombre y apellidos, foto de perfil y correo.

Inscripción en equipos: Los usuarios pueden buscar y solicitar acceso a formar parte de tantos equipos como deseen.

Gestión de Usuarios: Los administradores de liga pueden gestionar los usuarios, incluyendo la asignación de roles y permisos. Así como editar la información de cualquier usuario.

3.1.2. Módulo de Equipos

Creación de equipos: Cualquier usuario registrado puede crear nuevos equipos. Esto implica proporcionar información sobre el equipo, como el nombre y escudo. También permite configurar si admite o no nuevas solicitudes de usuarios.

Aceptar inscripciones de jugadores: Los administradores pueden ver dentro del equipo las invitaciones pendientes de usuarios que quieren formar parte del equipo como jugadores. El administrador puede aprobar o denegar el acceso al equipo.

Inscripción en ligas: Los administradores pueden buscar y solicitar acceso a tantas ligas como deseen que su equipo tome parte.

3.1.3. Módulo de Ligas

Creación de Ligas: Los usuarios registrados y con rol de gestor pueden crear nuevas ligas deportivas. Esto implica proporcionar información sobre la liga, como el nombre, el deporte que se juega y las fechas de inicio y fin.

Aceptar inscripciones de equipos: Los administradores pueden ver dentro de la liga las invitaciones pendientes de equipos que han decidido inscribirse a la liga. El administrador puede aprobar o denegar el acceso a la liga.

Iniciar la competición: Los administradores pueden escoger cuándo dar comienzo a la competición, generando así la clasificación, el modelo de puntuación (basado en el deporte seleccionado) y las jornadas de toda la liga.

Modificar fechas de jornadas y partidos: Los administradores pueden (y deben) cambiar la fecha de las jornadas generadas. Al ser un constructor genérico, la liga establece todas las jornadas con fecha igual al inicio de la liga. Esto es para que cada administrador adapte las fechas acordes a la situación particular de cada competición.

Insertar resultados: Los administradores de partido pueden ingresar los resultados de los partidos, incluyendo la puntuación de cada equipo.

Estadísticas de Partidos: La aplicación calculará automáticamente en función del tipo de deporte, estadísticas como goles totales, goles por partido, victorias, derrotas y empates.

Generación de Clasificaciones: La aplicación generará automáticamente tablas de clasificación para cada liga en función de los resultados de los partidos.

Actualización de Puntos: Los puntos se otorgarán a los equipos según los resultados de los partidos (por ejemplo, 3 puntos por victoria, 1 punto por empate). Estos puntos se utilizarán para clasificar a los equipos.

3.1.4. Módulo de Seguridad

Seguridad de Datos: La aplicación garantizará la seguridad de los datos de los usuarios, incluyendo la protección de contraseñas y la encriptación de datos sensibles.

Protección contra Amenazas: Se ha diseñado el código con buenas prácticas que implementa medidas de seguridad para protegerse contra amenazas como la inyección SQL y el acceso no autorizado.

3.2. ESPECIFICACIÓN DE REQUISITOS

Mediante una especificación de requisitos de software (ERS) se proporciona una descripción detallada de los requisitos funcionales y no funcionales de la aplicación. Esta aplicación tiene como objetivo principal facilitar la creación y administración de ligas deportivas en línea, brindando a los usuarios la capacidad de registrar resultados, llevar un seguimiento de las clasificaciones y promover la comunicación entre equipos y jugadores.

Objetivo
El objetivo principal de la aplicación de gestión de ligas deportivas es proporcionar una plataforma en línea que permita a los usuarios crear y administrar ligas deportivas, programar partidos y llevar un seguimiento de las clasificaciones y resultados.

Alcance
Esta aplicación cubre la gestión de múltiples ligas deportivas, siendo posible adaptar las competiciones y sus puntuaciones a cualquier deporte. Los usuarios pueden registrarse, iniciar sesión y crear sus propias ligas o unirse a ligas existentes. La aplicación permite la administración de equipos, programación de partidos, registro de resultados y seguimiento de estadísticas.

3.2.1. Requisitos Funcionales

Registro y Autenticación

- RF-01: Los usuarios pueden registrarse proporcionando su nombre, correo electrónico y contraseña.
- RF-02: Los usuarios pueden iniciar sesión utilizando su correo electrónico y contraseña.

Gestión de Ligas

- RF-03: Los usuarios con rol de gestores pueden crear una nueva liga, especificando el nombre, deporte y reglas. Este rol debe darse por parte de un administrador.
- RF-04: Los usuarios que sean creadores de algún equipo pueden unirse a ligas existentes solicitando el acceso a la liga seleccionada.
- RF-05: Los administradores de las ligas pueden personalizar la configuración de la liga; nombre, logo, deporte y las fechas de inicio y fin. El deporte seleccionado repercute en las normas de clasificación y puntuación.

Gestión de Equipos

- RF-06: Los usuarios pueden crear equipos, proporcionando el nombre del equipo, escudo, y configurando si otros usuarios pueden solicitar el acceso.
- RF-07: Los usuarios pueden solicitar unirse a los equipos.
- RF-08: Los creadores de equipos pueden aceptar invitaciones para que otros usuarios se unan al equipo.

Programación de Partidos

- RF-09: Los administradores de ligas pueden dar comienzo a la liga en cualquier momento que deseen, una clasificación y emparejamientos serán generados en formato de jornadas con la opción de ida y vuelta configurable al comienzo.
- RF-10: Los administradores de ligas pueden programar la fecha de las jornadas y partidos.

Registro de Resultados

- RF-11: Los administradores de partidos pueden registrar los resultados de los partidos, incluyendo goles marcados por cada equipo.
- RF-12: Los resultados de los partidos se reflejan automáticamente en las clasificaciones de la liga.

Clasificaciones y Estadísticas

- RF-13: Los usuarios pueden ver las clasificaciones de la liga, que se actualizan automáticamente en función de los resultados de los partidos.
- RF-14: Los usuarios pueden ver todo el calendario dividido en jornadas de toda la competición.
- RF-15: Los usuarios pueden acceder a estadísticas detalladas, como puntos, partidos, goles marcados, goles encajados y diferencias de goles.

Perfiles de Usuarios

- RF-16: Cada usuario tiene un perfil personalizado que muestra su información básica, ligas en las que participa y equipos a los que pertenece.

3.2.2. Requisitos No Funcionales

Seguridad

- RNF-01: La aplicación debe utilizar técnicas de cifrado para proteger la información de los usuarios, como contraseñas y datos personales.
- RNF-02: Se deben implementar medidas de seguridad para prevenir el acceso no autorizado a las ligas y datos de los usuarios.

Rendimiento

- RNF-03: La aplicación debe ser capaz de manejar múltiples ligas y partidos sin experimentar retrasos significativos.

Usabilidad

- RNF-04: La interfaz de usuario debe ser intuitiva y fácil de usar, incluso para usuarios no técnicos.

Escalabilidad

- RNF-05: La aplicación debe ser escalable para acomodar un crecimiento futuro en el número de usuarios y ligas.

3.3. ANÁLISIS DE LA SEGURIDAD

Dada la programación desde cero en PHP 8 y sin el uso de ningún framework de trabajo, la seguridad es una pata fundamental del proyecto. Uno de los posibles trabajos futuros, como se tratará más a delante, sería llevar a cabo un análisis exhaustivo de posibles vulnerabilidades, como inyección de SQL, ataques XSS y CSRF, y la implementación medidas de seguridad adecuadas.

Pese a que no se ha empleado ningún framework de trabajo, durante la implementación de la aplicación se han aplicado buenas prácticas de desarrollo seguro, como la protección contra la divulgación de información sensible, creando un middleware (simulando los implementados por frameworks conocidos). Esto incluye la validación de entradas, la autenticación de usuarios y la gestión segura de sesiones. También se ha hecho uso de encriptado de contraseñas y algunas prácticas para evitar la inyección de SQL.

3.4. ANÁLISIS ENERGÉTICO O DE EFICIENCIA ALGORÍTMICA

El rendimiento eficiente es esencial para garantizar una experiencia de usuario fluida. Se prestará atención a los algoritmos utilizados en la creación y administración de ligas deportivas. Esto incluye el emparejamiento de equipos según la cantidad de participantes, la programación de partidos de ida y vuelta y la gestión de eventos deportivos. A futuro, se buscarán oportunidades para optimizar algoritmos y procesos, reduciendo el tiempo de cálculo y el uso de recursos del servidor.

3.5. ANÁLISIS DEL MARCO LEGAL Y ÉTICO

El cumplimiento de las regulaciones legales y éticas es una consideración crítica. Dada la simplicidad de la aplicación, únicamente se almacena el nombre apellidos y email del usuario. Siendo esta información únicamente visible por el propio usuario o por los administradores.

3.6. ANÁLISIS DE RIESGOS

Los posibles riesgos que pueden afectar al proyecto y al producto final. Son de tipo; interrupciones del servicio, ataques cibernéticos, problemas de escalabilidad y problemas de rendimiento. Dentro del análisis de seguridad se han contemplado los ataques directos a la plataforma, siendo estos ciertamente mejorables a futuro.

Los ataques centrados en la infraestructura son ajenos al control dentro del desarrollo, y dependen totalmente del hosting contratado para el hospedaje del proyecto. Sin embargo, se ha buscado un centro especializado para llevar a cabo esta tarea. El cual establece medidas preventivas y de mitigación, como la implementación de copias de seguridad regulares, la monitorización continua de seguridad y la planificación de contingencias en caso de interrupciones graves.

3.7. CASOS DE USO

Para dar respuesta a los requisitos funcionales y no funcionales del apartado anterior, se van a exponer casos de uso que definan posibles interacciones de los usuarios con la aplicación. Proporcionando una visión más detallada de cómo debería responder el sistema.

En este contexto, exploraremos una serie de casos de uso específicos relacionados con la aplicación de gestión de ligas deportivas. Estos casos de uso describen las interacciones clave que los usuarios tendrán con la aplicación, desde el registro inicial hasta la gestión de ligas, equipos, partidos y resultados. Cada caso de uso desglosa una funcionalidad particular y proporciona una comprensión completa de cómo los usuarios y el sistema interactuarán en situaciones específicas.

CU01: Registro de un usuario

Requisito funcional relacionado: RF-01

Descripción: Este caso de uso permite a un usuario registrarse en la aplicación.

Actores: Usuario no registrado.

Flujo Principal:

1. El usuario accede a la página de registro.
2. El usuario proporciona su dirección de correo electrónico y contraseña.
3. El sistema valida la información ingresada.
4. El sistema crea una cuenta de usuario y almacena los datos en la base de datos.
5. El usuario es redirigido a su perfil para completar sus datos personales.

CU02: Inicio de sesión del usuario

Requisito funcional relacionado: RF-02

Descripción: Este caso de uso permite a un usuario iniciar sesión en la aplicación.

Actores: Usuario registrado.

Flujo Principal:

1. El usuario accede a la página de login.
2. El usuario proporciona su dirección de correo electrónico y contraseña.
3. El sistema valida la información ingresada.
4. El usuario es redirigido a su perfil para completar sus datos personales. O en el caso de que los tenga completados le redirige a la página principal.

CU03: Creación de una liga

Requisito funcional relacionado: RF-03

Descripción Este caso de uso permite a un usuario crear una nueva liga.

Actores: Usuario registrado con roles de gestión.

Flujo Principal:

1. El usuario inicia sesión en la aplicación.
2. El usuario accede a la sección de creación de ligas.
3. El usuario proporciona detalles de la liga, como nombre, deporte o logo.
4. El sistema valida la información ingresada.
5. El sistema crea la liga y almacena los datos en la base de datos, asignando al usuario como administrador de la liga.

CU04: Solicitud de acceso a una liga

Requisito funcional relacionado: RF-04

Descripción: Este caso de uso permite a un usuario administrador de un equipo solicitar acceso para unirse a una liga en la que desee que su equipo participe.

Actores: Usuario registrado y administrador de un equipo.

Flujo Principal:

1. El usuario inicia sesión en la aplicación.
2. El usuario accede a la sección de equipos, y en el apartado que indica los equipos que él administra accede a uno de ellos.
3. En el apartado de ligas en las que participa el equipo el usuario presiona el botón de inscribir.
4. El usuario accede a una lista de ligas que todavía están en fase de aceptación de equipos, que no han comenzado, busca la liga que en la que quiere que su equipo participe y solicita el acceso.

CU05: Cambios en la configuración de una liga

Requisito funcional relacionado: RF-05

Descripción: Este caso de uso permite a un usuario administrador de una liga cambiar la información de la misma.

Actores: Usuario registrado y administrador de una liga.

Flujo Principal:

1. El usuario inicia sesión en la aplicación.
2. El usuario accede a la sección de ligas, y en el apartado que indica las ligas que él administra accede a una de ellas.
3. En la pantalla de visualización de la liga, al ser administrador, se le habilita un botón de editar información.
4. El usuario accede al formulario de edición mediante el botón y realiza cambios en las fechas en las que la liga se desarrollará.
5. El sistema valida la información y actualiza los datos de ser correctos.

CU06: Creación de un equipo

Requisito funcional relacionado: RF-06

Descripción Este caso de uso permite a un usuario crear un nuevo equipo.

Actores: Usuario registrado.

Flujo Principal:

1. El usuario inicia sesión en la aplicación.
2. El usuario accede a la sección de creación de ligas.
3. El usuario proporciona detalles del equipo, como nombre, y escudo.
4. El sistema valida la información ingresada.
5. El sistema crea la liga y asigna al usuario como administrador del equipo.

CU07: Solicitud de acceso a un equipo

Requisito funcional relacionado: RF-07

Descripción: Este caso de uso permite a un usuario solicitar acceso para unirse a un equipo.

Actores: Usuario registrado.

Flujo Principal:

1. El usuario inicia sesión en la aplicación.
2. El usuario accede a la sección de equipos, y accede al que quiere comenzar a formar parte.
3. En el apartado de jugadores del equipo, al no formar todavía parte del equipo, aparece un botón para inscribirse
4. El usuario accede a una lista de ligas que todavía están en fase de aceptación de equipos, que no han comenzado, busca la liga que en la que quiere que su equipo participe y solicita el acceso.

CU08: Aceptar solicitudes de jugadores

Requisito funcional relacionado: RF-08

Descripción: En este caso, un administrador de equipos, acepta solicitudes pendientes.

Actores: Usuario registrado y administrador de algún equipo.

Flujo Principal:

1. El usuario inicia sesión en la aplicación.
2. El usuario accede a la sección de equipos, en el equipo que él administra.
3. Debajo de los jugadores del equipo, en la sección de solicitudes pendientes acepta una petición de un jugador.
4. El sistema crea la relación entre equipo y usuario en la tabla jugador.

CU09: Comenzar una liga

Requisito funcional relacionado: RF-09

Descripción: En este caso, un administrador de una liga, da comienzo a la misma.

Actores: Usuario registrado y administrador de una liga.

Flujo Principal:

1. El administrador de liga accede a la página la liga y presiona el botón de "*Comenzar liga*".
2. El administrador recibe indicaciones de las condiciones que implica comenzar una liga; emparejamientos, jornadas calendario, etc.
3. El sistema crea en base de datos una tabla de clasificación para cada equipo participante, relacionando cada equipo y cada liga con la clasificación. También crea en la tabla de partidos los encuentros que enfrentan a los equipos durante todo el desarrollo de la competición, creando también en la tabla jornadas una relación para agrupar estos partidos.
4. El administrador revisa las fechas de cada jornada y configura los enfrentamientos para que cumplan el calendario establecido por su organización.
5. El sistema queda a la espera de ir archivando resultados a medida que estos ocurran y sean introducidos por el administrador.

CU10: Cambiar fechas de una jornada

Requisito funcional relacionado: RF-10

Descripción: En este caso, un administrador de una liga, modifica las fechas de una jornada concreta.

Actores: Usuario registrado y administrador de una liga.

Flujo Principal:

1. El administrador de liga accede a la página la liga.
2. El administrador acude al apartado de jornadas y selecciona los partidos que desea modificar presionando el botón de editar.
3. El sistema realiza los cambios en la base de datos.

CU11: Registro de Resultados

Requisito funcional relacionado: RF-11 y RF-12

Descripción: Este caso de uso permite a un administrador de partido registrar los resultados de un partido.

Actores: Usuario registrado y administrador de una liga.

Flujo Principal:

1. El administrador de la liga accede a la página de la liga y al apartado de jornadas.
2. El administrador ingresa los resultados del partido, incluyendo goles marcados por cada equipo.

3. El sistema actualiza las estadísticas de los equipos y las clasificaciones de la liga y estos se muestran automáticamente en la clasificación general.

CU12: Ver las estadísticas de la liga

Requisito funcional relacionado: RF-13, RF-14 y RF-15

Descripción: Este caso de uso permite a cualquier usuario ver las estadísticas de la liga.

Actores: Usuario cualquiera, sin necesidad de registro.

Flujo Principal:

1. El usuario accede al apartado de ligas. Y selecciona la liga de la que desea conocer la información.
2. El usuario desplazándose a lo largo de la información de la liga comprueba que los resultados están al día, las jornadas pendientes y cuándo se han programado los partidos y las estadísticas de todos los equipos participantes.

CU13: Ver la información del usuario

Requisito funcional relacionado: RF-16

Descripción: Este caso de uso permite a cualquier usuario ver la información pública de cada usuario.

Actores: Usuario cualquiera, sin necesidad de registro.

Flujo Principal:

1. El usuario accede al perfil del jugador del que desea conocer la información.
2. El usuario desplazándose a lo largo de la información del usuario comprueba a qué equipos pertenece y en qué ligas están inscritos dichos equipos.

3.8. ENTIDADES CLAVE

Usuario: Representa a un usuario registrado en la aplicación. Contiene información como nombre, correo electrónico y contraseña.

Deporte: Representa un deporte. Su único atributo es el nombre y un estado para que se pueda o no seleccionar a la hora de crear una liga. Se utiliza para filtrar las ligas para facilitar su búsqueda. Otorga la posibilidad de personalizar los sistemas de puntuación para cada deporte en particular.

Equipo: Representa un equipo que participa en una liga. Contiene detalles del equipo y una lista de jugadores y un usuario administrador.

Liga: Representa una liga deportiva. Tiene atributos como nombre, deporte, reglas y una lista de equipos participantes. También tiene un atributo para identificar al administrador de la liga.

Jornada: Representa un conjunto de partidos que se deben de jugar en un plazo de tiempo determinado. Se utiliza para el emparejamiento de equipos a lo largo de una liga.

Partido: Representa un partido programado entre dos equipos. Contiene detalles como la fecha y hora, resultados y estadísticas.

Solicitud: Representa la petición de acceso de un usuario a un equipo o de un equipo a una liga. Tiene atributos que identifican al usuario que quiere acceder a un equipo, al administrador del equipo para aceptar las solicitudes de usuarios, y a su vez solicitar acceso a ligas, y por último a los administradores de las ligas para aceptar a los equipos.

Relaciones Entre Entidades:

- Un usuario puede ser administrador de una o varias ligas.
- Un usuario puede ser administrador de uno o varios equipos.
- Un usuario puede ser jugador de uno o varios equipos.
- Un jugador esto asociado a un usuario y a un equipo
- Un deporte puede estar asociado a una o varias ligas.
- Una liga tiene varios equipos participantes.
- Una liga sólo puede tener un administrador.
- Un equipo puede jugar varios partidos en diferentes ligas.
- Un equipo sólo puede tener un administrador.
- Una jornada está asociada a una liga.
- Una jornada tiene varios partidos.
- Un partido está asociado a dos equipos.
- Una solicitud está asociada a un usuario y un equipo o a un equipo y una liga.
- Una clasificación está asociada a los resultados de un equipo en una liga.

Estos casos de uso y entidades proporcionan una base sólida para el desarrollo de la aplicación de gestión de ligas deportivas. Los usuarios podrán registrarse, crear ligas, programar partidos y registrar resultados, mientras que las entidades clave ayudarán a mantener la coherencia de los datos en la aplicación.

4. Diseño y arquitectura

El proceso de diseño implica la creación de una interfaz de usuario intuitiva y atractiva, mientras que la arquitectura se encarga de definir cómo los diferentes componentes de la aplicación interactúan entre sí. En esta sección, exploraremos la planificación y toma de decisiones detrás del diseño y la arquitectura de nuestra aplicación de gestión de ligas deportivas, asegurando que cumpla con los requisitos funcionales y no funcionales, así como con las mejores prácticas de desarrollo.

A continuación, se muestran unos bocetos del prototipo para la interfaz gráfica.

Ejemplo de página principal para los equipos, en las que se puede observar la posición del logo, el menú de navegación, la posición del login del usuario y el contenido.

Aquí se mostrará todo el listado de equipos y un buscador, además de otorgar la posibilidad al usuario de dar de alta un nuevo equipo.

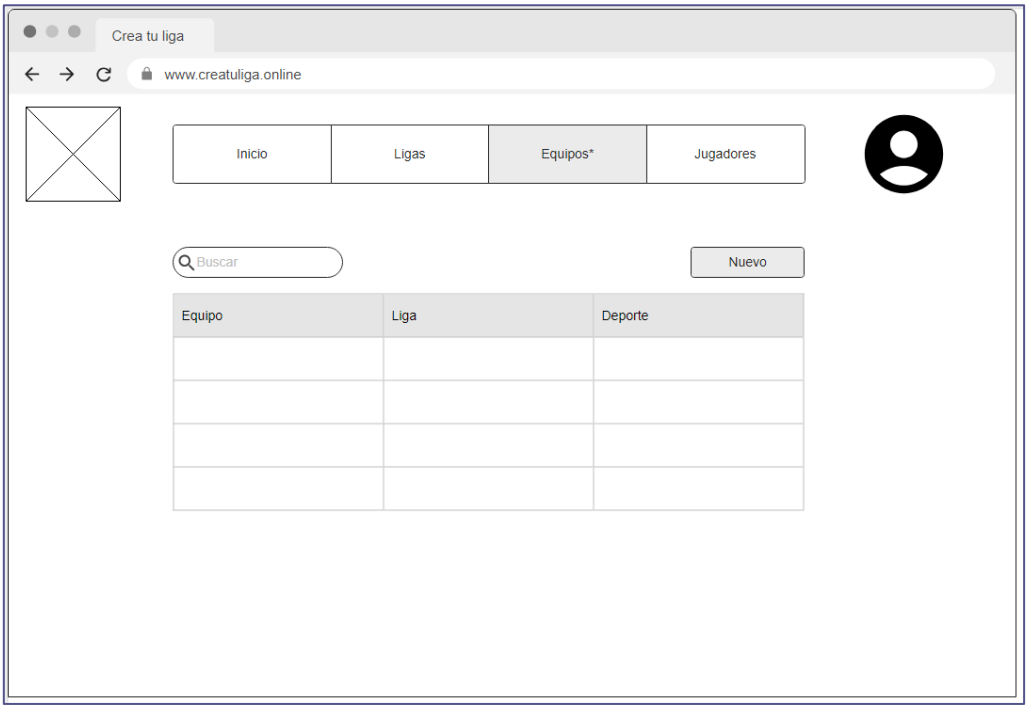


Figura 4-1 Boceto interfaz apartado equipos

Página de consulta de una liga concreta, donde veremos los equipos participantes, calendario, partidos, jornadas, etc.

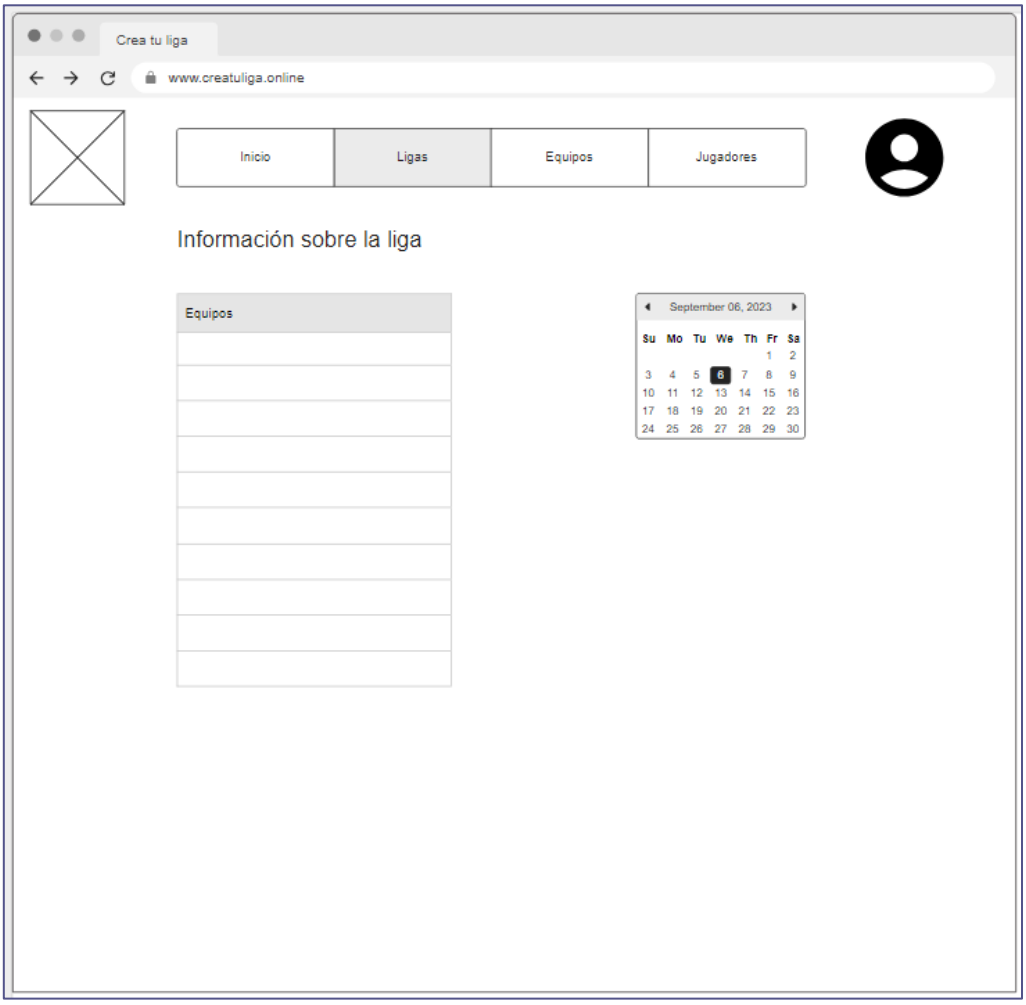


Figura 4-2 Boceto interfaz apartado ligas

Página principal del perfil del usuario conectado. En ella se observan los equipos a los que pertenece y los deportes en los que participan dichos equipos, así como información relevante del usuario como su nombre, correo, calendario, etc.

A continuación, se muestran diagramas que ilustran las estructuras de clases, relaciones y base de datos que se han diseñado para el proyecto.

4.1. DIAGRAMA DE CLASES

En este diseño de datos se observa el modelo organizativo de las entidades clave y cómo interaccionan entre ellas.

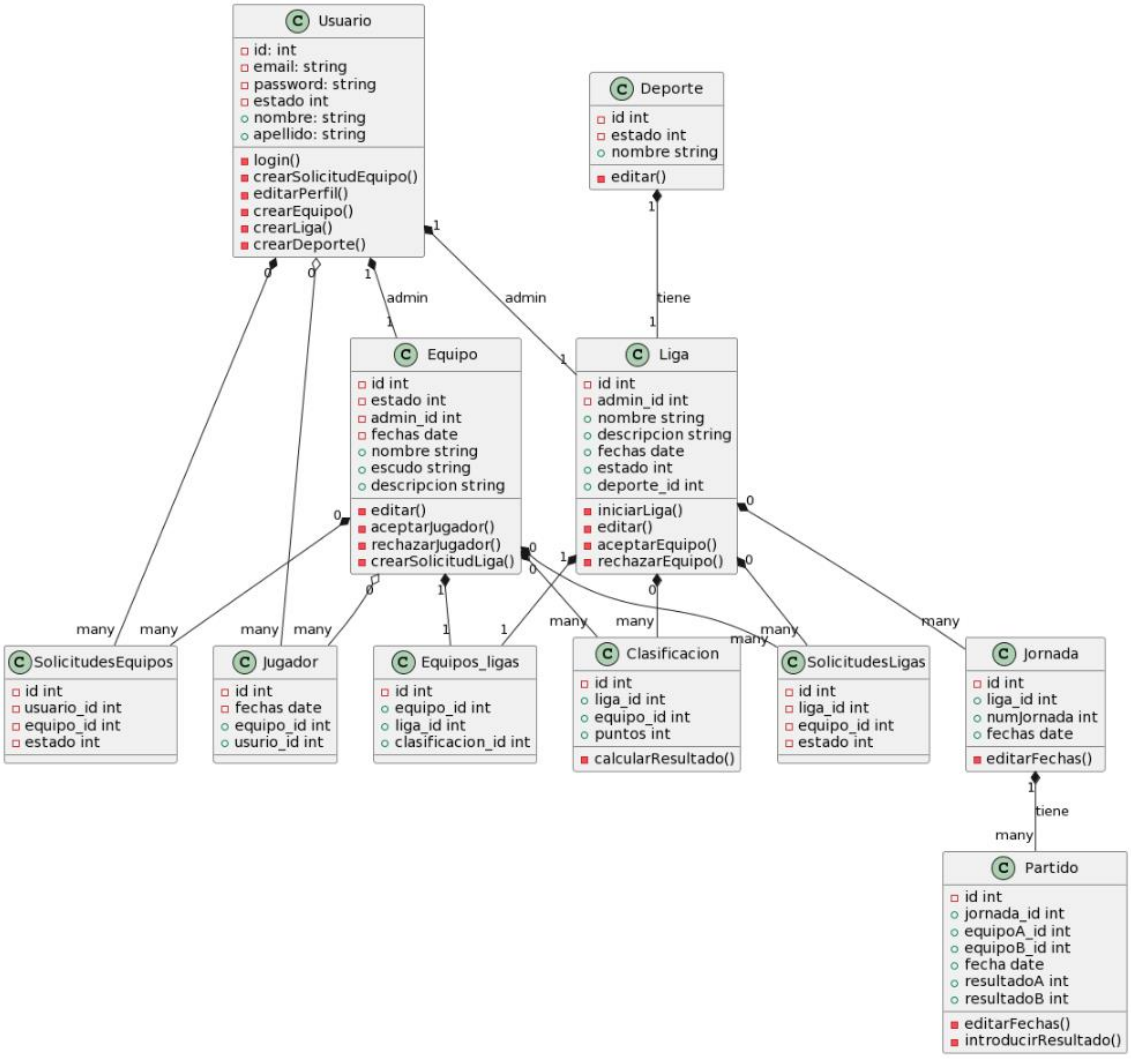


Figura 4-3 Diagrama de clases

4.2. DIAGRAMA DE ENTIDAD-RELACIÓN

En el diagrama entidad-relación se muestra la estructura de datos y las relaciones que hay entre las entidades que conforman las clases antes nombradas. Se indicarán las tablas y sus claves principales (PK, FK…) y cómo se relacionan entre ellas.

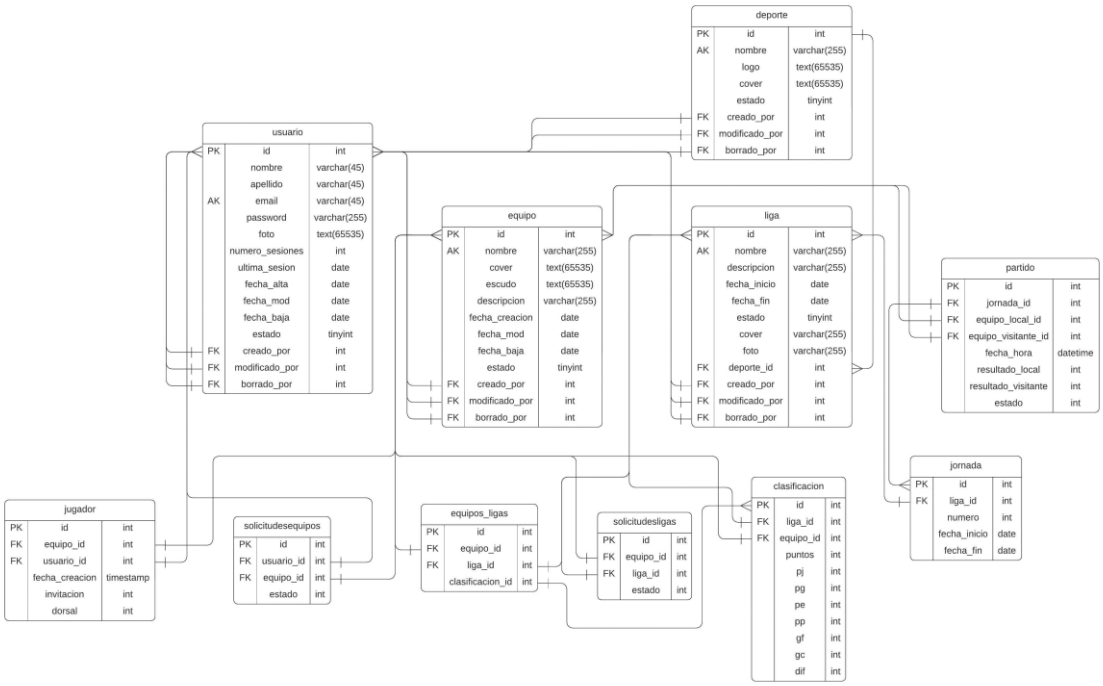
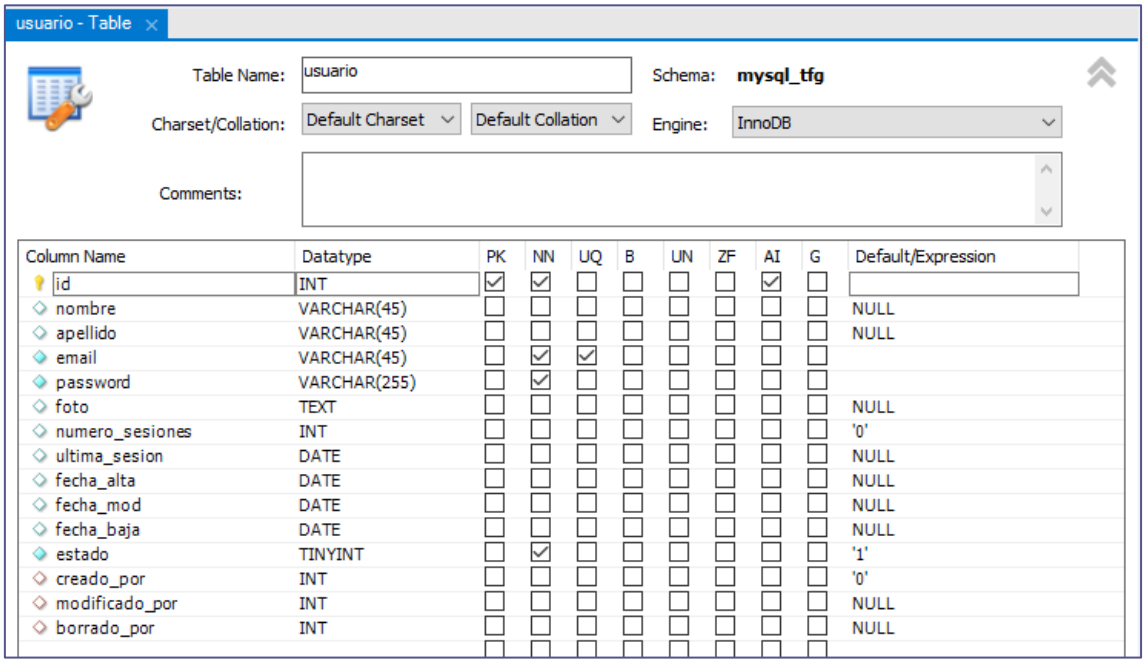


Figura 4-4 Diagrama Entidad-Relación

El modelo de datos empleado en el proyecto es una base de datos relacional (RDBMS), que son sistemas de gestión de bases de datos que utilizan tablas para almacenar y administrar datos. Son una opción sólida para almacenar y gestionar datos estructurados. Ofrecen un alto nivel de integridad de datos, seguridad y capacidad para realizar consultas complejas.

Usuario



Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id	INT	☑	☑	☐	☐	☐	☐	☑	☐	
nombre	VARCHAR(45)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
apellido	VARCHAR(45)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
email	VARCHAR(45)	☐	☑	☑	☐	☐	☐	☐	☐	
password	VARCHAR(255)	☐	☑	☐	☐	☐	☐	☐	☐	
foto	TEXT	☐	☐	☐	☐	☐	☐	☐	☐	NULL
numero_sesiones	INT	☐	☐	☐	☐	☐	☐	☐	☐	'0'
ultima_sesion	DATE	☐	☐	☐	☐	☐	☐	☐	☐	NULL
fecha_alta	DATE	☐	☐	☐	☐	☐	☐	☐	☐	NULL
fecha_mod	DATE	☐	☐	☐	☐	☐	☐	☐	☐	NULL
fecha_baja	DATE	☐	☐	☐	☐	☐	☐	☐	☐	NULL
estado	TINYINT	☐	☑	☐	☐	☐	☐	☐	☐	'1'
creado_por	INT	☐	☐	☐	☐	☐	☐	☐	☐	'0'
modificado_por	INT	☐	☐	☐	☐	☐	☐	☐	☐	NULL
borrado_por	INT	☐	☐	☐	☐	☐	☐	☐	☐	NULL

Figura 4-5 Modelo BD Usuario

- Id: clave primaria, no nula autoincremental para identificar de manera única a cada usuario
- Nombre, apellido y email: varchars con la información del usuario
- Password: varchar de mayor espacio debido a que durante su almacenamiento recibe un cifrado con el fin de no almacenar la contraseña en texto plano en la base de datos
- Foto, numero_sesiones y ultima_sesion: información adicional para dar más funcionalidades en un futuro
- Fecha_alta, fecha_mod, fecha_baja: registros para mantener información sobre posibles modificaciones del usuario
- Estado: variable de control que se evalúa en diferentes momentos de la aplicación para conocer si el usuario está dado de baja, si es jugador, gestor de ligas o administrador de la aplicación
- Creado_por, modificado_por y borrador_por: registros para mantener información sobre posibles modificaciones del usuario. Estos registros son a la vez Claves ajenas de otros registros de usuarios

Esquema lógico:

- PK: id
- FK: creado_por -> usuario(id)
- FK: modificado_por -> usuario(id)
- FK: borrado_por -> usuario(id)

Equipo

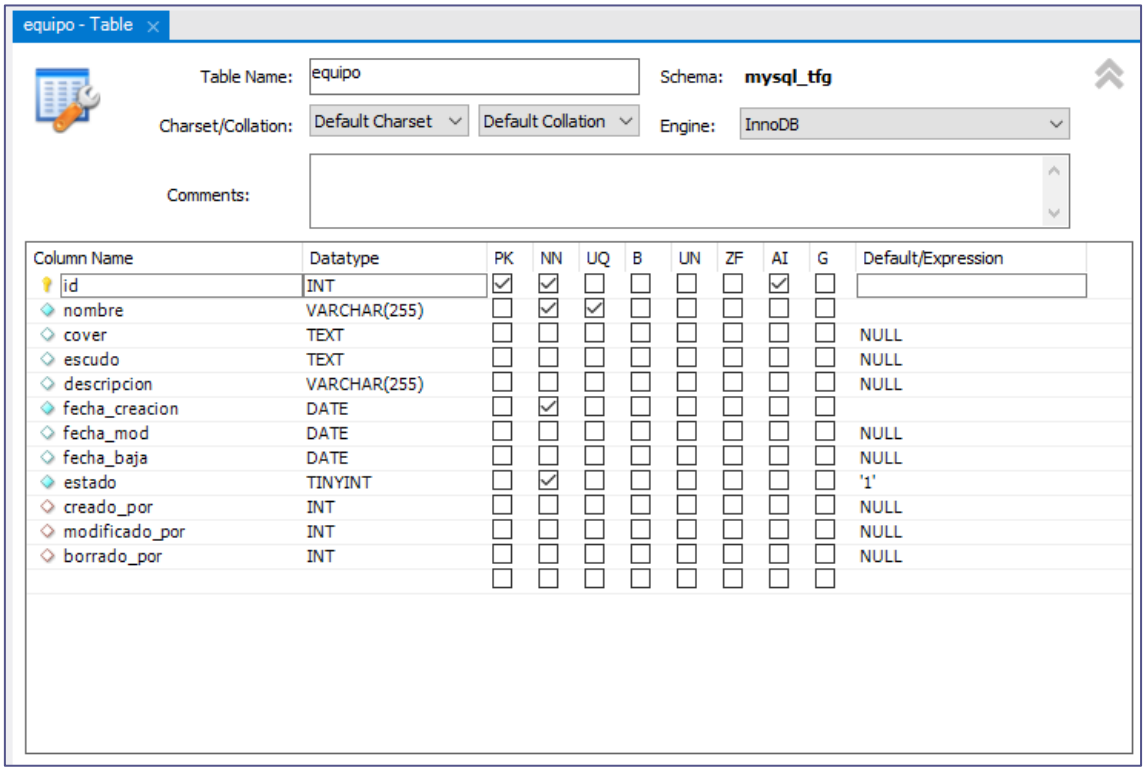


Figura 4-6 Modelo BD Equipo

- Id: clave primaria, no nula autoincremental para identificar de manera única a cada equipo
- Nombre: varchar con el nombre único de cada equipo
- Cover, escudo, descripción: varchars con la información del equipo
- Fecha_creacion, fecha_mod, fecha_baja: registros para mantener información sobre posibles modificaciones del equipo
- Estado: variable de control que se evalúa en diferentes momentos de la aplicación para conocer si el equipo admite o no nuevos jugadores
- Creado_por, modificado_por y borrador_por: registros para mantener información sobre posibles modificaciones del equipo. Estos registros son a la vez Claves ajenas de otros registros de usuarios

Esquema lógico:

- PK: id
- FK: creado_por -> usuario(id)
- FK: modificado_por -> usuario(id)
- FK: borrado_por -> usuario(id)

Ligas

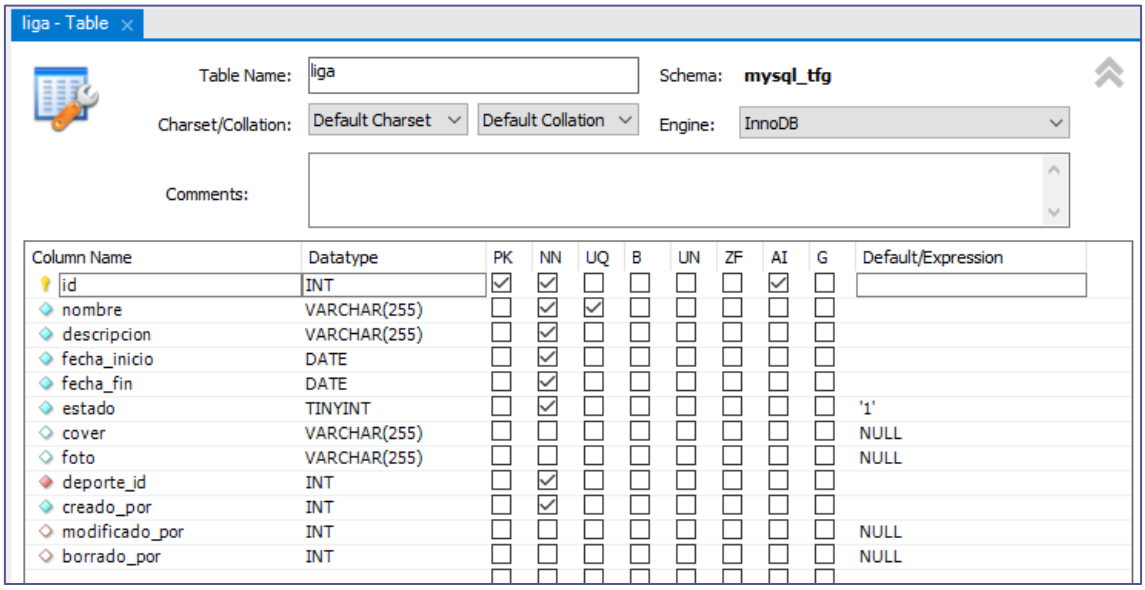


Figura 4-7 Modelo BD Liga

- Id: clave primaria, no nula autoincremental para identificar de manera única a cada liga
- Nombre: varchar con el nombre único de cada liga
- Cover, descripción y foto: varchars con la información del equipo
- Fecha inicio, fecha fin: registros de fecha para indicar cuándo comienza y cuándo termina la liga correspondiente
- Deporte_id: clave ajena de la tabla deportes, entrada que se usará para determinar el modelo de puntuación y clasificación de cada liga
- Estado: variable de control que se evalúa en diferentes momentos de la aplicación para conocer si la admite o no nuevos equipos
- Creado por, modificado por y borrador por: registros para mantener información sobre posibles modificaciones del equipo. Estos registros son a la vez Claves ajenas de otros registros de usuarios

Esquema lógico:

- PK: id
- FK: creado_por -> usuario(id)
- FK: modificado_por -> usuario(id)
- FK: borrado_por -> usuario(id)
- FK: deporte_id -> deporte(id)

Deporte

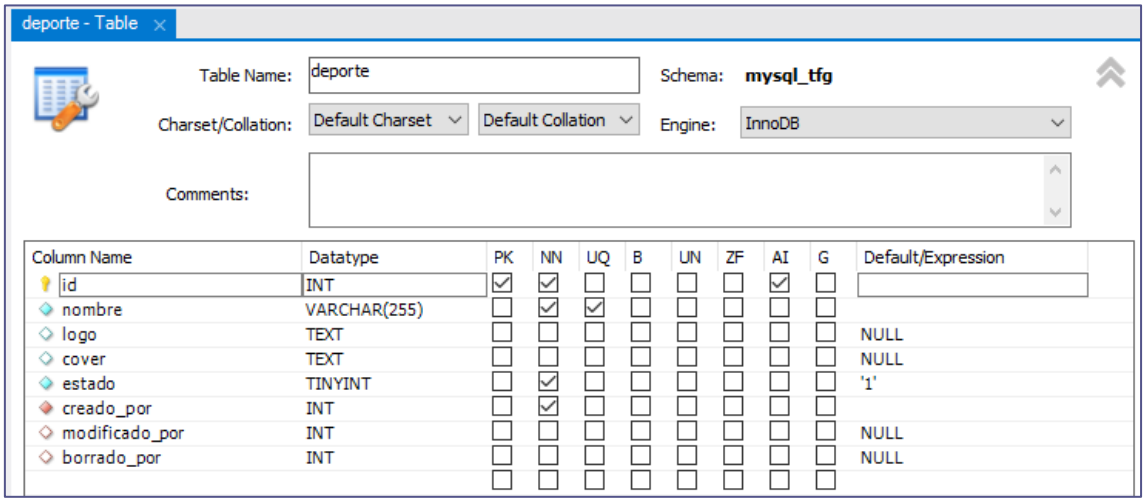


Figura 4-8 Modelo BD Deporte

- Id: clave primaria, no nula autoincremental para identificar de manera única a cada deporte
- Nombre: varchar con el nombre único de cada deporte
- Logo y cover: varchars con la información del equipo
- Fecha inicio, fecha fin: registros de fecha para indicar cuándo comienza y cuándo termina la liga correspondiente
- Estado: variable de control que se evalúa en diferentes momentos de la aplicación para conocer si el deporte esta activo y es seleccionable
- Creado por, modificado por y borrador por: registros para mantener información sobre posibles modificaciones del equipo. Estos registros son a la vez Claves ajenas de otros registros de usuarios

Esquema lógico:

- PK: id
- FK: creado_por -> usuario(id)
- FK: modificado_por -> usuario(id)
- FK: borrado_por -> usuario(id)

Jornadas

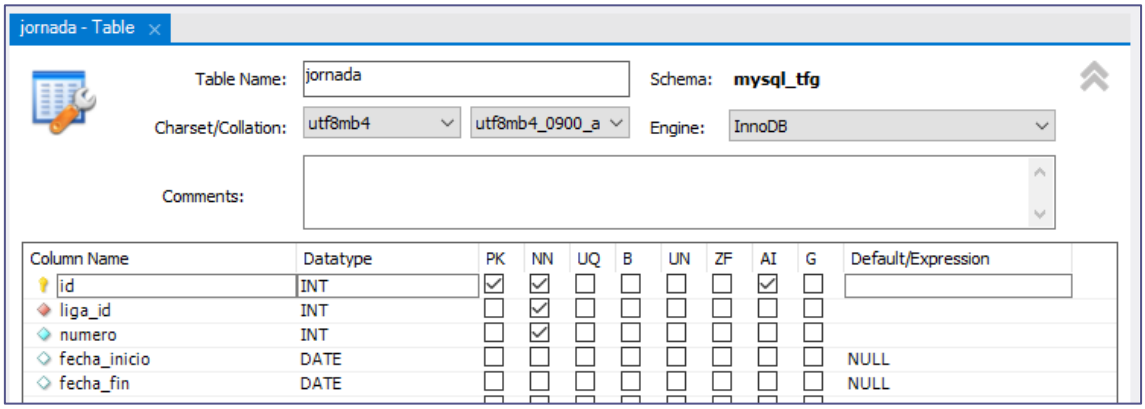


Figura 4-9 Modelo BD Jornada

- Id: clave primaria, no nula autoincremental para identificar de manera única a cada relación entre jornadas y ligas
- Numero: número de jornada para agilizar las consultas a bases de datos
- Liga_id: clave ajena con la tabla liga
- Fecha_inicio y fecha_fin: registros con las fechas donde se efectúan los partidos de la jornada

Esquema lógico:

- PK: id
- FK: liga_id -> liga(id)

Partidos

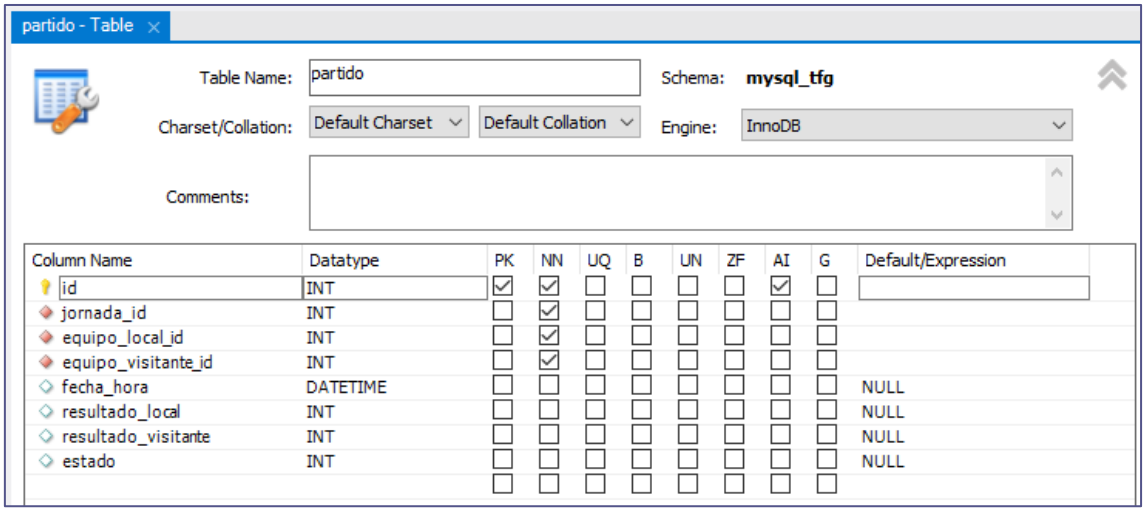


Figura 4-10 Modelo BD Partido

- Id: clave primaria, no nula autoincremental para identificar de manera única a cada partido
- Jornada_id: clave ajena con la tabla jornada

- Equipo_local_id y equipo_visitante_id: clave ajena con la tabla equipos para identificar los contrincantes del partido
- Fecha_hora: registros con las fechas donde se efectúa el partido
- Resultado_local y resultado_visitante: registros para calcular posteriormente la puntuación en la clasificación
- Estado: registro para en futuras mejoras, poder tener un control a tiempo real del resultado

Esquema lógico:

- PK: id
- FK: jornada_id -> jornada(id)
- FK: equipo_local_id -> equipo(id)
- FK: equipo_visitante_id -> equipo(id)

Clasificación

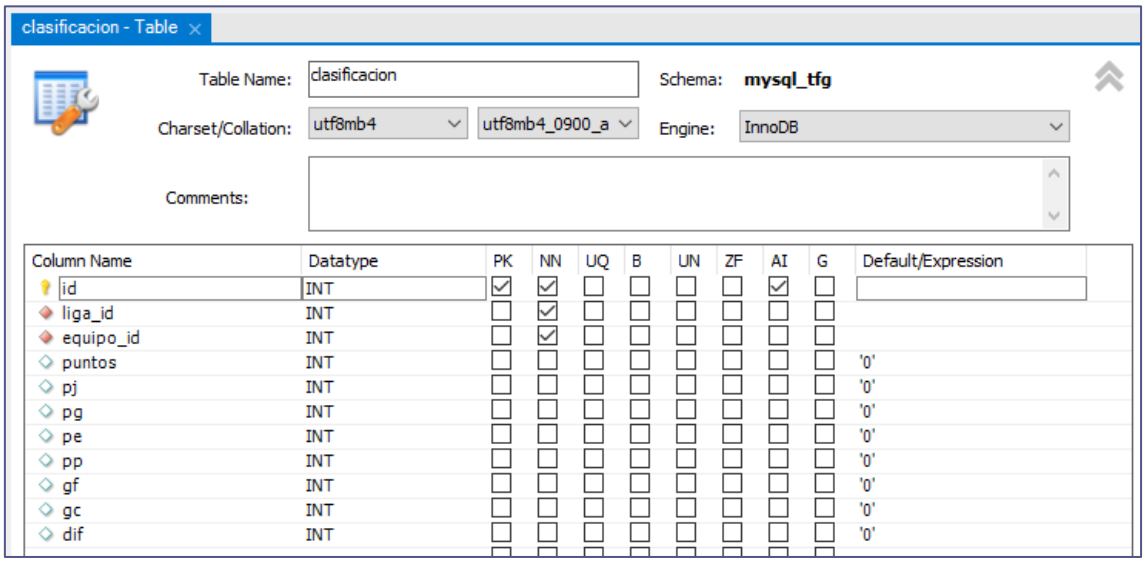


Figura 4-11 Modelo BD Clasificación

- Id: clave primaria, no nula autoincremental para identificar de manera única a cada partido
- Liga_id: clave ajena con la tabla liga
- Equipo_id: clave ajena con la tabla equipo para recuperar los resultados del equipo durante la competición
- Puntos y resto de registros: registros donde, en función de la liga (y su relación con cada tipo de deporte) se calculan los puntos en la clasificación, partidos jugados, tantos anotados y recibidos, etc.

Esquema lógico:

- PK: id
- FK: liga_id -> liga(id)
- FK: equipo_id -> equipo (id)

SolicitudesEquipos

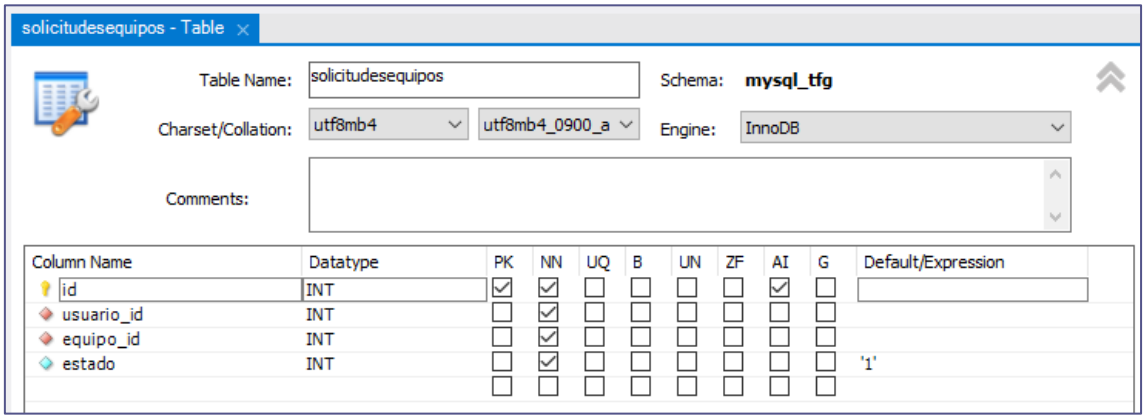


Figura 4-12 Modelo BD SolicitudesEquipos

- Id: clave primaria, no nula autoincremental para identificar de manera única a cada relación de solicitud entre un equipo (y su relación con el usuario creador) y un jugador que quiere unirse al equipo
- Usuario_id: clave ajena con la tabla usuario que indica qué usuario realiza la petición
- Equipo_id: clave ajena con la tabla equipo que indica a qué equipo quiere unirse el usuario
- Estado: estado de la solicitud, si esta ha sido aprobada o rechazada

Esquema lógico:

- PK: id
- FK: usuario_id -> usuario (id)
- FK: equipo_id -> equipo (id)

Jugador

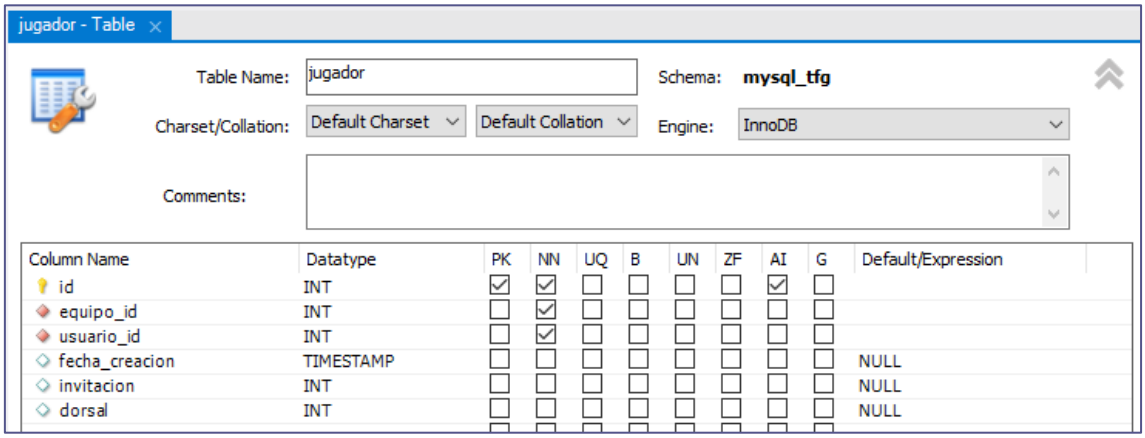


Figura 4-13 Modelo BD Jugador

- Id: clave primaria, no nula autoincremental para identificar de manera única a cada relación entre equipos y usuarios una vez se ha aprobado la solicitud
- Equipo_id: clave ajena con la tabla equipos. No nula, ya que la entrada en jugador se establece a la hora de agregar un usuario al equipo
- Usuario_id: clave ajena con la tabla usuario. No nula, ya que la entrada en jugador se establece a la hora de agregar un usuario al equipo
- Fecha_creacion: información de la fecha en la que el usuario se ha unido al equipo
- Invitación: variable de estado para gestionar la solicitud
- Dorsal: variable para mejoras futuras donde controlar a tiempo real las actuaciones de cada jugador; amonestaciones, anotaciones, etc.

Esquema lógico:

- PK: id
- FK: equipo_id -> equipo (id)
- FK: usuario_id -> usuario (id)

SolicitudesLigas

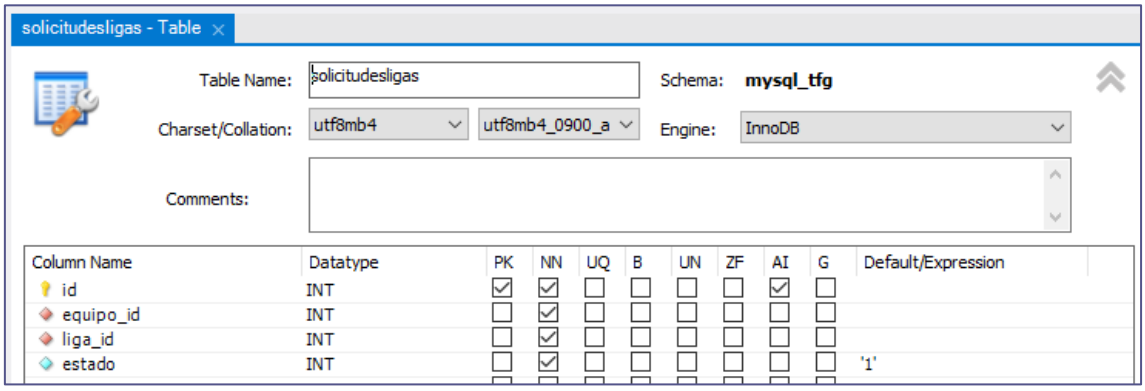


Figura 4-14 Modelo BD SolicitudesLigas

- Id: clave primaria, no nula autoincremental para identificar de manera única a cada relación de solicitud entre una liga (y su relación con el usuario creador) y un equipo (y su relación con el usuario creador) que quiere unirse a la liga
- Liga_id: clave ajena con la tabla liga que indica a qué liga se realiza la petición
- Equipo_id: clave ajena con la tabla equipo que indica qué equipo quiere unirse a la liga
- Estado: estado de la solicitud, si esta ha sido aprobada o rechazada

Esquema lógico:

- PK: id
- FK: equipo_id -> equipo (id)
- FK: liga_id -> liga(id)

Equipos_ligas

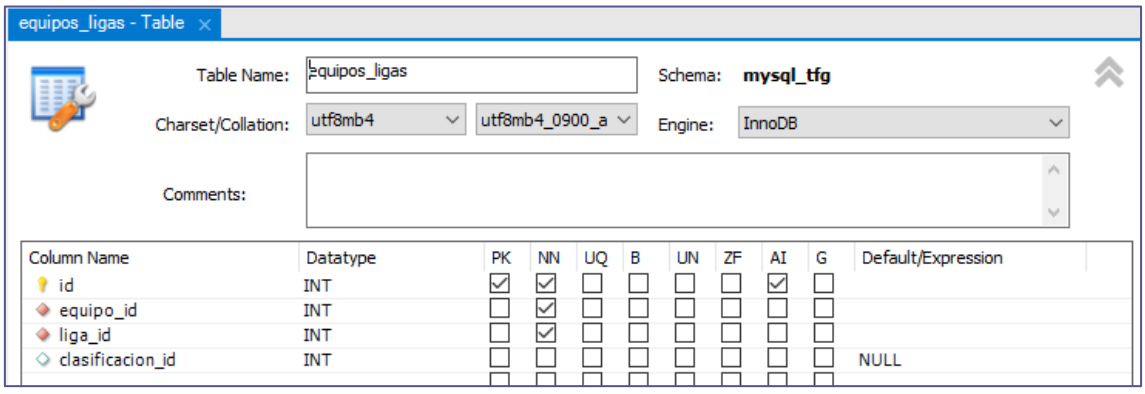


Figura 4-15 Modelo BD Equipos_ligas

- Id: clave primaria, no nula autoincremental para identificar de manera única a cada relación entre equipos y ligas una vez se ha aprobado la solicitud
- Equipo_id: clave ajena con la tabla equipos. No nula, ya que la entrada en equipos_ligas se establece a la hora de agregar un equipo a una liga
- Liga_id: clave ajena con la tabla liga. No nula, ya que la entrada en equipos_ligas se establece a la hora de agregar un equipo a una liga
- Clasificacion_id: clave ajena con la tabla clasificación. Se rellenará con el id correspondiente una vez se hayan generado las jornadas y partidos de la competición

Esquema lógico:

- PK: id
- FK: equipo_id -> equipo (id)
- FK: liga_id -> liga(id)

Notificaciones

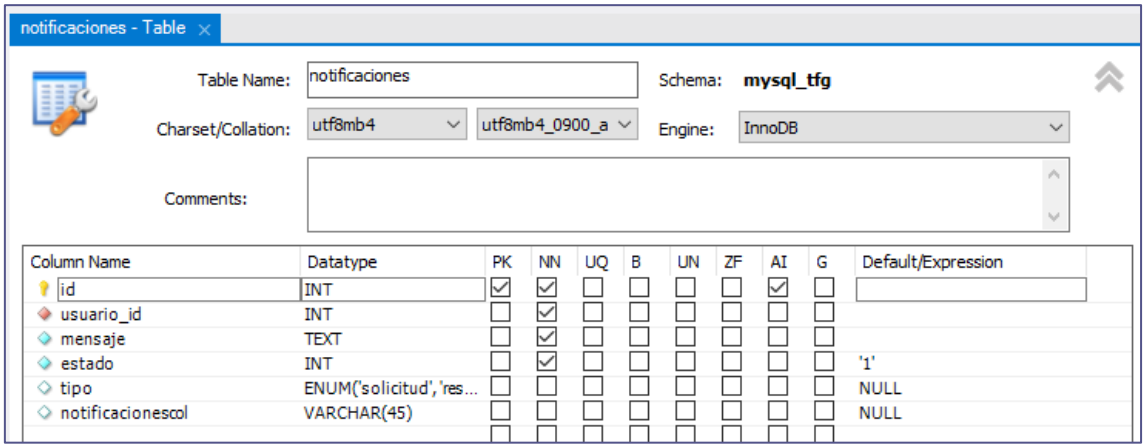


Figura 4-16 Modelo BD Notificaciones

- Id: clave primaria, no nula autoincremental para identificar de manera única cada notificación generada
- Usuario_id: clave ajena con la tabla usuario. No nula, ya que es necesario que se indique quién recibe la notificación
- Mensaje: cuerpo del mensaje de la notificación
- Tipo: el campo "tipo" en la tabla es útil para distinguir entre diferentes tipos de notificaciones, en este caso, entre notificaciones de solicitudes y notificaciones de respuestas
- Notificacionescol: para posible uso de colección de notificaciones

Esquema lógico:

- PK: id
- FK: usuario_id -> usuario(id)

Tabla actualmente en desuso. Está creada para la futura implementación de notificaciones en cualquier dirección que pueda ser relevante para los usuarios; solicitudes aceptadas, resultados añadidos, cambios de fecha en partidos en los que es jugador, etc.

4.3. FLUJO DE INFORMACIÓN

La navegabilidad en una aplicación es esencial para garantizar una experiencia de usuario fluida y efectiva. En este contexto, el diagrama de navegabilidad se convierte en una herramienta crucial para comprender cómo los usuarios interactuarán con la aplicación en la capa de lógica de negocio y cómo fluirá la información.

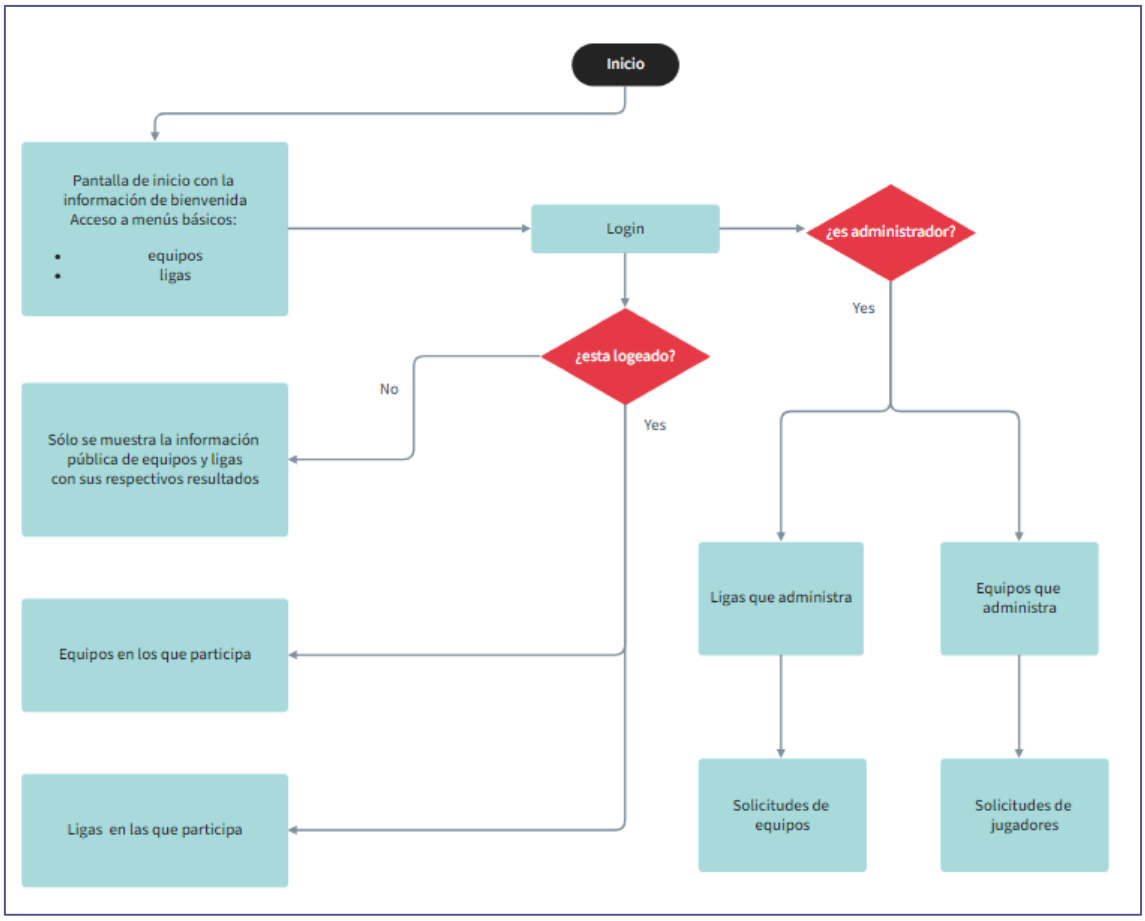


Figura 4-17 Diagrama de navegabilidad. Lógica de negocio

4.4. TECNOLOGÍAS Y HERRAMIENTAS UTILIZADAS

1. MySQL: La base de datos MySQL se utiliza para almacenar y gestionar la información relacionada con las ligas deportivas, equipos, partidos, usuarios y otros datos relevantes. Se utilizarán tablas y relaciones adecuadas para garantizar una gestión eficiente de los datos.

2. Servidor Privado FTP: Este servidor se utiliza para el almacenamiento de archivos y recursos multimedia relacionados con las ligas deportivas, como imágenes de equipos, logotipos y otros archivos relevantes. El servidor FTP garantiza la disponibilidad y la velocidad de acceso a estos recursos.

3. Estructura de Datos Modelo-Vista-Controlador (MVC): La arquitectura Modelo-Vista-Controlador (MVC) es un patrón de diseño ampliamente utilizado en el desarrollo de aplicaciones web y de software que busca separar la lógica de la aplicación en tres componentes principales: el Modelo, la Vista y el Controlador. A continuación, se presenta un resumen de los aspectos clave relacionados con MVC.

   Inicios de MVC:

   MVC fue conceptualizado en la década de 1970 por Trygve Reenskaug en el contexto de la programación orientada a objetos. Su objetivo inicial era estructurar las aplicaciones de interfaz de usuario en Smalltalk-80, un lenguaje de programación orientado a objetos. Desde entonces, se ha convertido en uno de los patrones de diseño más populares y ampliamente adoptados en el desarrollo de software.

   Usos Actuales:

   MVC se utiliza en una amplia variedad de aplicaciones web y de software en la actualidad. Se ha convertido en el estándar de facto para el desarrollo de aplicaciones web basadas en lenguajes como PHP, Ruby on Rails, Java Spring y muchos otros. También se emplea en aplicaciones de escritorio y móviles, ya que proporciona una estructura clara para separar las preocupaciones y facilita el mantenimiento y la escalabilidad del código.

   Aplicación en PHP:

   En PHP, MVC se utiliza para estructurar aplicaciones web de manera eficiente. El Modelo representa la capa de datos y la lógica de negocio, utilizando PHP para interactuar con la base de datos y gestionar la información. La Vista se encarga de la presentación de datos y utiliza HTML y PHP para renderizar las páginas web. El

Controlador actúa como intermediario entre el Modelo y la Vista, gestionando las solicitudes del usuario, procesando datos y actualizando la interfaz de usuario.

Pros:

Separación de preocupaciones: MVC separa las responsabilidades clave de una aplicación, lo que facilita el mantenimiento y la escalabilidad.

Reutilización de código: Los componentes de MVC pueden ser reutilizados en diferentes partes de la aplicación.

Claridad de diseño: Proporciona una estructura organizativa clara y legible para el código fuente.

Contras:

Mayor complejidad inicial: La implementación de MVC puede requerir un esfuerzo adicional en comparación con enfoques más simples.

Curva de aprendizaje: Los desarrolladores nuevos pueden necesitar tiempo para familiarizarse con los conceptos y la estructura de MVC.

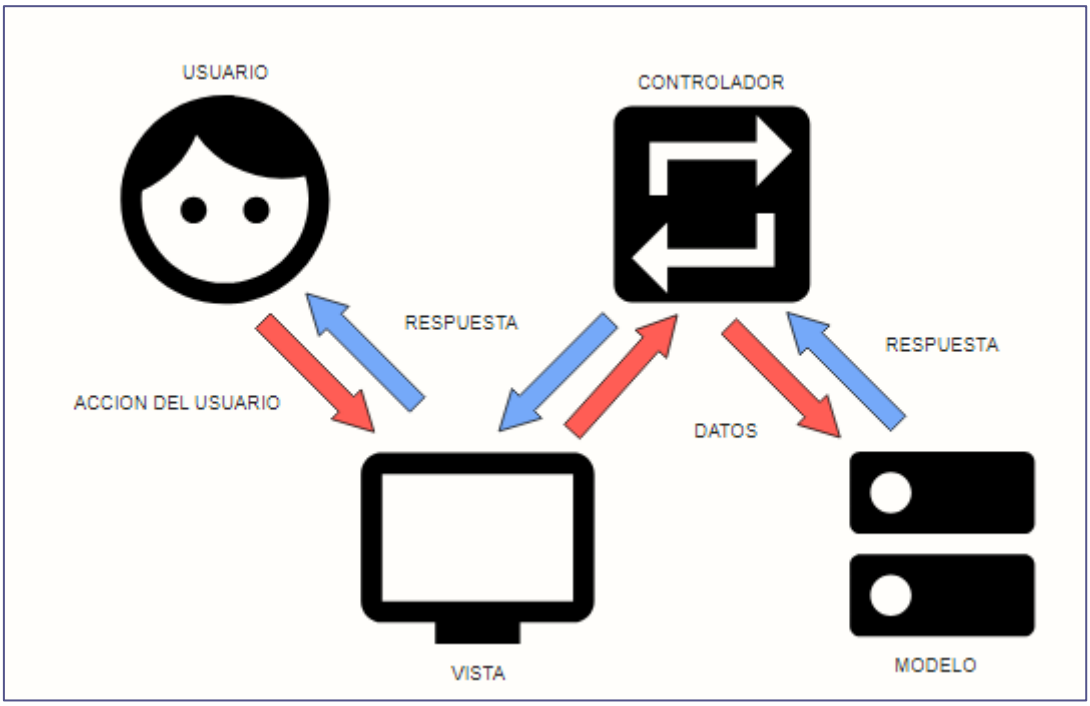


Figura 4-18 Imagen del flujo de datos en el Modelo MVC

4.5. FLUJO DE DATOS

El flujo de la aplicación se basa en el patrón MVC. Cuando un usuario accede a la aplicación a través de un navegador web, la solicitud se enruta al Controlador correspondiente. El Controlador interactúa con el Modelo para recuperar o modificar datos y actualiza la Vista para mostrar la información al usuario. Los formularios y acciones del usuario generan nuevas solicitudes que siguen este ciclo.

4.6. CONSIDERACIONES DE SEGURIDAD

La seguridad es una consideración clave en el diseño de la aplicación. Se han implementado prácticas de diseño y medidas de seguridad para proteger la base de datos MySQL contra ataques como inyecciones SQL. Además, de garantizar la autenticación segura de usuarios y se gestionarán adecuadamente las sesiones para prevenir el acceso no autorizado.

4.7. ESCALABILIDAD Y RENDIMIENTO

El diseño de la aplicación se centra en la escalabilidad y el rendimiento. Esto incluye la optimización de consultas a la base de datos, el uso eficiente de recursos del servidor y la posibilidad de ampliar la aplicación para gestionar un mayor número de ligas, equipos y usuarios.

La arquitectura y el diseño de la aplicación de gestión de ligas deportivas se basan en tecnologías sólidas, como MySQL y FTP, junto con el patrón MVC. Esto permite una gestión eficiente de datos, una interfaz de usuario atractiva y un alto nivel de seguridad. Además, se presta atención a la escalabilidad y el rendimiento para garantizar que la aplicación pueda crecer y adaptarse a las necesidades cambiantes de los usuarios.

5. Desarrollo e implementación

Primeras decisiones y problemas enfrentados

Durante los inicios del desarrollo, entraron sobre debate qué herramientas utilizar, tanto a nivel de desarrollo como de servidor. Dado que la propuesta del tutor era usar lenguaje PHP, lo más obvio para este caso era acompañarlo de un sistema de bases de datos en lenguaje SQL.

Una vez decididos los fundamentos, la siguiente cuestión planteada fue si desarrollar directamente en servidor o realizar primero el código en local e ir actualizando de manera regular un servidor alojado en un hosting. Trabajar en local es más rápido, sin embargo, hacerlo directamente en servidor dota de la posibilidad de ir enseñando los avances a tiempo real y da la seguridad de poder acceder al código desde cualquier ubicación. La decisión final fue realizarlo directamente en servidor.

Dado el bajo coste de los almacenamientos web y dominios, se contrató un hosting para almacenar y gestionar el proyecto.

Rápidamente aparecieron problemas con esta decisión, ya que fue imposible configurar un IDE que conectase por FTP al servidor de archivos y guardara a tiempo real los cambios realizados en el código. Cambiar de decisión no fue fácil, ya que parecía la más correcta. Tras perder bastante esta idea fue descartada, cambiando a desarrollar en local e ir actualizando el servidor con frecuencia.

Como herramienta para programar se comenzó a usar Sublime Text y la suit de desarrollo debido a su interfaz minimalista y simple.

Debido a experiencias anteriores, para la gestión del servidor MySQL y Apache en local se estudió la posibilidad de usar la herramienta XAMP. Pero en esta ocasión se tuvo que buscar alternativas, ya que el proyecto iba a ser desarrollado tanto en un ordenador con sistema operativo Windows como en otro portátil con MacOS. XAMP sólo está disponible para Windows. Después de un proceso de investigación la aplicación de Laragon fue la escogida para la gestión, ya que se encuentra disponible en ambos sistemas operativos.

Dada la necesidad de compartir el código entre ambos ordenadores, se intentó configurar una carpeta compartida en Google Drive o en iCloud donde mantener el código actualizado. Los archivos de base de datos generaban infinidad de archivos temporales que tardaban demasiado tiempo en actualizarse en la nube, generando inconsistencias constantemente.

En este momento se tomó la decisión de separar el código de los archivos de base de datos, y comenzar a utilizar un gestor de versiones, GIT.

Debido al poco conocimiento de la herramienta, nuevos problemas aparecieron entre versiones de las ramas generadas entre dispositivos. Haciendo que el repositorio fuera borrado y regenerado en repetidas ocasiones. Se buscaron soluciones a este

problema, dando con la posibilidad de usar Visual Studio Code junto a un plugin con Git-Hub para facilitar la carga (push) y descarga (pull) de las versiones del código. De nuevo, un cambio de estrategia en las herramientas principales de desarrollo con el uso de VSCode.

Poco tiempo después, en una de las ocasiones en las que se subían manualmente los archivos al FTP mediante la herramienta web de Plesk, se pudo comprobar que existe un plugin en Plesk que sincroniza directamente los archivos con el repositorio de Git-Hub.

Gracias a esto fue cuando se consiguió de manera definitiva un entorno de trabajo cómodo para continuar con el desarrollo del código del proyecto presente.

5.1. TECNOLOGÍA UTILIZADA

En el desarrollo del proyecto, se han utilizado diversas tecnologías tanto en el servidor como en el cliente de desarrollo para lograr la implementación de la aplicación web. Estas tecnologías contribuyeron al desarrollo, la gestión de la base de datos, la administración de contenido multimedia y el control de versiones del código fuente. A continuación, se detallan las tecnologías utilizadas tras las diferentes decisiones tomadas después de solventar los problemas comentados en el apartado anterior:

Servidor:

- Plesk: panel de control de hosting que se utilizó para gestionar y administrar el servidor web y las bases de datos. Proporciona una interfaz gráfica de usuario que facilita la configuración de servicios como Apache, PHP y MySQL, así como la administración de dominios y cuentas de correo electrónico.
- PHPMyAdmin: herramienta de administración de bases de datos MySQL a través de una interfaz web. Se empleó para interactuar con la base de datos del proyecto, realizar consultas SQL, gestionar tablas y asegurar la integridad de los datos.
- GIT: se ha utilizado como sistema de control de versiones para rastrear y gestionar cambios en el código fuente del proyecto.
- FTP (Protocolo de Transferencia de Archivos): para la transferencia de archivos multimedia, como imágenes y archivos relacionados con el contenido de la aplicación. Facilitó la carga y descarga de recursos multimedia desde el servidor.

Cliente de Desarrollo:

- Visual Studio Code (VS Code): entorno de desarrollo integrado (IDE) altamente personalizable y ampliamente utilizado. Se utilizó como el principal entorno de desarrollo para escribir y depurar el código fuente de la aplicación web.
- GIT: además de su uso en el servidor, Git también se implementó en el cliente de desarrollo para el control de versiones del código fuente.

- Laragon: plataforma de desarrollo local que permite configurar fácilmente un entorno de servidor web Apache, bases de datos MySQL y PHP en el sistema local de un desarrollador. Se utilizó para desarrollar y probar la aplicación en un entorno de desarrollo local antes de implementarla en el servidor de producción.

Modelo Vista Controlador y ausencia del uso de framework

Para adquirir conocimientos de desarrollo en PHP se han realizado cursos en la plataforma web https://laracasts.com/. En ella, se inicia al alumno en el desarrollo de elementos de middleware para conocer la arquitectura del modelo MVC. Creando también las bases de lo que son los fundamentos en los frameworks profesionales.

Debido a esto, se ha continuado con el uso de estructuras empleadas en los cursos para el total desarrollo de la aplicación. Y pese a que es cierto que esto puede crear fallas de seguridad a nivel de proyecto, se ha valorado más el aprendizaje del alumno teniendo que crear estos modelos sobre la integridad del proyecto. Siendo esto fácilmente implementable en futuras versiones en el caso de que el proyecto llegue a tener un uso real como aplicación fuera del ámbito académico.

La organización del código empleando VSCode y la estructura de MVC queda de la siguiente manera:

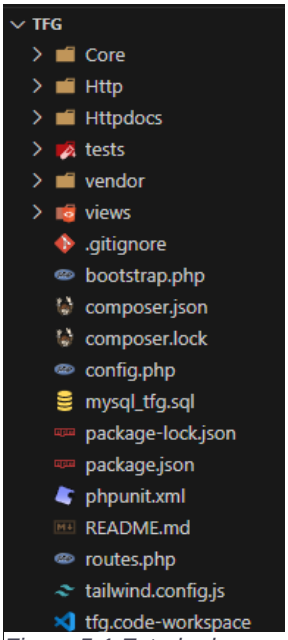


Figura 5-1 Estado de carpetas

En ella se pueden observar carpetas como contenedores para separar los diferentes ámbitos que tienen cada uno de los ficheros de código. Separando así las funciones de cada uno de ellos, y evitando posibles filtraciones maliciosas.

Existe una parte **Core** donde se almacenan los archivos que constituyen los pilares de la estructura de datos, siendo posible migrar esta misma estructura a otros proyectos cambiando pocas líneas de código.

Http corresponde a la ubicación de los **controladores**, que serán archivos de código donde se valide y se cargue la información necesaria para ser mostrada en las vistas.

Httpdocs es la puerta de enlace al exterior, donde el usuario final accede y donde se redirige a las diferentes partes del contenido.

Test contiene las pruebas unitarias empleadas para validar la lógica del código.

En **Vendor** se almacenan algunas funciones de PHP empleadas durante la ejecución de la aplicación.

La carpeta **Views** tiene los archivos que finalmente son mostrados por pantalla, y que recogen la información proporcionada por los controladores. También es donde se encuentran los **formularios** donde el usuario interactúa con la aplicación introduciendo los datos necesarios.

El resto de archivos contiene información variada; los archivos git y de workspace son necesarios para el control de versiones y el entorno de trabajo del propio IDE VSCode. Bootstrap y TailWind son conjuntos de funciones para las hojas de estilos CSS empleadas durante el proyecto.
config.php es donde se encuentra almacenada la configuración de acceso remoto con las credenciales de la base de datos que se emplean para las conexiones durante las consultas. Mientras que **routes.php** es el archivo que almacena la gestión de rutas y concesión de accesos a las diferentes partes de la aplicación en función del rol del que dispongas a nivel de usuario.
El resto de archivos conforman las estructuras de datos para la base de datos.

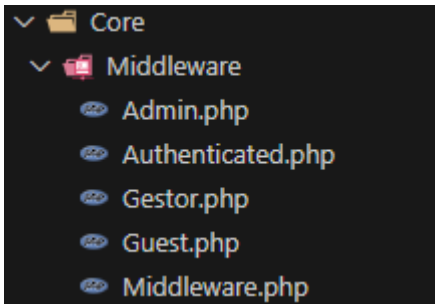


Figura 5-2 Carpeta Middleware y ficheros php.

Dentro de la carpeta de Middleware se comprueban los roles que conceden acceso a los usuarios para las diferentes partes de la aplicación.

Así como funciones que conceden o restringen el acceso en determinadas circunstancias.

En el siguiente código se ven las diferentes opciones de niveles en cuanto a roles pueden adquirir los usuarios y su gestión mediante middleware.

```php
namespace Core\Middleware;

2 references | 0 implementations | Viking Vela, hace 3 semanas |
class Middleware
{
    1 reference
    public const MAP = [
        'guest' => Guest::class,
        'auth' => Authenticated::class,
        'gestor' => Gestor::class,
        'admin' => Admin::class
    ];
```

Figura 5-3 Clase Middleware

Dentro de la clase del Middleware se han diseñado cuatro modelos de usuario que más adelante serán empleados en el router para indicar a qué ubicaciones puede o no puede acceder según qué tipo de usuario esté intentando acceder.

Las funcionalidades de los usuarios se basan en la pertenencia estos perfiles.

Se ha creado una clase para realizar las conexiones a base de datos, también en ella, se han diseñado funciones que facilitan enormemente consultas, agilizando todo el proceso en la parte de controllers de la aplicación.

```php
class Database {
    2 references
    public $connection;
    4 references
    public $statement;
    1 reference | 0 overrides
    public function __construct($config, $username = 'root', $password = ''){
        $dsn = 'mysql:' . http_build_query($config, '', ';');

        $this->connection = new PDO($dsn, $username, $password, [
            PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC
        ]);
    }
    0 references | 0 overrides
    public function insert($table, $data){
        $fields = array_keys($data);
        $sql = sprintf(
            'INSERT INTO %s (%s) VALUES (%s)',
            $table,
            implode(', ', $fields),
            ':' . implode(', :', $fields)
        );

        $this->query($sql, $data);       You, hace 2 meses • insertar en depor

        return $this;
    }
    3 references | 0 overrides
    public function query($query, $params = []){
        $this->statement = $this->connection->prepare($query);
        $this->statement->execute($params);
        return $this;
    }
    0 references | 0 overrides
    public function updateID($table, $id, $data){
        $data['id'] = $id;
        $fields = array_keys($data);
        $update_fields = [];
        foreach ($fields as $field) {
            $update_fields[] = $field . '=:' . $field;
        }

        // Crear sentencia SQL
        $sql = sprintf(
            'UPDATE %s SET %s WHERE id=:id',
            $table,
            implode(', ', $update_fields)
        );

        $this->query($sql, $data);
        return $this;
    }
```

Figura 5-4 Clase Database

En el apartado de **controllers** se puede comprobar como cada uno de los elementos definidos en el diseño de base de se divide en partes en las que dentro del modelo MVC es posible controlar todo el flujo.

Elementos **create.php** página inicial donde se muestra el formulario inicial para dar de alta algún elemento.

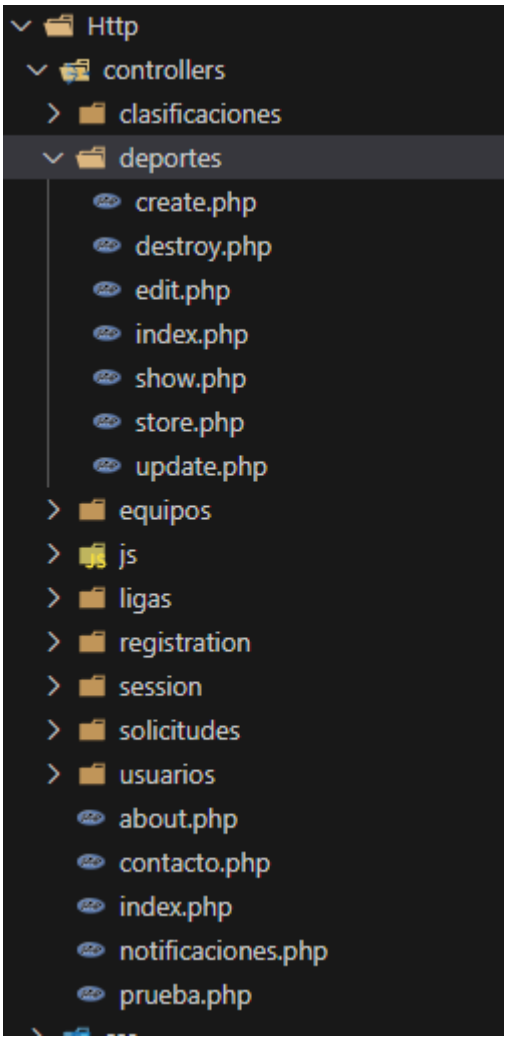


Figura 5-5 Archivos controllers MVC

Cuando la información es introducida por el usuario, esta se valida antes de ser almacenada en base de datos mediante el código que hay en **store.php.**

Cuando se desea listar el conjunto de elementos de un mismo modelo se emplea el **index.php** donde se encuentra la colección de estos elementos.

Si se desea ver los datos concretos de un elemento se ejecutará **show.php**, con la información detallada del seleccionado.

Si ahora se quiere editar este elemento seleccionado, el código lanzara **edit.php** donde, de nuevo mediante un formulario,

se carga la información ya almacenada del elemento, pero con la función de editarla.

Para finalizar y validar los datos del elemento que se ha editado se ejecuta **update.php.**

Por último y para des referenciar algún elemento en base de datos, cambiar su estado a 0 y que no sea visible para el usuario, etc. Se ejecutará **destroy.php**.

Las páginas como **create**, **index**, **show** y **edit** que muestran contenido al usuario, siempre tienen una redirección a su view correspondiente. Donde se encuentra el código HTML/CSS con las llamadas y funciones PHP necesarias para la carga del backend.

Una vez el controlador redirige a la vista, se carga el HTML que visualiza finalmente el usuario.

A continuación, se muestra un ejemplo del código, y capturas de la aplicación final.

```
<main>
<div class="m-5 px-8 py-6 bg-white rounded grid">
  <form action="/usuarios" method="POST" enctype="multipart/form-data">
    <input type="hidden" name="_method" value="PATCH">
    <input type="hidden" name="id" value="<?= $usuario['id'] ?>">
    <div class="space-y-12">
      <div class="border-b border-gray-900/10 pb-12">
        <h2 class="text-base font-semibold leading-7 text-gray-900">Perfil de usuario</h2>
        <div class="mt-4 grid grid-cols-1 gap-x-6 gap-y-8 sm:grid-cols-4">
          <div class="col-span-1 ">
            <div class="mt-5 flex items-center gap-x-3">
              <svg class="h-12 w-12 text-gray-300" viewBox="0 0 24 24" fill="currentColor" aria-hidden="true">
                <path fill-rule="evenodd" d="M18.685 19.097A9.723 9.723 0 0021.75 12c0-5.385-4.365-9.75-9.75-9.75S2.25 6.615 2.25 12a9.723 9.
              </svg>
              <button type="button" class="rounded-md bg-white px-2.5 py-1.5 text-sm font-semibold text-gray-900 shadow-sm ring-1 ring-inset
            </div>
          </div>
          <div class="col-span-1">
            <label for="nombre" class="block text-sm font-medium leading-6 text-gray-900"></label>
            <div class="mt-2">
              <input type="text" class="block w-full rounded-md border-0 py-1.5 text-gray-900 shadow-sm ring-1 ring-inset ring-gray-300 place
                name="nombre"
                id="nombre"
                placeholder="Nombre"
                required
                value="<?=isset($usuario['nombre']) ? $usuario['nombre'] : "" ?>">
            </div>
            <?php if (isset($errors['apellido'])) : ?><p class="text-red-500 text-xs mt-2"><?= $errors['apellido'] ?></p><?php endif; ?>
          </div>
          <div class="col-span-1">
            <label for="apellido" class="block t->only('auth')ext-sm font-medium leading-6 text-gray-900"></label>
            <div class="mt-2">
              <input type="text" class="block w-full rounded-md border-0 py-1.5 text-gray-900 shadow-sm ring-1 ring-inset ring-gray-300 place
                name="apellido"
                id="apellido"
                placeholder="Apellido"
                required
                value="<?=isset($usuario['apellido']) ? $usuario['apellido'] : "" ?>">
              </div>
              <?php if (isset($errors['apellido'])) : ?><p class="text-red-500 text-xs mt-2"><?= $errors['apellido'] ?></p><?php endif; ?>
          </div>
```

Figura 5-6 Ejemplo de código HTML con PHP incrustado

Cómo se puede ver en la captura, hay partes del código en PHP que carga contenido ya almacenado en la base de datos, creando la posibilidad de recuperar y editar esta información.

Durante el desarrollo del MVC, se han tenido que plantear conceptos como:

- Separar la lógica de las plantillas. De modo que la información quede validada en cada paso que da el usuario
- Creación de parciales, como la barra de navegación que se encuentra constantemente en la parte superior, y que cambia en función del rol y permisos que tiene el usuario logeado, dando acceso a páginas que otros usuarios no tienen
- Enrutado del flujo de navegación, dirigiendo al usuario en cada momento según sus necesidades e impidiendo que se salga de este camino evitando posibles vulnerabilidades
- Organización del proyecto acorde a las convenciones
- Gestión de sesión de usuario, creando y flasheando los datos almacenados en la caché

5.2. CASOS DE USO EN LA APLICACIÓN FINAL

CU01: Registro de un usuario

Descripción: Este caso de uso permite a un usuario registrarse en la aplicación.

El usuario accede desde la página de inicio

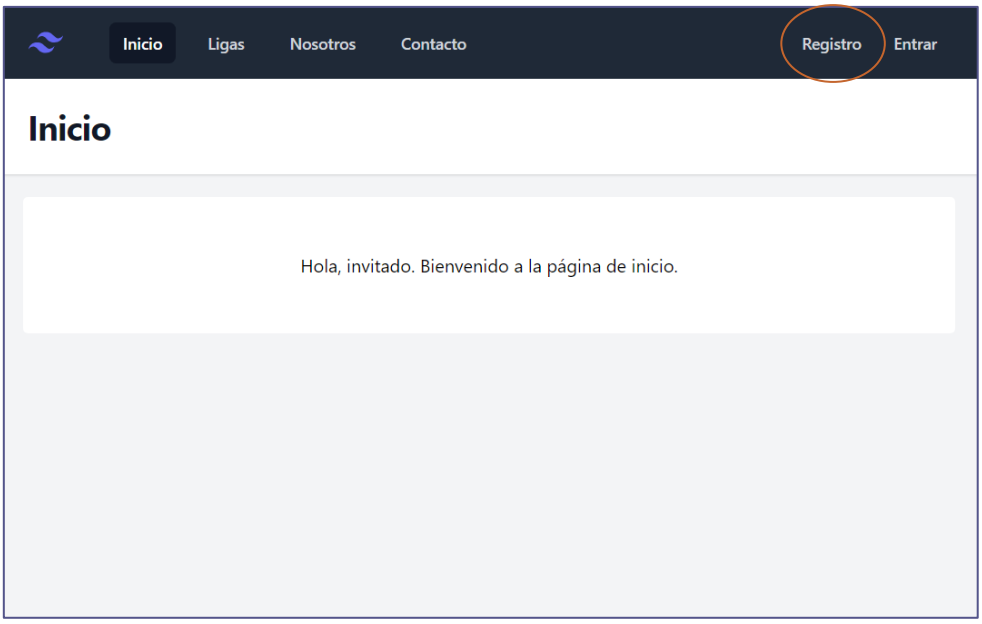


Figura 5-7 Aplicación final, pantalla de inicio

El usuario proporciona su dirección de correo electrónico y contraseña.

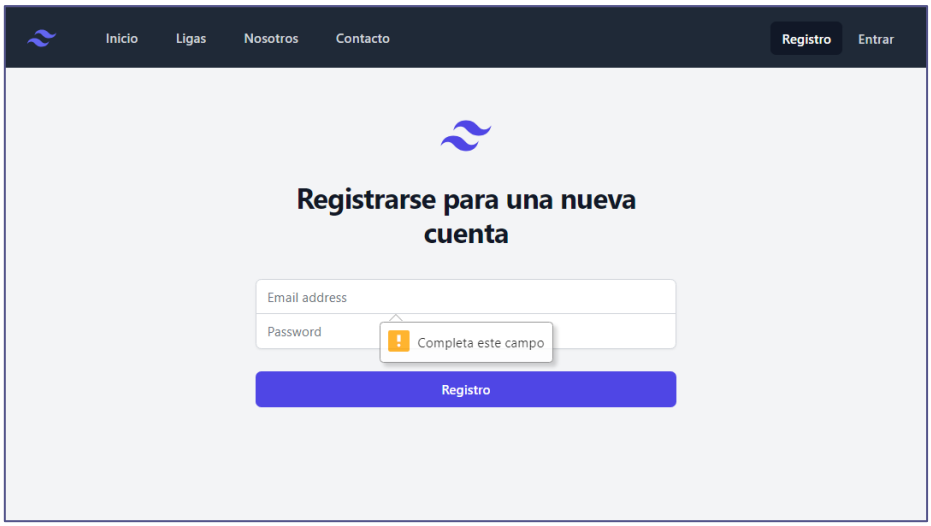


Figura 5-8 Aplicación final, pantalla de inicio

El usuario es redirigido a su perfil para completar sus datos personales ya que es su primer inicio de sesión.

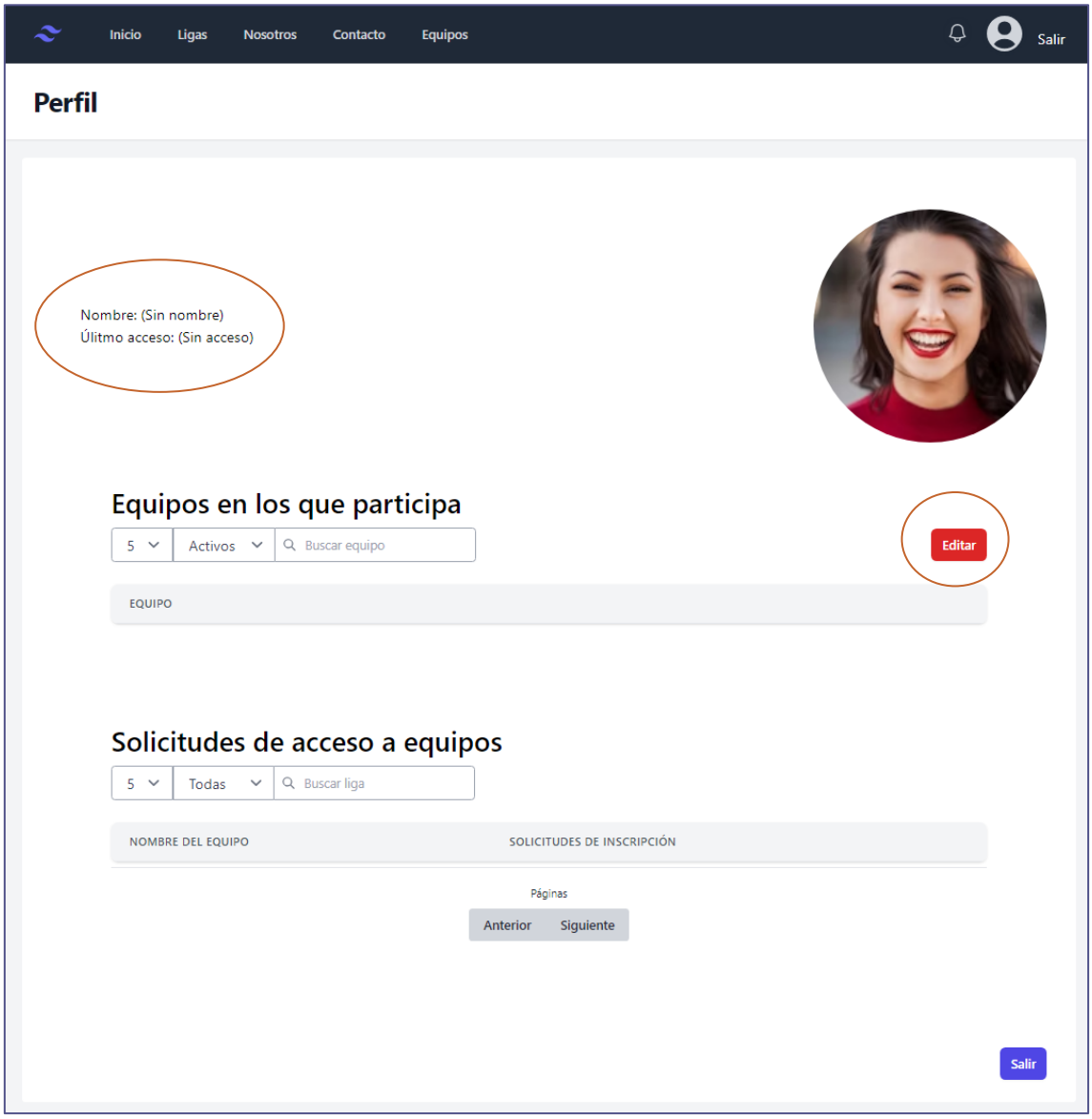


Figura 5-9 Aplicación final, pantalla de usuario

CU02: Creación de un equipo

Descripción: Este caso de uso permite a un usuario registrarse en la aplicación.

El usuario accede al apartado de equipos, presiona sobre nuevo, y accede al formulario de creación de equipos.

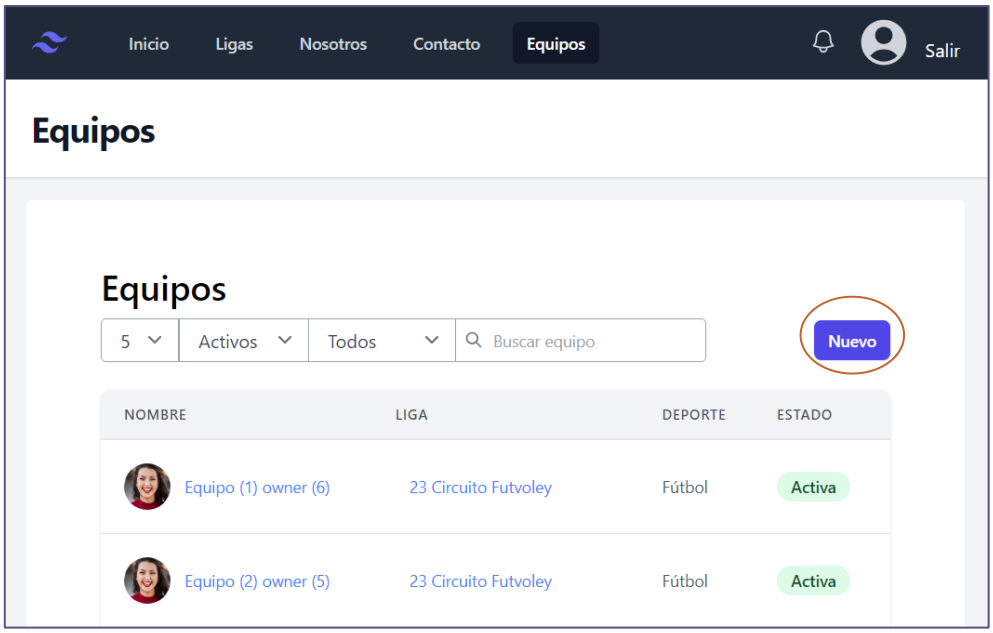


Figura 5-10 Aplicación final, el usuario crea un nuevo equipo

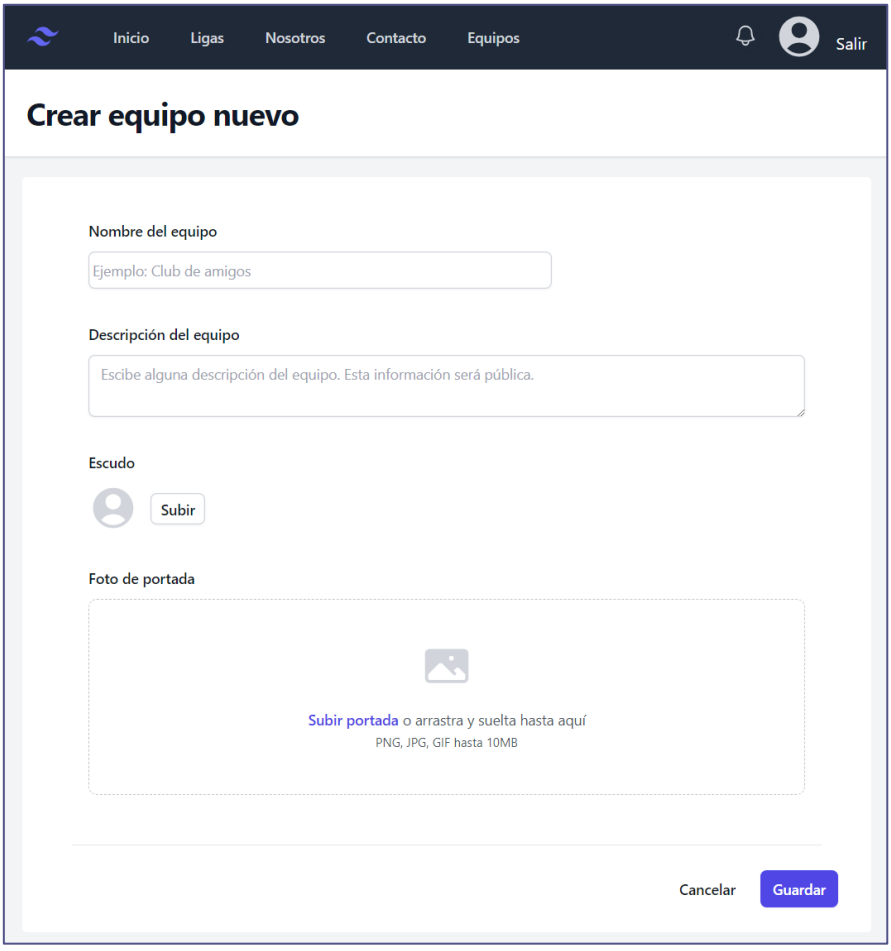


Figura 5-11 Aplicación final, formulario de nuevo equipo

Ahora, en la parte de Equipos, además del listado total de equipos, aparecerá un nuevo menú con los equipos administrados por el usuario actual.

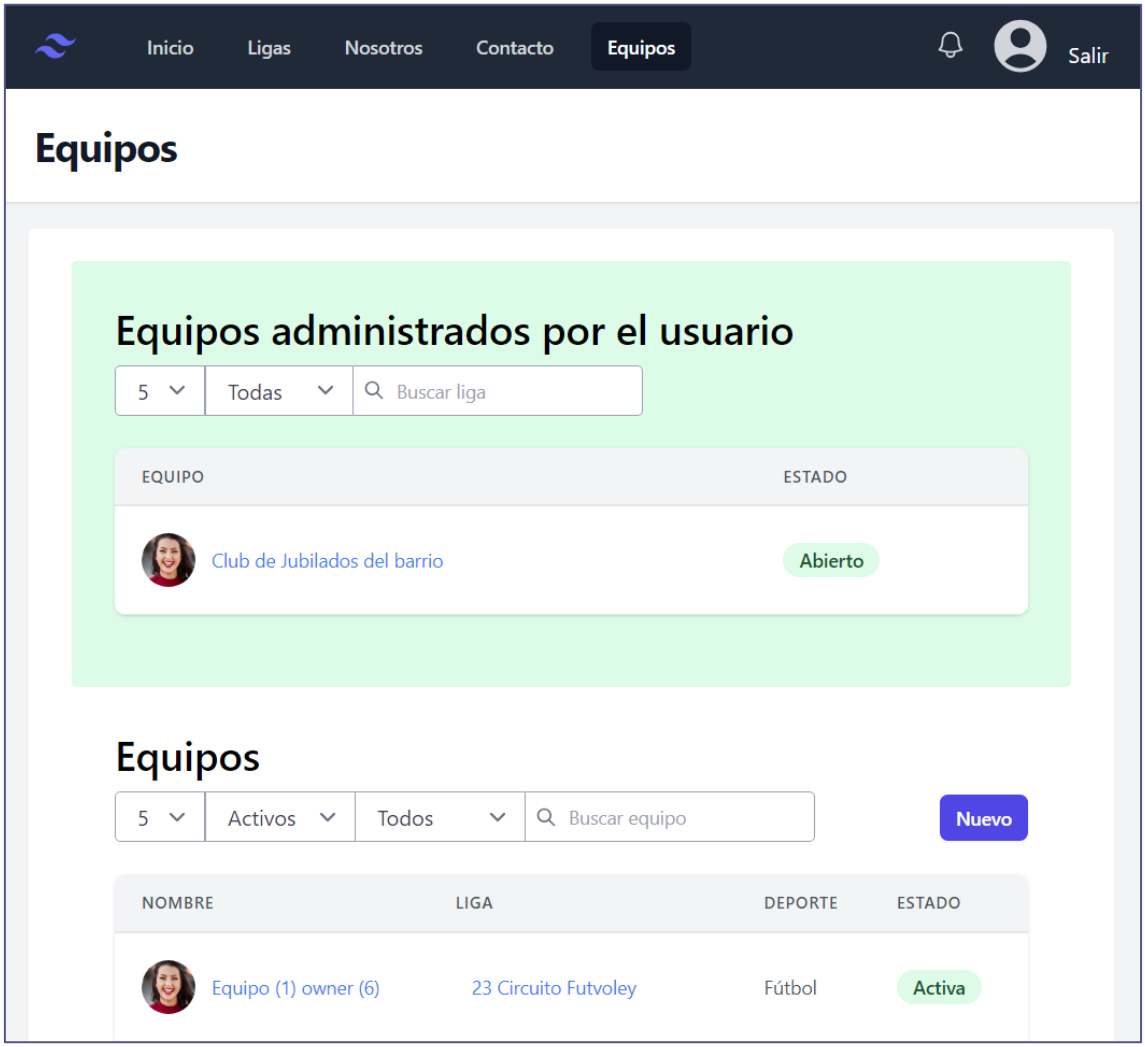


Figura 5-12 Aplicación final, equipos siendo administrador de algún equipo

CU03: Aceptar solicitudes de jugadores y equipos

Descripción: En este caso, un administrador de ligas y de equipos, acepta solicitudes pendientes.

Dentro del equipo, en la parte inferior, aparecen las solicitudes de jugadores que han solicitado el acceso al equipo del que el usuario es administrador.

Figura 5-13 Aplicación final, administrador aprobando solicitud de acceso

Una vez el jugador es aceptado por el administrador, la petición deja de aparecer en la parte de solicitudes, para hacer que el jugador forme parte del equipo.

En este momento, y de manera interna, se ha creado un registro en la tabla jugador, que vincula al usuario con el equipo.

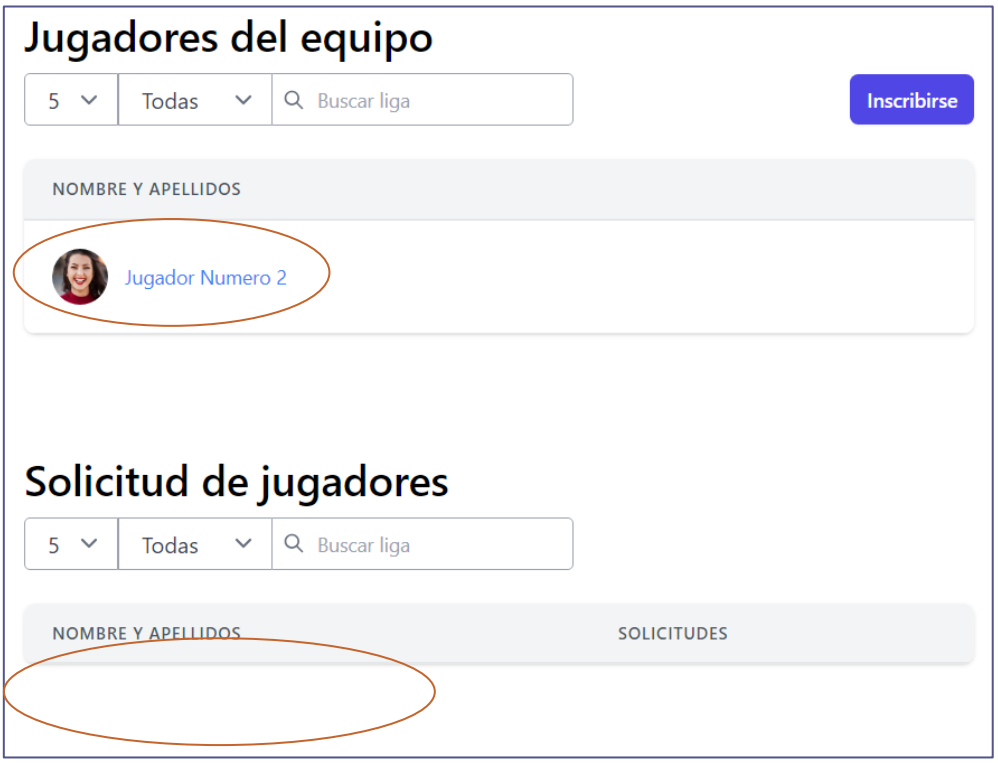


Figura 5-14 Aplicación final, solicitud de acceso aprobada

CU04: Generar partidos y clasificación de una liga

Descripción: Este caso de uso permite a un administrador de liga dar por comenzada la liga.

El usuario administrador de la liga, accede a la liga que desea dar comienzo, y una vez validados los equipos que van a componerla, presiona el botón de comenzar liga.

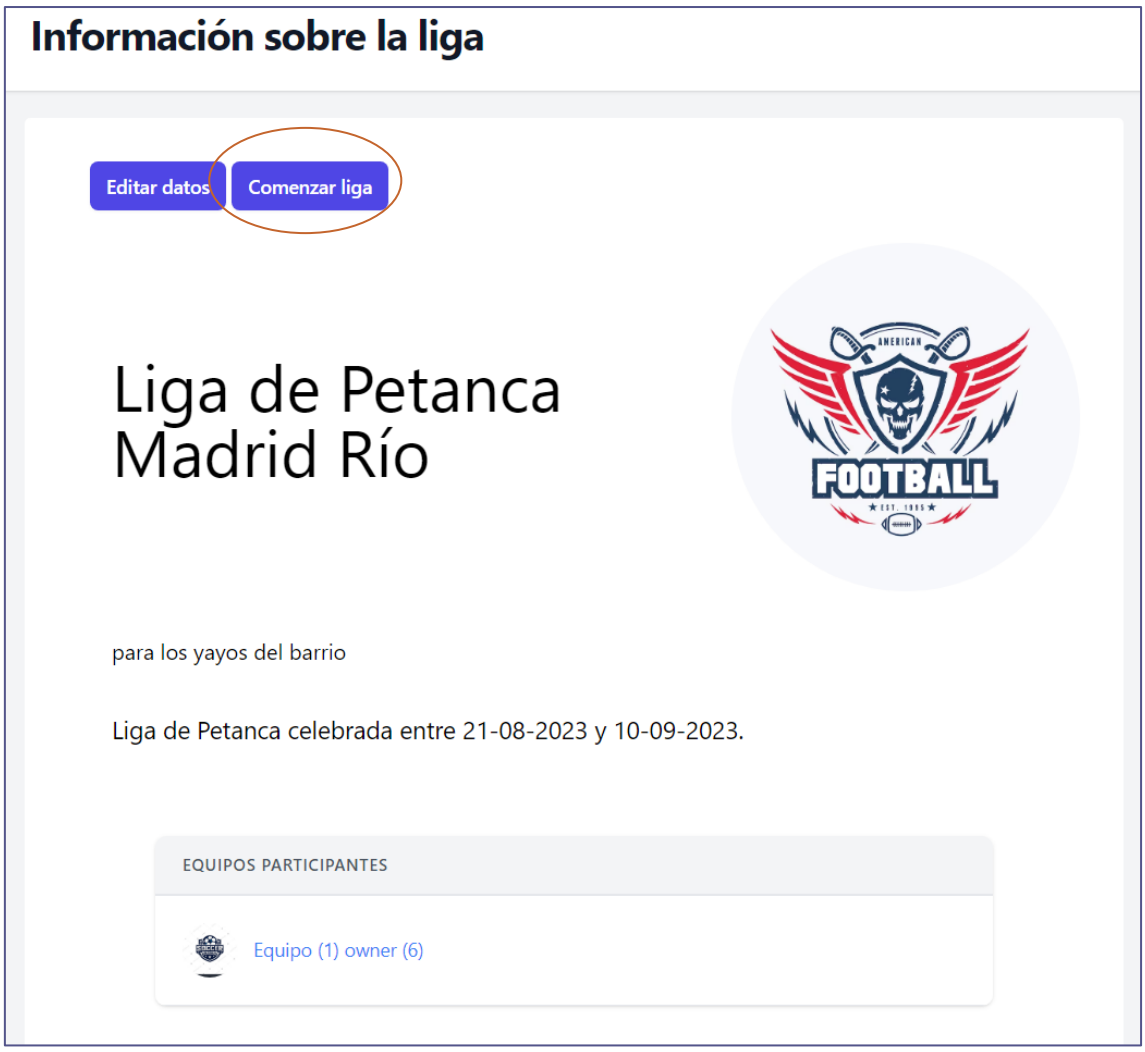


Figura 5-15 Aplicación final, comienzo de una liga

La liga da unas indicaciones al administrador para que pueda confirmar el comienzo de la liga.

Figura 5-16 Aplicación final, indicaciones

Se genera la liga, con la lista de equipos participantes, la clasificación, y las jornadas con los enfrentamientos.

A partir de ahora, y cada vez que se produzca un encuentro, el administrador tiene que acudir para insertar los resultados de los partidos. En ese momento, y en función de las reglas del deporte en el que la liga esté configurada, se darán unos puntos predefinidos a ganadores y perdedores.

Listado de equipos participantes

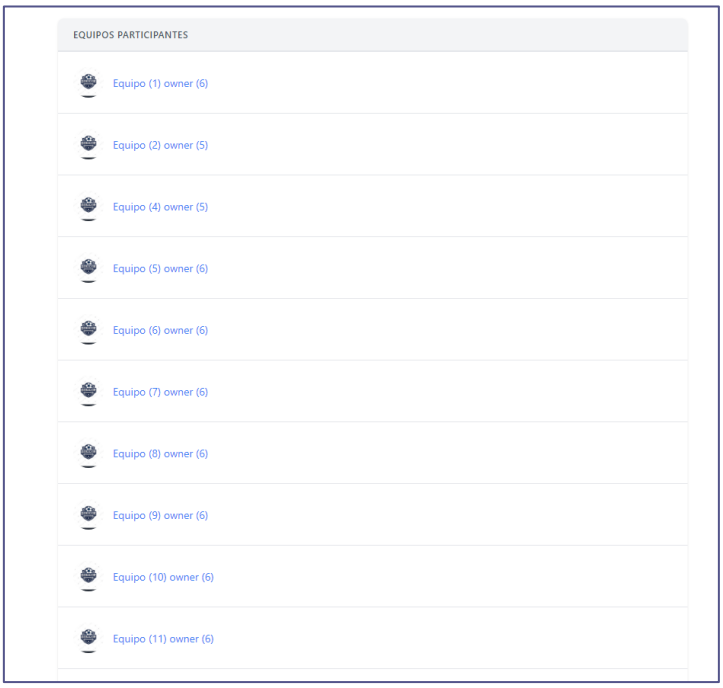


Figura 5-17 Aplicación final, equipos participantes de la liga

Clasificación actual de la liga

EQUIPO	PUNTOS	PJ	PG	PE	PP	GF	GC	DIF
Equipo Equipo (10) owner (6)	3	1	1	0	0	3	0	3
Equipo Equipo (8) owner (6)	3	1	1	0	0	3	1	2
Equipo Equipo (11) owner (6)	3	1	0	0	9	2	1	1
Equipo Equipo (1) owner (6)	1	1	0	1	0	1	1	0
Equipo Equipo (15) owner (6)	1	1	0	1	0	1	1	0
Equipo Equipo (2) owner (5)	1	1	0	1	0	0	0	0
Equipo Equipo (4) owner (5)	1	1	0	1	0	0	0	0
Equipo Equipo (13) owner (6)	1	1	0	1	0	0	0	0

Figura 5-18 Aplicación final, clasificación actual de la liga

CU05: Registro de Resultados

Descripción: Este caso de uso permite a un administrador de partido registrar los resultados de un partido.

Vista de jornadas desplegables, donde se pueden consultar los resultados anteriores y añadir nuevos.

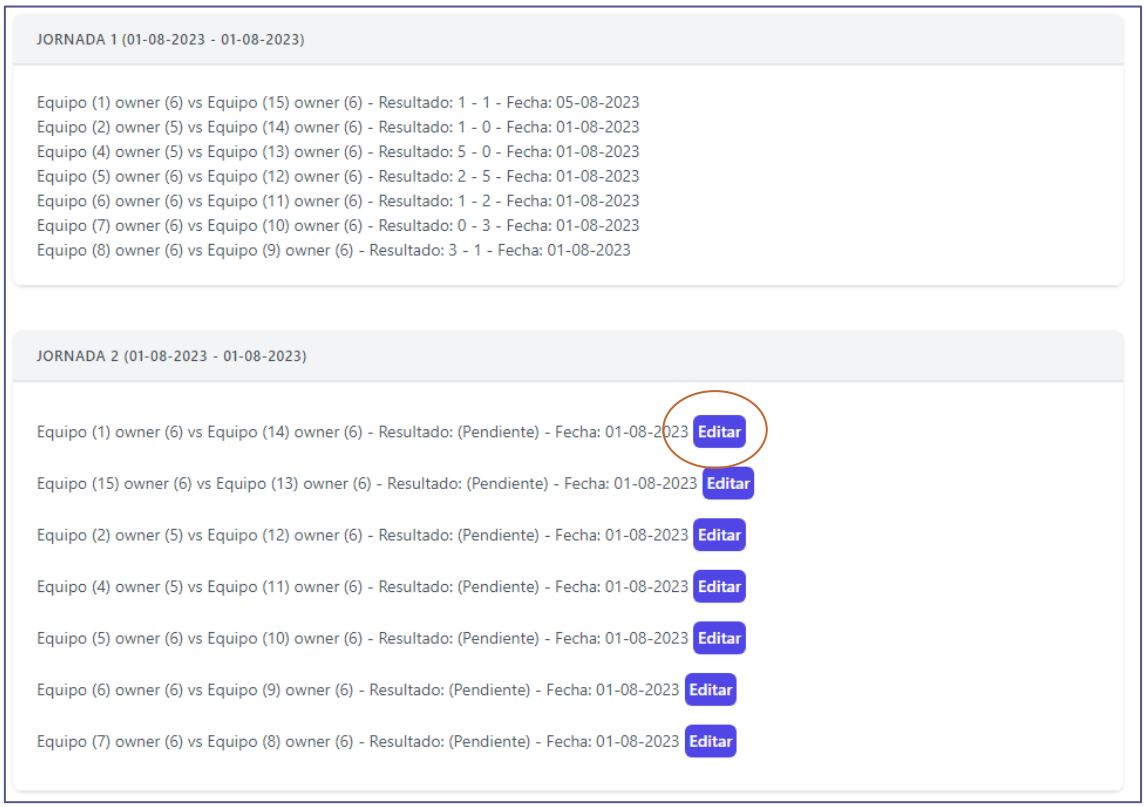


Figura 5-19 Aplicación final, resultados de jornadas

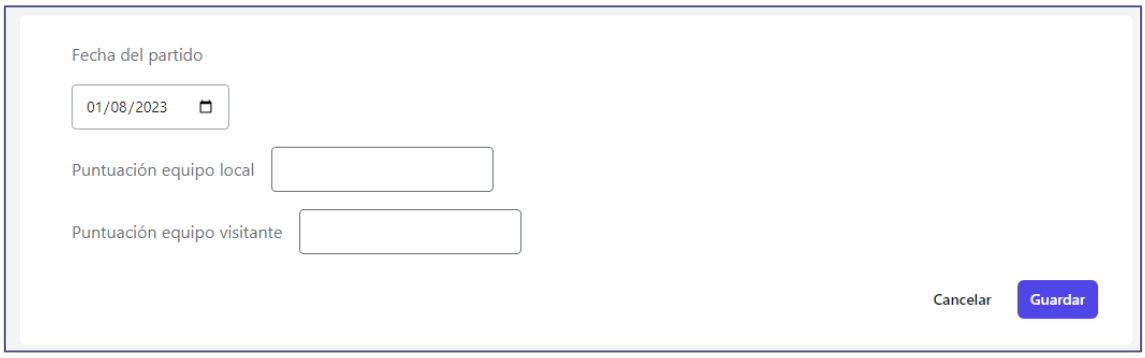


Figura 5-20 Aplicación final, cambiar fechas del partido y agregar resultados

6. Pruebas y despliegue

6.1. VISUALIZACIÓN

Se han ido realizando pruebas de visionado en diferentes navegadores; Google Chrome, Brave, Firefox y Safari.

El diseño del HTML se ha intentado que fuera responsable con la resolución de la pantalla en todo momento. Encontrando problemas en la correcta visualización de ciertas tablas que tienen un tamaño muy grande para el diseño de una pantalla reducida como es el de clasificaciones.

6.2. TIEMPOS DE RESPUESTA

Las pruebas de carga de datos son menores a 100ms en todos los casos. Teniendo en cuenta que la información actual en la base de datos es muy reducida.

Se han completado una liga con 16 equipos y se ha generado una temporada entera de jornadas y partidos de ida y vuelta.

En este caso, la generación en inserción en base de datos ha sido inferior a 2 segundos, siendo este el pico de tiempos de respuesta más alto en toda la navegación

Validación de formularios

Los formularios tienen validación de cantidad de caracteres, e incluso tipo; texto, entero fecha, email, etc. También se ha establecido el valor de **required** en los formularios que requieran de manera imprescindible un valor, imposibilitando continuar con el proceso si la variable no ha sido rellenada.

Solicitudes

Los usuarios deben de realizar una solicitud de acceso a los equipos. Siendo el creador del equipo el encargado de aceptar o denegar la solicitud.

Se han realizado pruebas en ambos casos siendo favorable tanto la aceptación como la denegación de acceso al equipo.

Los responsables de los equipos deben de solicitar acceso a las ligas para sus equipos. Siendo el creador de la liga el encargado de aceptar o denegar la solicitud.

Se han realizado pruebas en ambos casos siendo favorable tanto la aceptación como la denegación de acceso a la liga.

Ejemplo de solicitud y rechazo en la solicitud de acceso a un equipo:

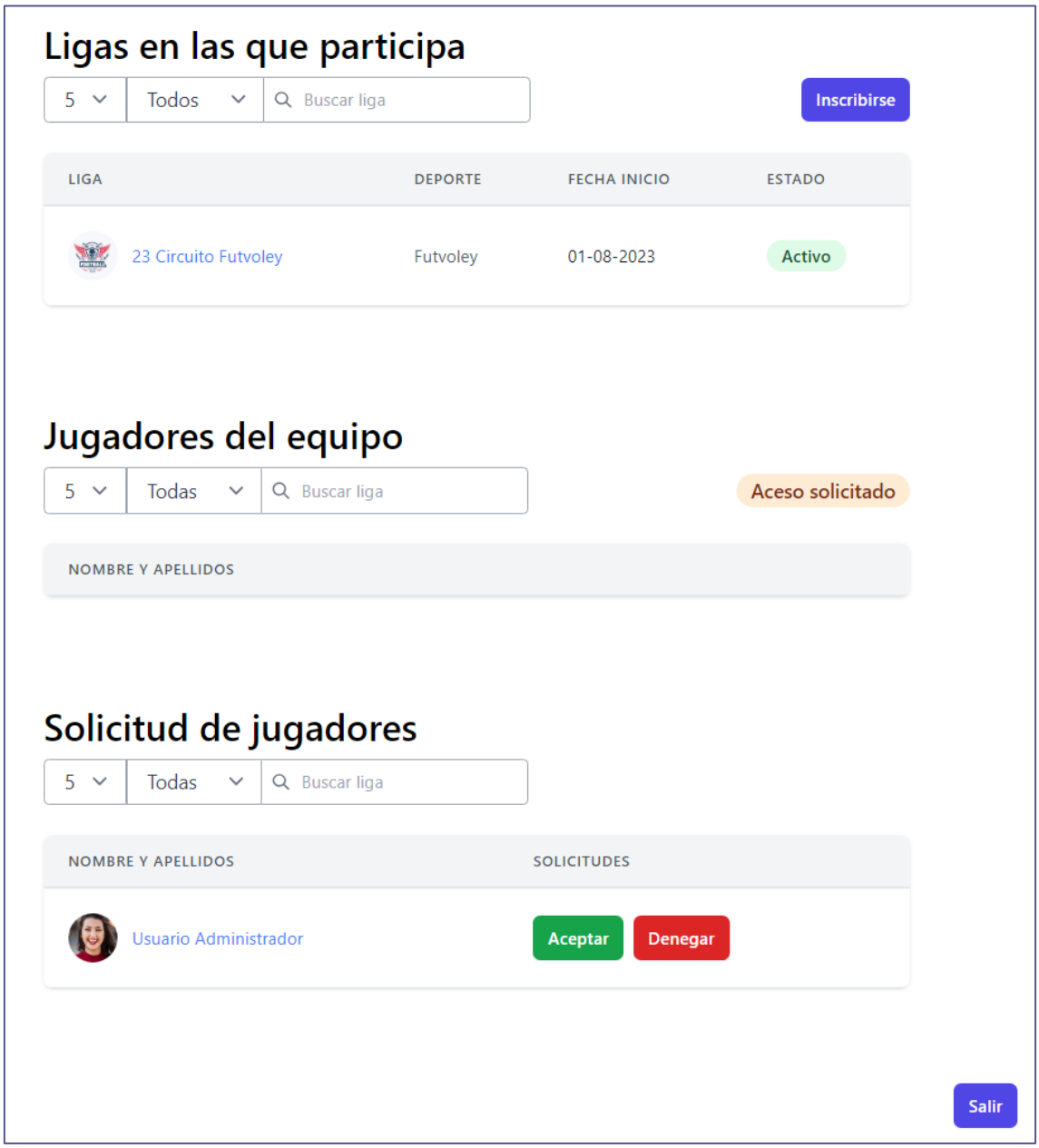


Figura 6-1 Aplicación final, Acceso solicitado

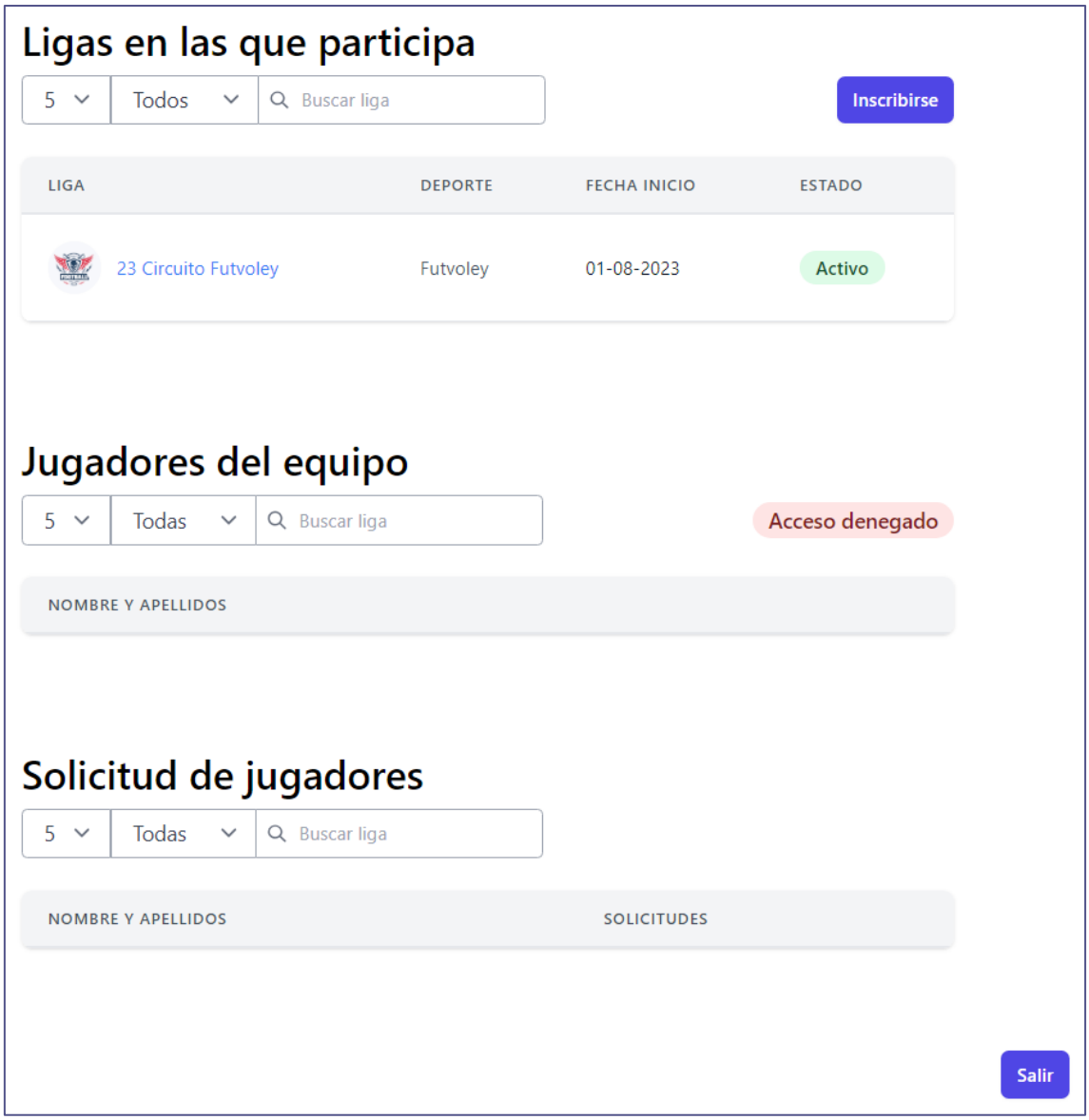


Figura 6-2 Aplicación final, Acceso denegado

7. Conclusiones

Para concluir con la memoria, me gustaría indicar lo mucho que me ha costado mentalmente compatibilizar el desarrollo de la aplicación con el trabajo. Sobre todo, después de haber dejado de lado el TFG durante varios años.

Personalmente, y aunque haya estado trabajando estos años, no he desempeñado las funciones de desarrollador al 100% en ninguna de las empresas en las que he estado. Por lo que he tenido que repasar conceptos de programación. La realización de los cursos ha sido muy importante.

Precisamente por esto, me siento muy realizado, ya que la programación es una parte importante en nuestro sector, y aunque no me haya especializado en ella considero que nunca hay que perder el contacto con el código. Además, siempre es posible reconvertirse y virar el rumbo profesional para enfocarlo a desarrollo.

Me he visto frustrado sobre todo al inicio, cuando me he visto bloqueado por errores en GIT, imposibilitando durante un tiempo el poder programar en equipos diferentes. O cuando tenía que estructurar la lógica en la base de datos. A medida que van saliendo las cosas coges carrerilla y programas más a gusto.

He tenido que poner límites al alcance en varios puntos, ya que una aplicación como esta puede crecer tanto como se desee, un ejemplo de ello es la parte de notificaciones, donde a mitad de diseño la he dejado de lado por la gran posibilidad que se abre con esta funcionalidad.

No trabajar con un *framework* también ha supuesto un reto, ya que respuestas a dudas que encontraba en webs como *StackOverflow* no eran válidas para mi aplicación, y tenía que revisar la documentación de cómo se había construido por dentro algo que un *framework* trae de base, como por ejemplo el *login* y el control de sesiones.

En resumen, el trabajo realizado ha supuesto una experiencia muy regeneradora y del que estoy contento con el resultado. Me gustaría intentar ampliar al menos los apartados que se exponen en el punto siguiente, ya que sería una aplicación más completa y funcional.

Pero sobre todo me quito la espina clavada de poder terminar la carrera de una vez, después de tantos años ya era algo personal.

8. Trabajos futuros

En cuanto a posibles trabajos futuros para ampliar el alcance y funcionalidad de la aplicación se pueden enumerar los siguientes:

- Control de notificaciones. Cuando los usuarios ingresan en equipos, o los equipos en los que participan ingresan en ligas, así como cuando resultados son añadidos, o un partido es cambiado de día.
- Registro más amplio de qué usuario modifica registros de datos. Ya que actualmente sólo queda registrado quién lo crea, quién lo edita, y quien lo borra. Esto ayudaría a posibles trazas en conflictos futuros.
- Aplicación móvil con notificaciones push. Para dar seguimiento de enfrentamientos entre rivales, etc.
- Ampliar la información de los partidos, siendo posible identificar goleadores, amonestaciones, etc.

9. Bibliografía

[1] Chacon, S y Straub, B (2014). *Pro GIT (Second Edition)* Nueva York, Apress Media.

[2] Dev.to, *Make your own service container (php)* https://dev.to/azibom/make-your-own-service-container-php-51oe [Consulta: junio 2023]

[3] Doeken, *How the PHP Middleware Pattern works and can easily be applied*. https://doeken.org/blog/middleware-pattern-in-php [Consulta: junio 2023]

[4] Gauchat, JD, (2012). El gran libro de HTML5, CSS3 y Javascript, Barcelona Marcombo S.A

[5] HTML Goodies, *Web Developer How To: Upload Images Using PHP*. https://www.htmlgoodies.com/php/web-developer-how-to-upload-images-using-php/ [Consulta: agosto 2023]

[6] Matthiasnoback, *Hand-written service containers* https://matthiasnoback.nl/2019/03/hand-written-service-containers/ [Consulta: junio 2023]

[7] Medium, *The Role of Middleware in PHP Frameworks* https://medium.com/legacybeta/the-role-of-middleware-in-php-frameworks-c80b988d4273 [Consulta: junio 2023]

[8] Mmdn Web Docs, *MVC.* https://developer.mozilla.org/es/docs/Glossary/MVC [Consulta: septiembre 2023]

[9] Laracast, *PHP For Beginners*. https://laracasts.com/series/php-for-beginners-2023-edition [Consulta: abril 2023]

[10] Platzi, *Qué es y cómo funciona un middleware en Laravel* https://platzi.com/blog/laravel-middleware/ [Consulta: junio 2023]

[11] PHP.net, *Session Handling.* https://www.php.net/manual/en/book.session.php [Consulta: junio 2023]

[12] PHP Tutorial*, PHP PRG (Post-Redirect-Get)* https://www.phptutorial.net/php-tutorial/php-prg/ [Consulta: junio 2023]

[13] Programacion.NET, *Como interactuar con una base de datos MySQL usando PHP* https://programacion.net/articulo/como_interactuar_con_una_base_de_datos_mysql_usando_php_141 [Consulta: junio 2023]

[14] Siteground, *Tutoriales phpMyAdmin* https://www.siteground.es/tutoriales/phpmyadmin/ [Consulta: mayo 2023]

[15] Tailwind, *Documentation*. https://tailwindui.com/documentation [Consulta: abril 2023]

[16] Vegibit, *Composer Autoloading Tutorial.* https://vegibit.com/composer-autoloading-tutorial/ [Consulta: julio 2023]

[17] Wikipedia, *Modelo–vista–controlador*. https://es.wikipedia.org/wiki/Modelo%E2%80%93vista%E2%80%93controlador [Consulta: septiembre 2023]

[18] W3Schools, *Bootstrap 5 Tutorial*. https://www.w3schools.com/bootstrap5/index.php [Consulta: abril 2023]

[19] W3Schools, *PHP File Upload*. https://www.w3schools.com/php/php_file_upload.asp [Consulta: agosto 2023]

10. Anexos

El objetivo del proyecto no fue pensado para tener alineación directa con ningún ODS planteado en la Agenda2030, sin embargo, el hecho de realizar un proyecto enfocado a facilitar el acceso a plataformas digitales a la gente, de manera global tanto a nivel de diversidad como de localización, edad, género, etc. Y realizarlo en un área de salud y bienestar como es el deporte, hace que, dado su carácter, de manera indirecta se pueda relacionar con varios de ellos.

10.1. OBJETIVOS DE DESARROLLO SOSTENIBLE (ODS)

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenibles	Alto	Medio	Bajo	No Procede
ODS 1. Fin de la pobreza.				X
ODS 2. Hambre cero.				X
ODS 3. Salud y bienestar.		X		
ODS 4. Educación de calidad.			X	
ODS 5. Igualdad de género.			X	
ODS 6. Agua limpia y saneamiento.				X
ODS 7. Energía asequible y no contaminante.				X
ODS 8. Trabajo decente y crecimiento económico.				X
ODS 9. Industria, innovación e infraestructuras.				X
ODS 10. Reducción de las desigualdades.		X		
ODS 11. Ciudades y comunidades sostenibles.		X		
ODS 12. Producción y consumo responsables.				X
ODS 13. Acción por el clima.				X
ODS 14. Vida submarina.				X
ODS 15. Vida de ecosistemas terrestres.				X
ODS 16. Paz, justicia e instituciones sólidas.			X	
ODS 17. Alianzas para lograr objetivos.		X		

ODS 3: Salud y Bienestar

El tercer ODS se centra en garantizar una vida sana y promover el bienestar para todos en todas las edades. La práctica del deporte y la participación en ligas deportivas contribuyen directamente a este objetivo al fomentar un estilo de vida activo y saludable. La aplicación web de gestión de ligas proporciona una plataforma para que las personas se unan, organicen eventos deportivos y participen en actividades que mejoren su salud y bienestar general.

ODS 4: Educación de Calidad

La educación es un componente crucial del desarrollo sostenible. El objetivo 4 se enfoca en garantizar una educación inclusiva, equitativa y de calidad para todos. La gestión de ligas deportivas a menudo implica la organización de eventos deportivos y competiciones. Estos eventos no solo brindan entretenimiento, sino que también pueden educar a las personas sobre el deporte en sí, promover valores como el trabajo en equipo y la ética deportiva, y enseñar habilidades de liderazgo y organización.

3. ODS 5: Igualdad de Género

La igualdad de género es un objetivo fundamental de desarrollo sostenible. Las ligas deportivas pueden jugar un papel importante en la promoción de la igualdad de género al ofrecer oportunidades equitativas para que hombres y mujeres participen y compitan. Una aplicación como esta puede facilitar la inscripción y la participación en ligas deportivas sin importar el género, y puede promover activamente la igualdad de género en el deporte. Esto ha sido evidente a lo largo de este verano, cuando las jugadoras de la selección absoluta de fútbol femenino han resultado campeonas del mundo, y ha sido más sonada la noticia por la polémica con el director de la RAFE que el propio triunfo de las jugadoras.

4. ODS 10: Reducción de las Desigualdades

El deporte puede servir como un punto de unión para personas de diferentes orígenes, edades y habilidades. El proyecto está enfocado para facilitar el uso y el acceso a aplicaciones de gestión a todo el mundo, incentivando el crear comunidades inclusivas al fomentar la participación de diversas personas en actividades deportivas. En el caso de que el proyecto se publique fuera el ámbito académico, se hará con acceso gratuito a todo el mundo.

5. ODS 11: Ciudades y Comunidades Sostenibles

Las ciudades y comunidades sostenibles son una parte esencial del desarrollo sostenible. Las ligas deportivas locales pueden desempeñar un papel en la creación de comunidades más vibrantes y saludables. La gestión eficiente de las ligas y la promoción de eventos deportivos pueden fortalecer los lazos comunitarios y contribuir a un entorno urbano más sostenible.

ODS 16: Paz, Justicia e Instituciones Sólidas
Este objetivo se centra en la promoción de sociedades pacíficas, justas e inclusivas. La gestión de ligas deportivas puede contribuir a este objetivo al fomentar la resolución pacífica de conflictos y promover valores de justicia y ética deportiva. Pudiendo incluir mecanismos para abordar disputas y mantener un entorno deportivo justo y seguro.

ODS 17: Alianzas para lograr los Objetivos
Finalmente, se subraya la importancia de las alianzas y la colaboración global en la consecución de los demás objetivos. La aplicación puede servir como una plataforma para fomentar la colaboración entre organizaciones deportivas, comunidades locales, empresas y otros actores interesados en la promoción del deporte y el desarrollo sostenible.

10.2. CONSULTAS SQL PARA LA CREACIÓN DE TABLAS

Usuario

```
CREATE TABLE IF NOT EXISTS `mysql_tfg`.`usuario` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `nombre` VARCHAR(45) NULL DEFAULT NULL,
  `apellido` VARCHAR(45) NULL DEFAULT NULL,
  `email` VARCHAR(45) NOT NULL,
  `password` VARCHAR(255) NOT NULL,
  `foto` TEXT NULL DEFAULT NULL,
  `numero_sesiones` INT NULL DEFAULT '0',
  `ultima_sesion` DATE NULL DEFAULT NULL,
  `fecha_alta` DATE NULL DEFAULT NULL,
  `fecha_mod` DATE NULL DEFAULT NULL,
  `fecha_baja` DATE NULL DEFAULT NULL,
  `estado` TINYINT NOT NULL DEFAULT '1',
```

```
  `creado_por` INT NULL DEFAULT NULL,
  `modificado_por` INT NULL DEFAULT NULL,
  `borrado_por` INT NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `email_UNIQUE` (`email` ASC) VISIBLE,
  INDEX `usuario_cread` (`creado_por` ASC) VISIBLE,
  INDEX `usuario_mod_idx` (`modificado_por` ASC) VISIBLE,
  INDEX `usuario_del_idx` (`borrado_por` ASC) VISIBLE,
  CONSTRAINT `usuario_creado`
    FOREIGN KEY (`creado_por`)
    REFERENCES `mysql_tfg`.`usuario` (`id`),
  CONSTRAINT `usuario_del`
    FOREIGN KEY (`borrado_por`)
    REFERENCES `mysql_tfg`.`usuario` (`id`),
  CONSTRAINT `usuario_mod`
    FOREIGN KEY (`modificado_por`)
    REFERENCES `mysql_tfg`.`usuario` (`id`))
```

Deporte

```
CREATE TABLE IF NOT EXISTS `mysql_tfg`.`deporte` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `nombre` VARCHAR(255) NOT NULL,
  `logo` TEXT NULL DEFAULT NULL,
  `cover` TEXT NULL DEFAULT NULL,
  `estado` TINYINT NOT NULL DEFAULT '1',
  `creado_por` INT NOT NULL,
  `modificado_por` INT NULL DEFAULT NULL,
  `borrado_por` INT NULL DEFAULT NULL,
```

```
  PRIMARY KEY (`id`),
  UNIQUE INDEX `nombre` (`nombre` ASC) VISIBLE,
  INDEX `creado_por` (`creado_por` ASC) VISIBLE,
  INDEX `modificado_por` (`modificado_por` ASC) VISIBLE,
  INDEX `borrado_por` (`borrado_por` ASC) VISIBLE,
  CONSTRAINT `borrado_por`
    FOREIGN KEY (`borrado_por`)
    REFERENCES `mysql_tfg`.`usuario` (`id`),
  CONSTRAINT `deporte_creado`
    FOREIGN KEY (`creado_por`)
    REFERENCES `mysql_tfg`.`usuario` (`id`),
  CONSTRAINT `deporte_modificaco`
    FOREIGN KEY (`modificado_por`)
    REFERENCES `mysql_tfg`.`usuario` (`id`))
```

Liga

```
CREATE TABLE IF NOT EXISTS `mysql_tfg`.`liga` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `nombre` VARCHAR(255) NOT NULL,
  `descripcion` VARCHAR(255) NOT NULL,
  `fecha_inicio` DATE NOT NULL,
  `fecha_fin` DATE NOT NULL,
  `estado` TINYINT NOT NULL DEFAULT '1',
  `cover` VARCHAR(255) NULL DEFAULT NULL,
  `foto` VARCHAR(255) NULL DEFAULT NULL,
  `deporte_id` INT NOT NULL,
  `creado_por` INT NOT NULL,
  `modificado_por` INT NULL DEFAULT NULL,
```

```
  `borrado_por` INT NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `nombre_UNIQUE` (`nombre` ASC) VISIBLE,
  INDEX `liga_deporte` (`deporte_id` ASC) VISIBLE,
  INDEX `liga_creado_idx` (`creado_por` ASC) VISIBLE,
  INDEX `liga_mod_idx` (`modificado_por` ASC) VISIBLE,
  INDEX `liga_del_idx` (`borrado_por` ASC) VISIBLE,
  CONSTRAINT `liga_del`
    FOREIGN KEY (`borrado_por`)
    REFERENCES `mysql_tfg`.`usuario` (`id`),
  CONSTRAINT `liga_deporte`
    FOREIGN KEY (`deporte_id`)
    REFERENCES `mysql_tfg`.`deporte` (`id`),
  CONSTRAINT `liga_mod`
    FOREIGN KEY (`modificado_por`)
    REFERENCES `mysql_tfg`.`usuario` (`id`))
```

Equipo

```
CREATE TABLE IF NOT EXISTS `mysql_tfg`.`equipo` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `nombre` VARCHAR(255) NOT NULL,
  `cover` TEXT NULL DEFAULT NULL,
  `escudo` TEXT NULL DEFAULT NULL,
  `descripcion` VARCHAR(255) NULL DEFAULT NULL,
  `fecha_creacion` DATE NOT NULL,
  `fecha_mod` DATE NULL DEFAULT NULL,
  `fecha_baja` DATE NULL DEFAULT NULL,
  `estado` TINYINT NOT NULL DEFAULT '1',
```

```
  `creado_por` INT NULL DEFAULT NULL,
  `modificado_por` INT NULL DEFAULT NULL,
  `borrado_por` INT NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `nombre` (`nombre` ASC) VISIBLE,
  INDEX `equipo_creado_idx` (`creado_por` ASC) VISIBLE,
  INDEX `equipo_mod_idx` (`modificado_por` ASC) VISIBLE,
  INDEX `equipo_del_idx` (`borrado_por` ASC) VISIBLE,
  CONSTRAINT `equipo_creado`
    FOREIGN KEY (`creado_por`)
    REFERENCES `mysql_tfg`.`usuario` (`id`),
  CONSTRAINT `equipo_del`
    FOREIGN KEY (`borrado_por`)
    REFERENCES `mysql_tfg`.`usuario` (`id`),
  CONSTRAINT `equipo_mod`
    FOREIGN KEY (`modificado_por`)
    REFERENCES `mysql_tfg`.`usuario` (`id`))
```

Jugador

```
CREATE TABLE IF NOT EXISTS `mysql_tfg`.`jugador` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `equipo_id` INT NOT NULL,
  `usuario_id` INT NOT NULL,
  `fecha_creacion` TIMESTAMP NULL DEFAULT NULL,
  `invitacion` INT NULL DEFAULT NULL,
  `dorsal` INT NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  INDEX `equipo_id` (`equipo_id` ASC) VISIBLE,
```

```
  INDEX `usuario_id` (`usuario_id` ASC) VISIBLE,
  CONSTRAINT `jugadorequipo`
    FOREIGN KEY (`equipo_id`)
    REFERENCES `mysql_tfg`.`equipo` (`id`),
  CONSTRAINT `jugadorusuario`
    FOREIGN KEY (`usuario_id`)
    REFERENCES `mysql_tfg`.`usuario` (`id`))
```

Jornada

```
CREATE TABLE IF NOT EXISTS `mysql_tfg`.`jornada` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `liga_id` INT NOT NULL,
  `numero` INT NOT NULL,
  `fecha_inicio` DATE NULL DEFAULT NULL,
  `fecha_fin` DATE NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  INDEX `jornadaliga_idx` (`liga_id` ASC) VISIBLE,
  CONSTRAINT `jornadaLiga`
    FOREIGN KEY (`liga_id`)
    REFERENCES `mysql_tfg`.`liga` (`id`))

CREATE TABLE IF NOT EXISTS `mysql_tfg`.`partido` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `jornada_id` INT NOT NULL,
  `equipo_local_id` INT NOT NULL,
  `equipo_visitante_id` INT NOT NULL,
  `fecha_hora` DATETIME NULL DEFAULT NULL,
  `resultado_local` INT NULL DEFAULT NULL,
```

```
`resultado_visitante` INT NULL DEFAULT NULL,
`estado` INT NULL DEFAULT NULL,
PRIMARY KEY (`id`),
INDEX `partidoEquipoA_idx` (`equipo_local_id` ASC) VISIBLE,
INDEX `partidoEquipoB_idx` (`equipo_visitante_id` ASC) VISIBLE,
INDEX `partido_jornada_idx` (`jornada_id` ASC) VISIBLE,
CONSTRAINT `partido_jornada`
  FOREIGN KEY (`jornada_id`)
  REFERENCES `mysql_tfg`.`jornada` (`id`),
CONSTRAINT `partidoEquipoA`
  FOREIGN KEY (`equipo_local_id`)
  REFERENCES `mysql_tfg`.`equipo` (`id`),
CONSTRAINT `partidoEquipoB`
  FOREIGN KEY (`equipo_visitante_id`)
  REFERENCES `mysql_tfg`.`equipo` (`id`))
```

Partido

```
DEFAULT CHARACTER SET = utf8mb3CREATE TABLE IF NOT EXISTS `mysql_tfg`.`partido` (
`id` INT NOT NULL AUTO_INCREMENT,
`jornada_id` INT NOT NULL,
`equipo_local_id` INT NOT NULL,
`equipo_visitante_id` INT NOT NULL,
`fecha_hora` DATETIME NULL DEFAULT NULL,
`resultado_local` INT NULL DEFAULT NULL,
`resultado_visitante` INT NULL DEFAULT NULL,
`estado` INT NULL DEFAULT NULL,
PRIMARY KEY (`id`),
```

```
  INDEX `partidoEquipoA_idx` (`equipo_local_id` ASC) VISIBLE,
  INDEX `partidoEquipoB_idx` (`equipo_visitante_id` ASC) VISIBLE,
  INDEX `partido_jornada_idx` (`jornada_id` ASC) VISIBLE,
  CONSTRAINT `partido_jornada`
    FOREIGN KEY (`jornada_id`)
    REFERENCES `mysql_tfg`.`jornada` (`id`),
  CONSTRAINT `partidoEquipoA`
    FOREIGN KEY (`equipo_local_id`)
    REFERENCES `mysql_tfg`.`equipo` (`id`),
  CONSTRAINT `partidoEquipoB`
    FOREIGN KEY (`equipo_visitante_id`)
    REFERENCES `mysql_tfg`.`equipo` (`id`))
```

Equipos_ligas

```
CREATE TABLE IF NOT EXISTS `mysql_tfg`.`equipos_ligas` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `equipo_id` INT NOT NULL,
  `liga_id` INT NOT NULL,
  `clasificacion_id` INT NULL DEFAULT NULL,
  PRIMARY KEY (`id`),
  INDEX `equipo_id` (`equipo_id` ASC) VISIBLE,
  INDEX `liga_id` (`liga_id` ASC) VISIBLE,
  INDEX `clasificacion_id` (`clasificacion_id` ASC) VISIBLE,
  CONSTRAINT `equipos_ligas_pk_equipo`
    FOREIGN KEY (`equipo_id`)
    REFERENCES `mysql_tfg`.`equipo` (`id`),
  CONSTRAINT `equipos_ligas_pk_liga`
    FOREIGN KEY (`liga_id`)
```

```
  REFERENCES `mysql_tfg`.`liga` (`id`))
```

SolicitudesLigas

```
CREATE TABLE IF NOT EXISTS `mysql_tfg`.`solicitudesligas` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `equipo_id` INT NOT NULL,
  `liga_id` INT NOT NULL,
  `estado` INT NOT NULL DEFAULT '1',
  PRIMARY KEY (`id`),
  INDEX `liga_id` (`liga_id` ASC) VISIBLE,
  INDEX `equipo_id` (`equipo_id` ASC) VISIBLE,
  CONSTRAINT `solicitudesligas_ibfk_2`
    FOREIGN KEY (`liga_id`)
    REFERENCES `mysql_tfg`.`liga` (`id`),
  CONSTRAINT `solicitudesligas_ibfk_3`
    FOREIGN KEY (`equipo_id`)
    REFERENCES `mysql_tfg`.`equipo` (`id`))
```

SolicitudesEquipos

```
CREATE TABLE IF NOT EXISTS `mysql_tfg`.`solicitudesequipos` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `usuario_id` INT NOT NULL,
  `equipo_id` INT NOT NULL,
  `estado` INT NOT NULL DEFAULT '1',
  PRIMARY KEY (`id`),
  INDEX `equipo_id` (`equipo_id` ASC) VISIBLE,
  INDEX `solicitudesequipos_ibfk_1` (`usuario_id` ASC) VISIBLE,
```

```
  CONSTRAINT `solicitudesequipos_ibfk_1`
    FOREIGN KEY (`usuario_id`)
    REFERENCES `mysql_tfg`.`usuario` (`id`),
  CONSTRAINT `solicitudesequipos_ibfk_2`
    FOREIGN KEY (`equipo_id`)
    REFERENCES `mysql_tfg`.`equipo` (`id`))
```

Clasificacion

```
CREATE TABLE IF NOT EXISTS `mysql_tfg`.`clasificacion` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `liga_id` INT NOT NULL,
  `equipo_id` INT NOT NULL,
  `puntos` INT NULL DEFAULT '0',
  `pj` INT NULL DEFAULT '0',
  `pg` INT NULL DEFAULT '0',
  `pe` INT NULL DEFAULT '0',
  `pp` INT NULL DEFAULT '0',
  `gf` INT NULL DEFAULT '0',
  `gc` INT NULL DEFAULT '0',
  `dif` INT NULL DEFAULT '0',
  PRIMARY KEY (`id`),
  INDEX `clasiliga_idx` (`liga_id` ASC) VISIBLE,
  INDEX `clasiequipo_idx` (`equipo_id` ASC) VISIBLE,
  CONSTRAINT `clasiequipo`
    FOREIGN KEY (`equipo_id`)
    REFERENCES `mysql_tfg`.`equipo` (`id`),
  CONSTRAINT `clasiliga`
    FOREIGN KEY (`liga_id`)
```

```
REFERENCES `mysql_tfg`.`liga` (`id`))
```