



ugr

Universidad
de Granada

TRABAJO FIN DE GRADO
INGENIERÍA EN INFORMÁTICA

Desarrollo de una aplicación web para la gestión de un equipo de fútbol

Autor

Gabriel Vico Arboledas

Director

Francisco Javier Rodríguez Díaz



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS INFORMÁTICA Y DE
TELECOMUNICACIÓN

Granada, junio de 2025

Desarrollo de una aplicación web para la gestión de un equipo de fútbol

Gabriel Vico Arboledas

Palabras clave: aplicación web, desarrollo web, *frameworks*, *frontend*, *backend*, control de versiones, equipo de fútbol, jugadores, aficionados, gestión de abonos, tienda *online*, calendario de partidos, clasificación, despliegue en la nube

Resumen

En este trabajo se abordarán el desarrollo de una aplicación web para la gestión de un equipo de fútbol. La plataforma web permitirá a los aficionados acceder a información del club, como el calendario de partidos, los resultados y la clasificación. Además, incluirá una tienda *online* para la venta de productos oficiales y un sistema de gestión de abonos de temporada, facilitando la interacción entre el club y sus seguidores.

La web también contará con un apartado de gestión de jugadores, donde se podrá consultar la plantilla del primer equipo masculino, femenino y categorías inferiores. Asimismo, se implementará un abono digital y su posible cesión a otros usuarios, proporcionando un recurso completo para la comunidad del equipo.

Los resultados serán evaluados mediante una serie de pruebas, contribuyendo al desarrollo de la aplicación web.

Development of a web application for the management of a soccer team.

Gabriel Vico Arboledas

Keywords: web application, web development, frameworks, frontend, backend, version control, soccer team, players, fans, season ticket management, online store, match schedule, standings, cloud deployment

Abstract

This work will address the development of a web application for the management of a soccer team. The platform will allow fans to access club information, such as the match schedule, results and standings. It will also include an online store for the sale of official products and a season ticket management system, facilitating interaction between the club and its supporters.

The website will also have a player management section, where the roster of the men's first team, women's first team and youth categories can be consulted. In addition, a digital subscription and its possible transfer to other users will be implemented, providing a complete resource for the team's community.

The results will be evaluated through a series of tests, contributing to the development of the web application.

Yo, **Gabriel Vico Arboledas**, alumno de la titulación INGENIERÍA INFORMÁTICA de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 26520115B, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Gabriel Vico Arboledas

Granada a 3 de junio de 2025 .

D. **Francisco Javier Rodríguez Díaz**, Profesor del Área de Ingeniería Informática del Departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada.

Informa:

Que el presente trabajo, titulado ***Desarrollo de una aplicación web para la gestión de un equipo de fútbol***, ha sido realizado bajo su supervisión por **Gabriel Vico Arboledas**, y autorizo la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a 3 de junio de 2025 .

El director:

Francisco Javier Rodríguez Díaz

Agradecimientos

Quiero expresar mi más sincero agradecimiento a todos aquellos que han participado de manera significativa en la culminación de este Trabajo de Fin de Grado.

En primer lugar, deseo agradecer a mi tutor, Francisco Javier Rodríguez Díaz, por su orientación experta, paciencia y apoyo constante. Su dedicación y profesionalismo han sido cruciales para la finalización de este trabajo. Agradezco especialmente a mi familia, especialmente a mis padres, Manuel y Ángeles, por su amor incondicional, apoyo y fe en mis capacidades. Su respaldo ha sido fundamental durante toda mi formación académica. A mis amigos y compañeros, les agradezco su comprensión, apoyo emocional y los momentos compartidos que hicieron más soportable este proceso.

También agradezco a todas las personas y organizaciones que, de una u otra forma, facilitaron la obtención de los recursos y la información necesarios para este proyecto.

Índice general

| | |
|---|-----------|
| 1. Introducción | 1 |
| 1.1. Motivación | 1 |
| 1.2. Objetivos | 2 |
| 1.2.1. Objetivo Principal | 2 |
| 1.2.2. Objetivos Secundarios | 3 |
| 1.3. Estructura de la memoria | 4 |
| 2. Estado del arte | 7 |
| 2.1. Origen de las páginas webs de clubes deportivos | 7 |
| 2.1.1. Características importantes de una aplicación web para la gestión de un club deportivo | 7 |
| 2.1.2. Ejemplos de aplicaciones web para la gestión de un equipo de fútbol | 8 |
| 2.2. Tipos de desarrollo web | 10 |
| 2.2.1. Desarrollo con CMS (Content Management System) | 10 |
| 2.2.2. Desarrollo con <i>frameworks</i> | 11 |
| 2.2.3. Desarrollo 100 % a medida | 12 |
| 2.3. Conclusiones del estado del arte | 13 |
| 3. Planificación y metodología | 15 |
| 3.1. Etapas | 15 |
| 3.1.1. Etapa inicial | 15 |
| 3.1.2. Etapa de análisis | 16 |
| 3.1.3. Etapa de desarrollo | 17 |
| 3.1.4. Etapa experimental | 17 |
| 3.2. Metodología | 18 |
| 3.3. Planificación | 19 |
| 3.4. Análisis económico del proyecto | 20 |
| 3.4.1. Gastos de personal | 20 |
| 3.4.2. Gastos de material | 20 |
| 3.4.3. Gastos de despliegue y mantenimiento | 21 |
| 3.4.4. Presupuesto total | 21 |

| | |
|---|-----------|
| 4. Análisis | 23 |
| 4.1. Análisis de los usuarios objetivos | 23 |
| 4.1.1. Perfiles de los usuarios | 23 |
| 4.2. Especificación de requisitos | 24 |
| 4.2.1. Requisitos funcionales | 24 |
| 4.2.2. Requisitos no funcionales | 38 |
| 4.2.3. Requisitos de información | 40 |
| 4.2.4. Restricciones del sistema | 42 |
| 4.3. Modelo de casos de uso | 44 |
| 4.3.1. Actores del sistema | 44 |
| 4.3.2. Diagramas de casos de uso | 45 |
| 4.3.3. Descripción de los casos de uso | 49 |
| 4.4. Diagramas de secuencia | 72 |
| 5. Diseño | 75 |
| 5.1. Diseño de la base de datos | 75 |
| 5.1.1. Modelo Entidad-Relación | 75 |
| 5.1.2. Paso a tablas y fusión | 77 |
| 5.1.3. Normalización de la base de datos | 79 |
| 5.2. Diseño de la arquitectura del sistema | 79 |
| 5.2.1. Estructura de MVC | 80 |
| 5.2.2. Ventajas para la aplicación web | 80 |
| 5.3. Diseño de la interfaz de usuario | 81 |
| 5.3.1. Mockups de la aplicación web | 81 |
| 6. Implementación | 87 |
| 6.1. Entorno y lenguajes utilizados | 88 |
| 6.1.1. Lenguajes de programación | 88 |
| 6.1.2. <i>Frameworks</i> y bibliotecas utilizadas | 88 |
| 6.1.3. Base de datos | 89 |
| 6.1.4. Entorno de desarrollo | 89 |
| 6.1.5. Control de versiones y flujo de trabajo | 90 |
| 6.1.6. Gestión de dependencias y herramientas adicionales | 90 |
| 6.2. Implementación de la base de datos | 91 |
| 6.2.1. Creación y estructura de la base de datos | 91 |
| 6.2.2. Conexión con el <i>backend</i> | 92 |
| 6.2.3. Evolución y mantenimiento del esquema | 92 |
| 6.3. Implementación del <i>backend</i> | 92 |
| 6.3.1. Estructura general del proyecto | 92 |
| 6.3.2. <i>Endpoints</i> y funcionalidades de la API | 93 |
| 6.3.3. Validación de datos | 93 |
| 6.3.4. Gestión de errores | 94 |
| 6.3.5. Autenticación y autorización | 94 |
| 6.3.6. Conexión con la base de datos | 94 |

| | |
|---|------------|
| 6.3.7. Servicios externos o integraciones | 94 |
| 6.3.8. Integración mediante Web Scraping para datos depor- tivos | 95 |
| 6.3.9. Otros aspectos técnicos | 95 |
| 6.4. Implementación del <i>frontend</i> | 96 |
| 6.4.1. Estructura del proyecto | 96 |
| 6.4.2. Tecnologías y herramientas | 96 |
| 6.4.3. Comunicación con el <i>backend</i> | 97 |
| 6.4.4. Autenticación en el <i>frontend</i> | 97 |
| 6.4.5. Diseño y experiencia de usuario | 97 |
| 6.4.6. Funcionalidades principales | 97 |
| 6.5. Despliegue | 98 |
| 6.5.1. Despliegue del <i>backend</i> y base de datos | 98 |
| 6.5.2. Despliegue del <i>frontend</i> | 99 |
| 7. Experimentos y resultados | 101 |
| 7.1. Pruebas Unitarias | 101 |
| 7.2. Pruebas Manuales | 102 |
| 7.3. Pruebas de Navegación y Flujo del Usuario | 102 |
| 8. Conclusiones y trabajos futuros | 103 |
| 8.1. Objetivos alcanzados | 103 |
| 8.2. Aprendizaje y reflexión personal | 104 |
| 8.3. Trabajos futuros | 104 |
| Bibliografía | 109 |

Índice de figuras

| | |
|--|----|
| 1.1. Objetivos del proyecto | 4 |
| 2.1. Real Madrid CF (2000) | 8 |
| 2.2. Granada CF (2011) | 9 |
| 2.3. Olympique de Marsella (2025) | 10 |
| 3.1. Idea propuesta | 16 |
| 3.2. Diagrama de Gantt | 20 |
| 4.1. Diagrama de caso de uso - Subsistema de usuarios | 45 |
| 4.2. Diagrama de caso de uso - Subsistema de abonos de temporada | 46 |
| 4.3. Diagrama de caso de uso - Subsistema de tienda del club | 47 |
| 4.4. Diagrama de caso de uso - Subsistema de competiciones | 48 |
| 4.5. Diagrama de caso de uso - Subsistema de jugadores | 49 |
| 4.6. Diagrama de caso de uso - Subsistema de noticias | 50 |
| 4.7. Diagrama de caso de uso - Subsistema de club | 51 |
| 4.8. Diagrama de secuencia - Registro Aficionado | 72 |
| 4.9. Diagrama de secuencia - Eliminar cuenta | 72 |
| 4.10. Diagrama de secuencia - Comprar abono | 73 |
| 4.11. Diagrama de secuencia - Cesión de abono | 73 |
| 4.12. Diagrama de secuencia - Generar abono digital | 73 |
| 4.13. Diagrama de secuencia - Administrar stock | 74 |
| 4.14. Diagrama de secuencia - Predicción de resultado | 74 |
| 5.1. Diagrama Entidad-Relación | 76 |
| 5.2. Modelo-Vista-Controlador (MVC) | 79 |
| 5.3. Mockups de inicio | 82 |
| 5.4. Mockups de plantilla y noticias | 83 |
| 5.5. Mockups de jugador y noticia | 83 |
| 5.6. Mockup de tienda | 84 |
| 5.7. Mockup de producto | 84 |
| 5.8. Mockups de inicio de sesión | 85 |
| 5.9. Mockups de registro | 85 |
| 5.10. Mockup de socio y administrador | 86 |

| | |
|---|----|
| 5.11. Mockup de página estática | 86 |
|---|----|

Capítulo 1

Introducción

El fútbol es uno de los deportes más seguidos a nivel mundial, con millones de seguidores que respaldan a sus equipos tanto en los estadios como en plataformas digitales. En este escenario, los clubes, ya sean profesionales o amateurs, necesitan recursos tecnológicos que les faciliten administrar su actividad de forma eficaz y potenciar su vínculo con los aficionados. La digitalización ha revolucionado la forma en que los equipos se relacionan con su comunidad.

En la actualidad, disponer de una plataforma web adecuadamente estructurada no solo optimiza la gestión interna del equipo, sino que también facilita la difusión de información relevante, tales como calendarios de partidos, resultados y noticias. Asimismo, ofrece un canal directo de interacción entre el club y su comunidad, promoviendo una mayor participación de los seguidores y fortaleciendo el sentido de pertenencia.

En este capítulo, se presentará una descripción detallada de la motivación que dio origen al proyecto. Asimismo, se establecerán de manera clara y precisa los objetivos generales y específicos que se pretenden alcanzar. Finalmente, se expondrá la estructura de la memoria, detallando la organización de los distintos capítulos y secciones que la componen.

1.1. Motivación

En la actualidad, la gestión de clubes deportivos sigue dependiendo en muchos casos de métodos tradicionales que pueden resultar ineficientes. La falta de plataformas modernas y accesibles dificulta la comunicación con los aficionados, la organización interna del club y la comercialización de productos y servicios. Ante esta realidad, surge la necesidad de desarrollar una solución tecnológica que facilite la gestión integral de un equipo de fútbol, mejorando su operatividad y fortaleciendo su vínculo con la afición.

El interés por este proyecto nace de la combinación de dos grandes áreas: el desarrollo web y la pasión por el deporte. A través de la creación de una plataforma digital completa, se busca aplicar conocimientos adquiridos durante la carrera de Ingeniería Informática, desde la arquitectura de software hasta el diseño de interfaces y la implementación de bases de datos. Además, este proyecto representa un desafío técnico interesante, ya que involucra múltiples funcionalidades como la gestión de usuarios, la venta en línea, la visualización de datos en tiempo real y la seguridad en el manejo de la información.

Otro factor motivador es la posibilidad de crear una herramienta con un impacto real y práctico. A menudo, los clubes de fútbol más pequeños o en desarrollo no cuentan con los recursos tecnológicos necesarios para gestionar sus actividades de manera eficiente, lo que limita su crecimiento y capacidad de atraer nuevos socios y patrocinadores. Con este proyecto, se pretende demostrar cómo una solución accesible y bien estructurada puede beneficiar a cualquier equipo, independientemente de su tamaño o presupuesto. Además, este tipo de plataformas pueden servir de modelo para futuras mejoras en la gestión de otras disciplinas deportivas, fomentando la transformación digital en el ámbito deportivo en general.

1.2. Objetivos

A continuación veremos los objetivos que nos planteamos a lo largo del proyecto.

1.2.1. Objetivo Principal

El objetivo principal de proyecto es desarrollar una plataforma web que permita la gestión integral de un club deportivo, proporcionando herramientas para optimizar su administración y mejorar la interacción con los aficionados. La web deberá ser capaz de gestionar los abonos de temporada, una tienda para la venta de productos oficiales, un sistema de seguimiento de competiciones, y una sección dedicada a la plantilla del club.

Con esta plataforma, se busca ofrecer una solución eficiente, accesible y escalable que contribuya a la digitalización del club y a la mejora del mismo.

1.2.2. Objetivos Secundarios

Para garantizar el correcto desarrollo del proyecto, se han establecido una serie de objetivos secundarios que permitirán alcanzar el objetivo principal de manera estructurada y eficiente (véase Figura 1.1):

- Diseñar y desarrollar una arquitectura web eficiente y escalable que permita gestionar todas las funcionalidades del sistema sin afectar el rendimiento. Asegurando una correcta organización de la lógica del servidor (*backend*) y la interfaz gráfica (*frontend*). Además, se implementará un sistema de gestión de usuarios que contempla distintos niveles de acceso, permitiendo diferenciar las funciones de cada usuario. Dado que la plataforma manejará información sensible, se incorporarán medidas de seguridad avanzadas para proteger los datos.
- Diseñar e implementar la gestión del club y sus operaciones. Por lo que se desarrollará un módulo específico para la administración de abonos de temporada. También se implementará una tienda *online* que facilitará la compra de productos oficiales del club. Para mantener informados a los seguidores, se diseñará un apartado que seguimiento de competiciones, proporcionando datos actualizados y accesibles en todo momento.
- Desarrollar una sección dedicada a la administración de las diferentes plantillas del club. Junto con esto, se incorporarán secciones informativas sobre el club, su directiva, y todo lo relacionado con este mismo, con el fin de ofrecer una visión completa de la institución y fortalecer la identidad del equipo dentro de su comunidad.
- Implementar una interfaz intuitiva y accesible, asegurando que la navegación dentro de la web sea sencilla y adaptable a distintos dispositivos, garantizando así una experiencia óptima tanto en ordenadores como en dispositivos móviles.
- Evaluar la experiencia de final de usuario, mediante una serie de pruebas funcionales y de rendimiento con el objetivo de detectar y corregir posibles errores en el sistema. Una vez realizadas las pruebas, la web quedará desplegada en un entorno accesible para los usuarios, asegurando su correcto funcionamiento y disponibilidad para el público objetivo.



Figura 1.1: Objetivos del proyecto

1.3. Estructura de la memoria

En esta sección se describe la organización general del presente trabajo [17], proporcionando una visión global de los capítulos que lo componen y su contenido. Esta estructura se ha diseñado con el objetivo de ofrecer una lectura coherente y lógica, facilitando así la comprensión de los temas abordados.

El **primer capítulo**, titulado *Introducción*, establece las bases del proyecto, como ya se ha ido comprobando al leer. Aquí, se presenta la motivación (sección 1.1), que detalla las razones subyacentes que impulsaron la realización de este trabajo. Se discuten las problemáticas actuales en el campo de estudio y se subraya la relevancia del proyecto. Además, se enuncian los objetivos (sección 1.2), donde se define tanto el objetivo principal como los objetivos secundarios. El objetivo principal representa la meta general y amplia del proyecto, mientras que los objetivos secundarios describen metas más específicas y concretas que contribuyen a alcanzar el objetivo principal.

El **segundo capítulo**, *Estado del arte*, se realiza un análisis de las soluciones existentes en el mercado, revisando plataformas similares y evaluando sus características (sección 2.1). Se estudian las tecnologías utilizadas en sistemas actuales de gestión deportiva, identificando sus ventajas y limitaciones (sección 2.2) con el fin de justificar las decisiones tecnológicas adoptadas en este proyecto (sección 2.3).

El **tercer capítulo**, *Planificación y metodología*, describe las etapas que abarcan desde la concepción del proyecto hasta su implementación final, detallando cada fase del desarrollo (sección 3.1). Asimismo, se explica la metodología de desarrollo de software seleccionada (sección 3.2). Además, incluye un análisis económico detallado para garantizar la viabilidad del proyecto (sección 3.4).

En el **cuarto capítulo**, *Análisis*, los resultados del análisis efectuado, abarcando la especificación de requisitos (sección 4.2), así como los modelos

de caso de uso (sección 4.3) y el comportamiento del sistema (sección 4.4).

El **quinto capítulo**, *Diseño*, se describe la arquitectura y diseño del sistema (sección 5.2), incluyendo el diseño de clases y de la interfaz de usuario (sección 5.3).

El **sexto capítulo**, *Implementación*, detalla el proceso de implementación del sistema (secciones 6.2, 6.3 y 6.4), incluyendo las herramientas utilizadas (sección 6.1) y los pasos del despliegue (sección 6.5).

El **séptimo capítulo**, *Experimentos y resultados*, presenta los experimentos realizados para validar las soluciones propuestas y discute los resultados obtenidos. Finalmente, el **octavo capítulo**, *Conclusiones y trabajos futuros*, sintetiza los hallazgos del proyecto, reflexiona sobre su impacto y sugiere posibles direcciones para investigaciones futuras.

Capítulo 2

Estado del arte

2.1. Origen de las páginas webs de clubes deportivos

Los inicios de las páginas web [3] [14] se remontan a finales de la década de 1980 e inicios de la década 1990, cuando el científico de la computación británico Tim Berners-Lee desarrolló la World Wide Web (WWW), el primer servidor web y la primera página del mundo.

Si nos centramos en el origen de las páginas web de clubes deportivos [5] se remonta a mediados de la década de 1990 en Estados Unidos, donde equipos icónicos de la NBA (como los Chicago Bulls) y la NFL (ejemplo de los Dallas Cowboys) fueron pioneros en explorar el potencial de internet para conectar con sus aficionados.

En España, este suceso llegó unos años más tarde: el Real Madrid CF lanzó su sitio oficial en 1997, marcando un hito en el fútbol europeo, mientras que el FC Barcelona lo hizo entre 1998 y 1999, consolidando una tendencia que transformaría la relación entre los clubes y sus seguidores a nivel global. Estos portales iniciales, aunque simples, sentaron las bases de la digitalización deportiva.

2.1.1. Características importantes de una aplicación web para la gestión de un club deportivo

Las aplicaciones web para la gestión de un club deportivo se caracterizan generalmente por una serie de características distintivas [6] [24]:

- **Gestión deportiva:** se centra en administrar eficientemente las plantillas de los distintos equipos del club, incluyendo el primer equipo masculino, el primer equipo femenino y las categorías inferiores.

- **Gestión de calendario y resultados:** permite a los aficionados y miembros del club acceder fácilmente a información actualizada sobre las diferentes competiciones.
- **Tienda *online*:** permite a los aficionados adquirir productos oficiales. Cuenta con un sistema de gestión de stock en tiempo real.
- **Gestión de abonos de temporada:** este apartado permite a los aficionados adquirir, renovar y gestionar sus abonos de temporada de manera digital.
- **Información institucional:** proporciona un recorrido por la historia del club. Asimismo, ofrece detalles sobre la junta directiva, sus responsabilidades. Además, incluye un apartado de contacto para atender consultas y solicitudes.

2.1.2. Ejemplos de aplicaciones web para la gestión de un equipo de fútbol

Real Madrid CF

La página web del Real Madrid CF, creada en 1997, fue una de las pioneras en el mundo del fútbol, marcando un hito en la digitalización de los clubes deportivos. La web era una plataforma sencilla, reflejo de la era temprana de internet. Con un diseño minimalista, dominado por texto, tablas y colores blanco y azul.

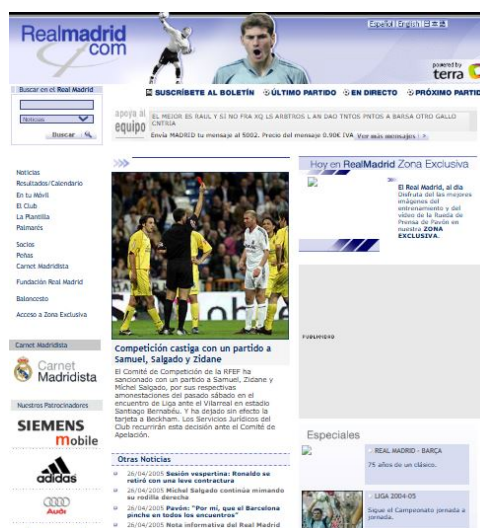


Figura 2.1: Real Madrid CF (2000)

Granada CF

La página web oficial del Granada CF, inaugurada alrededor de 2009, es una plataforma digital integral diseñada para conectar al club con su afición y promover su identidad deportiva. Para apreciar la evolución, se mostrará la página web del Granada CF en el año 2011, donde se observa una estructura más estática y menos interactiva en comparación con la actual. Hoy, tras años de actualizaciones, la página integra contenidos multimedia dinámicos, tienda *online* y gestión de abonos.



Figura 2.2: Granada CF (2011)

Olympique de Marsella

La página web oficial del Olympique de Marsella, es un espacio digital dinámico y moderno que refleja la identidad histórica y la pasión del club francés. Diseñada con un estilo visual impactante, combina los colores emblemáticos del club con contenido multimedia de alta calidad y actualizada en tiempo real.



Figura 2.3: Olympique de Marsella (2025)

2.2. Tipos de desarrollo web

En el mundo del desarrollo web, existen múltiples caminos para construir una aplicación web. A continuación, se explicará las diferentes formas que existen actualmente:

2.2.1. Desarrollo con CMS (Content Management System)

Descripción

Un CMS [4] es una plataforma web diseñada para crear, editar, gestionar y publicar contenido digital de manera organizada y sencilla, sin requerir conocimientos avanzados de programación.

Los CMS se diversifican según su propósito y licencia: por un lado, existen sistemas especializados (como Shopify para e-commerce o WordPress para blogs) y otros multipropósito (como Drupal); por otro, se clasifican por licencia en código abierto (gratuitos y personalizables, como Joomla) o propietarios (de pago y con soporte integrado, como Squarespace).

Ventajas

- No son necesarios conocimientos de programación.
- La velocidad de implementación es muy alta.
- El coste es muy bajo.
- Extensibilidad mediante plugins.

Desventajas

- Difíciles de adaptar a las necesidades específicas del cliente.
- Sólo son seguros si están actualizados, y salen actualizaciones continuamente.
- Difíciles de mantener actualizados cuando contienen multitud de modificaciones.
- Pueden aparecer actualizaciones con cambios que el usuario no quiera implementar.

2.2.2. Desarrollo con *frameworks*

Descripción

Un *framework* Web [12] es un conjunto de herramientas, bibliotecas, patrones y estructuras predefinidas que agilizan y estandarizan el desarrollo de aplicaciones o sitios web.

Funciona como un esqueleto base que proporciona una arquitectura organizada, permitiendo al programador construir toda la funcionalidad necesaria para la aplicación web.

Ventajas

- Permiten la creación de webs totalmente a medida.
- Existen multitud de *frameworks* en diferentes lenguajes (Java, PHP, Javascript, etc).
- Los *frameworks* proporcionan una solución a muchos de los problemas recurrentes en web.
- El mantenimiento es más sencillo porque las nuevas versiones sólo añaden nueva funcionalidad que no rompe la web desarrollada.

Desventajas

- El coste es mayor respecto al desarrollo con CMS.
- La velocidad de implementación es mayor que con CMS.

- Es necesario saber programación y al principio puede resultar difícil de aprender.
- Seguridad. Cuando se publica un exploit para un framework, los atacantes intentan explotarlo, por lo que es necesario parchearlo lo antes posible.

2.2.3. Desarrollo 100 % a medida

Descripción

El desarrollo 100 % a medida (o custom development) [28] es un enfoque en el que se crea software, aplicaciones o sitios web desde cero, diseñados exclusivamente para cumplir con los requisitos específicos de un proyecto. En este proceso, el programador define la arquitectura, las abstracciones y la estructura del código, utilizando directamente lenguajes de programación y librerías seleccionadas.

Ventajas

- Permite un control absoluto, tanto del código como de su arquitectura.
- Seguridad. Nadie externo conoce cómo está el código, por lo que, aunque sigue siendo hackeable, el código permanece como una caja negra para el atacante.

Desventajas

- La forma de mejorar su seguridad es mediante auditorías externas.
- Supone la forma de desarrollo más costosa y larga de implementar de todas.
- Es más difícil de mantener, pues alguien ajeno código debe invertir mucho tiempo y esfuerzo en entender cómo está desarrollado.

2.3. Conclusiones del estado del arte

A través de la revisión de los trabajos existentes en el campo del desarrollo de aplicaciones web para clubes deportivos, se han identificado enfoques variados y sus respectivas ventajas e inconvenientes. Cada uno de estos trabajos aporta elementos valiosos para el desarrollo, pero también deja áreas sin explorar que este TFG pretende abordar.

Gracias a esta investigación profunda se concluye que el uso de *frameworks* web representa la opción más óptima y equilibrada para abordar el proyecto de creación de una aplicación de gestión para el Campillo del Río CF. Este enfoque permite superar las limitaciones de los CMS tradicionales y evita los elevados costes y tiempos asociados al desarrollo 100 % a medida, que resultarían desproporcionados para un club de pequeña escala.

Además, se ha identificado que la ausencia de soluciones estandarizadas asequibles para equipos deportivos amateur justifica la necesidad de este proyecto. El desarrollo con un framework web permite ofrecer una solución accesible, pero profesional, alineada con las particularidades del fútbol base. El resultado será una plataforma que, además de su función práctica, servirá como modelo replicable para otros clubes en contextos similares.

Capítulo 3

Planificación y metodología

En este capítulo se aborda la planificación exhaustiva del proyecto, describiendo las etapas fundamentales y los recursos requeridos para ejecutar la solución planteada.

La programación temporal se visualiza mediante un diagrama de Gantt, que ilustra las etapas y sus plazos de realización. Complementariamente, se especifica un desglose detallado de los costes asociados al proyecto.

Esta planificación detallada de las etapas facilita una administración óptima de los recursos y permite monitorizar los avances. Asimismo, se describe la metodología implementada para la ejecución del proyecto, estableciendo los criterios y procesos que guiarán su desarrollo.

3.1. Etapas

En esta sección, se describen las diversas etapas que componen el proyecto. Estas han sido diseñadas para asegurar una metodología organizada y eficiente en su desarrollo, facilitando una administración óptima de los recursos y la finalización satisfactoria del proyecto final.

3.1.1. Etapa inicial

En esta etapa inicial del proyecto, se establecen los cimientos esenciales para su desarrollo futuro. Durante este periodo, se determinan los objetivos generales y se esbozan los pasos a seguir para la implementación de la solución prevista. Es un punto clave en el que se generan las ideas iniciales y se planifica el resultado final.

Asimismo, en esta primera fase, se realizan reuniones clave con el cliente, en este caso el club Campillo del Río CF, para debatir y definir la idea

del sistema. Estos encuentros son esenciales para comprender a profundidad los requerimientos y expectativas del cliente, además de establecer con claridad el alcance del proyecto. El intercambio constante de información con el cliente permite alinear expectativas y garantizar que el proyecto avance conforme a los objetivos planteados.

Uno de los elementos fundamentales de esta fase es la conceptualización del sistema propuesto. Se elabora un esquema detallado que actúa como referencia visual para el diseño y desarrollo posteriores. Este esquema muestra el funcionamiento más básico de la aplicación web.

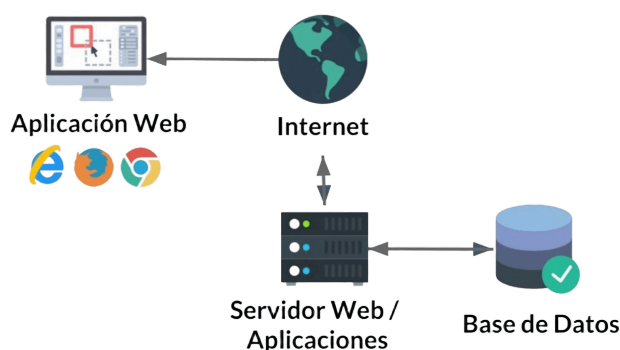


Figura 3.1: Idea propuesta

3.1.2. Etapa de análisis

La etapa de análisis se enfoca en comprender a fondo los requisitos y necesidades del sistema. En esta fase, se recopilan y documentan tanto los requisitos funcionales y no funcionales como la información relevante, además de considerar las restricciones y limitaciones que influirán en el diseño y desarrollo.

Para identificar y priorizar los requisitos del sistema, se realizan entrevistas con el cliente y sesiones de trabajo, lo que permite comprender en detalle el funcionamiento del Campillo del Río CF y detectar oportunidades para optimizar la eficiencia y la automatización.

Sin embargo, esta fase no solo consiste en recolectar información sobre el sistema, sino también en entender el entorno en el que se implementará la solución. Para ello, se llevará a cabo un análisis detallado del tipo usuarios que rodea al Campillo del Río CF, evaluando sus necesidades y expectativas respecto al sistema.

Además, se elaborará una documentación detallada sobre el entorno, con un enfoque específico en el contexto del Campillo del Río CF. Este proceso incluirá la recopilación de información clave sobre los jugadores, las competiciones y otros aspectos relevantes. Como resultado, el análisis proporcionará una visión completa de los desafíos existentes y las áreas que el nuevo sistema deberá mejorar.

3.1.3. Etapa de desarrollo

La siguiente etapa, correspondiente al desarrollo, constituye el núcleo del proyecto, ya que en ella se construye el sistema de información con base en los requisitos y especificaciones definidos previamente. Esta fase se organiza en dos fases fundamentales: el diseño y la implementación.

El proceso comienza con el diseño, donde se define la arquitectura general del sistema, estableciendo sus componentes principales, módulos funcionales y las tecnologías necesarias. Se busca crear una estructura sólida y escalable que permita una integración fluida de los diferentes elementos y que cumpla con los objetivos del proyecto.

Posteriormente, se lleva a cabo el diseño detallado de la base de datos, modelando entidades, relaciones y atributos para garantizar la integridad y coherencia de la información. Se definen las restricciones de integridad y las reglas de negocio que regirán su estructura, asegurando su eficiencia y confiabilidad.

La parte final del diseño se enfocará en la creación de las interfaces de usuario. Se desarrollarán interfaces gráficas y funcionales que facilitarán una interacción intuitiva y eficiente con el sistema. Para ello, se definirán los elementos esenciales, el flujo de navegación y la disposición visual en pantalla, garantizando una experiencia de usuario clara y satisfactoria.

Finalmente, en la fase de desarrollo, se procederá a la implementación del programa, llevando a cabo la codificación del sistema conforme a los diseños y especificaciones previamente establecidos. Durante este proceso, se construirán los distintos componentes, se integrarán los módulos y se realizarán los ajustes necesarios para asegurar el correcto funcionamiento del sistema en su entorno de producción.

3.1.4. Etapa experimental

Esta fase del proyecto es diferente porque no es un paso que comience solo cuando finaliza el anterior, sino que las pruebas y ajustes se realizan de manera iterativa a lo largo de todo el proceso. Cada vez que se incorpora una nueva funcionalidad relevante, se somete a evaluaciones para garantizar su

correcto funcionamiento. Estas pruebas no se limitan a una etapa final, sino que se ejecutan constantemente con el propósito de asegurar la operatividad, confiabilidad y desempeño del sistema en cada momento.

Cada funcionalidad implementada es objeto de rigurosas pruebas funcionales para verificar que cumple con los requisitos establecidos. Se llevan a cabo pruebas específicas, recreando escenarios de uso real para asegurar que todas las funciones operan correctamente. Si se detecta algún error o incidencia, se soluciona de inmediato.

Tras completar todas las pruebas y ajustes necesarios, el sistema se presenta al cliente para su validación final. Este lleva a cabo pruebas de aceptación para confirmar que la solución satisface sus necesidades y expectativas. Se recopilan y documentan sus comentarios y sugerencias con el fin de implementar mejoras futuras y garantizar un producto final que responda plenamente a sus requerimientos.

3.2. Metodología

Para el desarrollo de aplicaciones web, es recomendable emplear metodologías ágiles. Estas metodologías están diseñadas para proyectos en los que los equipos de desarrollo son reducidos, los plazos son cortos, los requisitos pueden cambiar y se utilizan tecnologías innovadoras. Su enfoque está dirigido a soluciones simples y precisas, sin descuidar la calidad del producto. Estas características se ajustan perfectamente a los objetivos del proyecto. Además, esta metodología exige una estrecha colaboración con el cliente, que en este caso serán los tutor responsables del trabajo.

Entre las metodologías ágiles, destaca Scrum, especialmente popular en este tipo de proyectos. Scrum se estructura en “sprints”, que representan fases de trabajo en las que se presenta una versión funcional del producto. Estas iteraciones son flexibles y no lineales, permitiendo que los desarrolladores informen sobre los avances desde la última reunión y establezcan las tareas a realizar hasta la siguiente. Dado que este proyecto involucra numerosas actividades además del desarrollo del software, se incluyen en las iteraciones las primeras reuniones en las que se tratan aspectos clave para la implementación del sistema, las cuales se detallan a continuación.

- **Sprint 1:**
 - Tarea: Selección de un modelo de memoria, análisis del estado del arte, formulación de objetivos, planificación de tareas y diseño de un modelo para la estimación de costos.
- **Sprint 2:**
 - Revisión: Documentación realizada en el sprint anterior.
 - Tarea: Análisis de requisitos.
- **Sprint 3:**
 - Revisión: Análisis de requisitos.
 - Tarea: Análisis de casos de uso y diseño del modelo de base de datos.
- **Sprint 4:**
 - Revisión: Casos de uso y diseño del modelo de base de datos.
 - Tarea: Ampliar casos de uso (plantillas básicas), corregir diseño de base de datos, realizar diagramas de secuencia y comenzar diseño web.
- **Sprint 5:**
 - Revisión: Casos de uso, diagramas de secuencia y diseño de la base de datos, dando por finalizada la fase de análisis. Se presenta un primer boceto del diseño web.
 - Tarea: comenzar desarrollo web con la gestión básica de usuarios y administradores.
- **Sprint 6:**
 - Revisión: Página web con toda la gestión de usuarios completa y siendo totalmente funcional y *Landing page*.
 - Tarea: Finalizar aplicación web.
- **Sprint 7:**
 - Revisión: Aplicación web completa y funcional, asegurando que todos los objetivos han sido alcanzados.
 - Tarea: Finalización de la documentación.

3.3. Planificación

Para ofrecer una visión más precisa sobre la planificación de este proyecto, se ha creado un diagrama de Gantt que refleja el tiempo estimado para cada tarea y el orden en que se ejecutarán. Este gráfico proporciona una representación visual de las actividades programadas y su duración estimada. En él se incluyen todas las etapas, desde la fase inicial de conceptualización hasta la entrega final del sistema. Asimismo, se han determinado las dependencias entre las tareas para asegurar un flujo de trabajo organizado y eficiente.

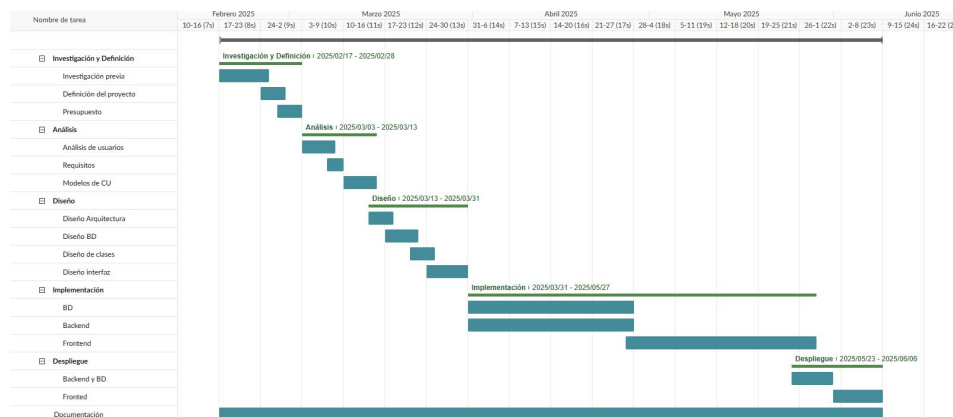


Figura 3.2: Diagrama de Gantt

3.4. Análisis económico del proyecto

En esta sección se detallan los diferentes elementos de gasto que han sido necesarios para llevar a cabo el desarrollo del proyecto, proporcionando un análisis estimado de los costes asociados.

3.4.1. Gastos de personal

El proyecto fue desarrollado íntegramente por una única persona, desempeñando el rol de desarrollador *full-stack* y realizando también las tareas de investigación necesarias. La dedicación fue de aproximadamente 3 meses, con una media diaria de 6 horas de trabajo. Considerando un coste estimado de 17€ por hora, el gasto total en personal para la ejecución del proyecto asciende a:

$$3 \text{ meses} \times 20 \text{ días/mes} \times 6 \text{ horas/día} \times 17 \text{ €/hora} = 6.120 \text{ €}$$

3.4.2. Gastos de material

Para el desarrollo se utilizó un ordenador personal cuyo coste fue de 1.000€. Calculando un uso proporcional de 3 meses, el coste imputable al proyecto es de aproximadamente 62,5€.

No se adquirieron licencias ni suscripciones de software o *frameworks*. Tampoco se contrataron servicios de pago para APIs externas o almacenamiento, ya que se utilizaron planes gratuitos cuando fue posible.

En cuanto a servicios externos con coste asociado, se incluyen los siguientes gastos mensuales:

- Base de datos alojada en Render.com: 7,5 €/mes
- *Backend* alojado en Render.com: 7 €/mes
- *Frontend* alojado en Render.com: 7 €/mes
- Dominio web en hostinger.com: 9,5 €/año

Para 3 meses, estos gastos ascienden a:

$$(7,5 + 7 + 7) \times 3 + \frac{9,5}{4} = 66,88 \text{ €}$$

3.4.3. Gastos de despliegue y mantenimiento

El proyecto está alojado en servicios cloud proporcionados por Render.com, y utiliza Cloudinary en su plan gratuito para el almacenamiento de imágenes.

No existen costes adicionales por licencias ni gastos recurrentes de mantenimiento, ya que no se ha considerado dedicación ni personal específico para estas tareas.

3.4.4. Presupuesto total

Sumando los gastos estimados anteriormente, el presupuesto total aproximado para la realización del proyecto es:

$$6.120 + 62,5 + 66,88 = 6.249,38 \text{ €}$$

Este presupuesto refleja el coste estimado del desarrollo, uso de hardware y servicios externos durante los tres meses de trabajo.

| Concepto | Coste (€) |
|------------------------------|-----------------|
| Gastos de personal | 6.120,00 |
| Gastos de material | 62,50 |
| Servicios externos y dominio | 66,88 |
| Total | 6.249,38 |

Tabla 3.1: Resumen de costes del proyecto

Capítulo 4

Análisis

4.1. Análisis de los usuarios objetivos

En la fase de análisis, es esencial enfocarse en los actores directamente involucrados en el problema final, es decir, los stakeholders o usuarios finales. Este primer paso dentro del análisis es clave para el diseño y desarrollo de cualquier sistema de información. En el caso de este proyecto, los principales usuarios son los aficionados (tanto socios como no socios) y los administradores .

Dado que los usuarios del sistema pueden no contar con conocimientos técnicos o especializados en informática, es fundamental crear una interfaz intuitiva y accesible. Esto permitirá que realicen sus tareas de manera sencilla y eficiente sin requerir una capacitación extensa.

4.1.1. Perfiles de los usuarios

| Aficionados Socios | |
|--------------------------|---|
| Representante | Nombre Apellidos |
| Descripción | Socio registrado del club con acceso a beneficios exclusivos. |
| Tipo | Usuario final con membresía activa. |
| Responsabilidades | Acceder a su abono, acceder a contenido exclusivo, entre otras funcionalidades. |
| Implicación | Alta, ya que representan la base de apoyo del club y requieren un servicio eficiente. |

Tabla 4.1: Perfil de usuario: Aficionados Socios

| Aficionados No Socios | |
|--------------------------|--|
| Representante | Nombre Apellidos |
| Descripción | Seguidor del club sin membresía activa. |
| Tipo | Usuario final sin registro formal en el club. |
| Responsabilidades | Consultar noticias, seguir eventos del club, entre otras funcionalidades. |
| Implicación | Media, ya que están interesados en el club, pero no forman parte de la base de socios. |

Tabla 4.2: Perfil de usuario: Aficionados No Socios

| Administradores | |
|--------------------------|--|
| Representante | Nombre Apellidos |
| Descripción | Personal del club encargado de gestionar la plataforma. |
| Tipo | Gestor del sistema con acceso administrativo. |
| Responsabilidades | Publicar noticias, administrar usuarios, gestionar abonos y mantener el sistema actualizado. |
| Implicación | Total, ya que es responsable del correcto funcionamiento de la plataforma. |

Tabla 4.3: Perfil de usuarios: Administradores

4.2. Especificación de requisitos

La especificación de requisitos es una fase fundamental en el desarrollo de cualquier proyecto de software, ya que define de manera clara las necesidades que el sistema debe cumplir. En este contexto, la creación de una página web para el club Campillo del Río CF, requiere una identificación detallada de los distintos requisitos que garanticen el correcto funcionamiento de la plataforma y su alineación con los objetivos del club y sus aficionados.

Este apartado tiene como objetivo establecer una base sólida para el diseño e implementación del sistema, asegurando que todas las funcionalidades estén bien definidas y documentadas.

4.2.1. Requisitos funcionales

A continuación se detallarán los requisitos del sistema en relación con las funcionalidades que debe incorporar y desarrollar, organizándolos en diferentes categorías.

Subsistema de usuarios

| | |
|--------------------|---|
| ID | RF1.1 |
| Nombre | Permitir el registro de aficionados |
| Descripción | El sistema permitirá a los aficionados registrarse proporcionando sus datos personales. |
| Prioridad | Alta |
| Estado | Completado |

| | |
|--------------------|--|
| ID | RF1.2 |
| Nombre | Permitir el registro de administradores |
| Descripción | El sistema permitirá a los administradores registrarse con credenciales verificadas. |
| Prioridad | Alta |
| Estado | Completado |

| | |
|--------------------|---|
| ID | RF1.3 |
| Nombre | Permitir el inicio de sesión de aficionados y administradores |
| Descripción | El sistema permitirá a los aficionados y administradores iniciar sesión con sus credenciales. |
| Prioridad | Alta |
| Estado | Completado |

| | |
|--------------------|---|
| ID | RF1.4 |
| Nombre | Permitir la modificación de aficionados |
| Descripción | El sistema permitirá a los aficionados modificar sus datos en el sistema. |
| Prioridad | Alta |
| Estado | Completado |

| | |
|--------------------|--|
| ID | RF1.5 |
| Nombre | Permitir la modificación de administradores |
| Descripción | El sistema permitirá a los administradores modificar sus datos en el sistema.. |
| Prioridad | Alta |
| Estado | Completado |

| | |
|--------------------|--|
| ID | RF1.6 |
| Nombre | Permitir eliminar la cuenta de aficionado |
| Descripción | El sistema permitirá a los aficionados eliminar su cuenta del sistema. |
| Prioridad | Alta |
| Estado | Completado |

| | |
|--------------------|--|
| ID | RF1.7 |
| Nombre | Permitir eliminar la cuenta de administrador |
| Descripción | El sistema permitirá a los administradores eliminar su cuenta del sistema. |
| Prioridad | Alta |
| Estado | Completado |

Subsistema de Abonos de temporada

| | |
|--------------------|---|
| ID | RF2.1 |
| Nombre | Permitir el registro de socios |
| Descripción | El sistema permitirá a los usuarios registrarse como socios para acceder a beneficios exclusivos. |
| Prioridad | Alta |
| Estado | Completado |

| | |
|--------------------|---|
| ID | RF2.2 |
| Nombre | Mostrar información sobre abonos disponibles |
| Descripción | El sistema mostrará a los usuarios información detallada sobre los distintos abonos de temporada. |
| Prioridad | Media |
| Estado | Completado |

| | |
|--------------------|--|
| ID | RF2.3 |
| Nombre | Gestionar la compra y renovación de abonos |
| Descripción | El sistema permitirá a los usuarios comprar y renovar abonos de temporada. |
| Prioridad | Alta |
| Estado | Completado |

| | |
|--------------------|---|
| ID | RF2.4 |
| Nombre | Generar tarjetas digitales para acceso al estadio |
| Descripción | El sistema generará tarjetas digitales para que los socios puedan acceder al estadio sin necesidad de una tarjeta física. |
| Prioridad | Media |
| Estado | Completado |

| | |
|--------------------|---|
| ID | RF2.5 |
| Nombre | Permitir la cancelación de abono |
| Descripción | El sistema permitirá a los socios cancelar su abono si así lo desean. |
| Prioridad | Media |
| Estado | Completado |

| | |
|--------------------|---|
| ID | RF2.6 |
| Nombre | Consultar historial de compras y renovaciones |
| Descripción | El sistema permitirá a los socios consultar su historial de compras y renovaciones de abonos. |
| Prioridad | Media |
| Estado | Completado |

| | |
|--------------------|---|
| ID | RF2.7 |
| Nombre | Permitir la cesión temporal de abonos |
| Descripción | Los socios podrán ceder temporalmente su abono a otra persona registrada en el sistema. |
| Prioridad | Media |
| Estado | Completado |

Subsistema de Tienda del club

| | |
|--------------------|---|
| ID | RF3.1 |
| Nombre | Mostrar catálogo de productos |
| Descripción | Mostrar un catálogo de productos con imágenes, descripciones y precios. |
| Prioridad | Alta |
| Estado | Completado |

| | |
|--------------------|--|
| ID | RF3.2 |
| Nombre | Compra a través de carrito |
| Descripción | Permitir la compra de productos a través de un carrito de compras. |
| Prioridad | Alta |
| Estado | Completado |

| | |
|--------------------|---|
| ID | RF3.3 |
| Nombre | Gestión de stock y disponibilidad |
| Descripción | Gestionar el stock y disponibilidad de los productos. |
| Prioridad | Media |
| Estado | Completado |

| | |
|--------------------|--|
| ID | RF3.4 |
| Nombre | Aplicar descuentos y promociones |
| Descripción | Aplicar descuentos y promociones especiales para socios. |
| Prioridad | Media |
| Estado | Completado |

| | |
|--------------------|--|
| ID | RF3.5 |
| Nombre | Añadir producto |
| Descripción | Permitir añadir nuevos productos a la tienda del club. |
| Prioridad | Alta |
| Estado | Completado |

| | |
|--------------------|--|
| ID | RF3.6 |
| Nombre | Eliminar producto |
| Descripción | Permitir eliminar productos de la tienda del club. |
| Prioridad | Media |
| Estado | Completado |

| | |
|--------------------|---|
| ID | RF3.7 |
| Nombre | Modificar producto |
| Descripción | Permitir modificar la información de un producto en la tienda del club. |
| Prioridad | Media |
| Estado | Completado |

Subsistema de Competiciones

| | |
|--------------------|---|
| ID | RF4.1 |
| Nombre | Mostrar calendario de partidos |
| Descripción | Mostrar el calendario de partidos con fechas, horarios y rivales. |
| Prioridad | Alta |
| Estado | Completado |

| | |
|--------------------|--|
| ID | RF4.2 |
| Nombre | Actualizar resultados de partidos |
| Descripción | Permitir actualizar los resultados de los partidos tras su finalización. |
| Prioridad | Alta |
| Estado | Completado |

| | |
|--------------------|---|
| ID | RF4.3 |
| Nombre | Mostrar clasificación de la liga |
| Descripción | Mostrar la clasificación de la liga y otros torneos en los que participe el equipo. |
| Prioridad | Media |
| Estado | Completado |

| | |
|--------------------|--|
| ID | RF4.4 |
| Nombre | Sistema de predicción de resultados |
| Descripción | Integrar un sistema de predicción de resultados para los usuarios basado en estadísticas y análisis previos. |
| Prioridad | Baja |
| Estado | Completado |

| | |
|--------------------|--|
| ID | RF4.5 |
| Nombre | Añadir competición |
| Descripción | Permitir añadir nuevas competiciones al sistema para su seguimiento. |
| Prioridad | Alta |
| Estado | Completado |

| | |
|--------------------|---|
| ID | RF4.6 |
| Nombre | Eliminar competición |
| Descripción | Permitir eliminar competiciones del sistema cuando ya no sean necesarias. |
| Prioridad | Media |
| Estado | Completado |

| | |
|--------------------|---|
| ID | RF4.7 |
| Nombre | Modificar competición |
| Descripción | Permitir modificar la información de una competición existente en el sistema. |
| Prioridad | Media |
| Estado | Completado |

Subsistema de Jugadores

| | |
|--------------------|---|
| ID | RF 5.1 |
| Nombre | Mostrar la plantilla completa del equipo |
| Descripción | El sistema mostrará la plantilla completa de los equipos (primer equipo masculino, femenino, alevines). |
| Prioridad | Alta |
| Estado | Completado |

| | |
|--------------------|--|
| ID | RF 5.2 |
| Nombre | Búsqueda y filtrado de jugadores |
| Descripción | El sistema permitirá la búsqueda y filtrado de jugadores por diferentes criterios. |
| Prioridad | Media |
| Estado | Completado |

| | |
|--------------------|--|
| ID | RF 5.3 |
| Nombre | Añadir un jugador |
| Descripción | El sistema permitirá añadir un nuevo jugador a la base de datos. |
| Prioridad | Alta |
| Estado | Completado |

| | |
|--------------------|---|
| ID | RF 5.4 |
| Nombre | Eliminar un jugador |
| Descripción | El sistema permitirá eliminar un jugador de la base de datos. |
| Prioridad | Media |
| Estado | Completado |

| | |
|--------------------|--|
| ID | RF 5.5 |
| Nombre | Modificar un jugador |
| Descripción | El sistema permitirá modificar la información de un jugador existente. |
| Prioridad | Media |
| Estado | Completado |

Subsistema de Noticias

| | |
|--------------------|--|
| ID | RF6.1 |
| Nombre | Publicar noticias |
| Descripción | Permitir a los administradores publicar noticias con texto, imágenes y videos. |
| Prioridad | Alta |
| Estado | Completado |

| | |
|--------------------|--|
| ID | RF6.2 |
| Nombre | Listado de noticias |
| Descripción | Mostrar un listado de noticias ordenadas por fecha de publicación. |
| Prioridad | Media |
| Estado | Completado |

| | |
|--------------------|--|
| ID | RF6.3 |
| Nombre | Vista detallada de noticias |
| Descripción | Mostrar una vista detallada de cada noticia, incluyendo título, contenido, imágenes y vídeos relacionados. |
| Prioridad | Alta |
| Estado | Completado |

| | |
|--------------------|---|
| ID | RF6.4 |
| Nombre | Clasificación de noticias |
| Descripción | Clasificar las noticias en diferentes categorías. |
| Prioridad | Media |
| Estado | Completado |

| | |
|--------------------|---|
| ID | RF6.5 |
| Nombre | Noticias recientes |
| Descripción | Mostrar las noticias más recientes en la página principal de la aplicación. |
| Prioridad | Alta |
| Estado | Completado |

| | |
|--------------------|---|
| ID | RF6.6 |
| Nombre | Filtrado de noticias |
| Descripción | Implementar filtrado de noticias por fecha y categoría. |
| Prioridad | Media |
| Estado | Completado |

| | |
|--------------------|---|
| ID | RF6.7 |
| Nombre | Edición de noticias |
| Descripción | Permitir a los administradores editar noticias ya publicadas. |
| Prioridad | Alta |
| Estado | Completado |

| | |
|--------------------|---|
| ID | RF6.8 |
| Nombre | Eliminación de noticias |
| Descripción | Permitir a los administradores eliminar noticias ya publicadas. |
| Prioridad | Alta |
| Estado | Completado |

Subsistema de Club

| | |
|--------------------|--|
| ID | RF7.1 |
| Nombre | Historia del club |
| Descripción | Mostrar una sección dedicada a la historia del club. |
| Prioridad | Media |
| Estado | Completado |

| | |
|--------------------|--|
| ID | RF7.2 |
| Nombre | Información de instalaciones |
| Descripción | Permitir a los usuarios explorar información detallada sobre las instalaciones del club. |
| Prioridad | Media |
| Estado | Completado |

| | |
|--------------------|---|
| ID | RF7.3 |
| Nombre | Mapa interactivo |
| Descripción | Integrar un mapa interactivo que muestre la ubicación de las instalaciones del club y permita a los usuarios obtener direcciones. |
| Prioridad | Alta |
| Estado | Completado |

| | |
|--------------------|--|
| ID | RF7.4 |
| Nombre | Listado de patrocinadores |
| Descripción | Mostrar un listado de todos los patrocinadores del club, con su logo y nombre. |
| Prioridad | Media |
| Estado | Completado |

| | |
|--------------------|---|
| ID | RF7.5 |
| Nombre | Directiva del club |
| Descripción | Mostrar una sección dedicada a la directiva del club, con perfiles de los miembros clave. |
| Prioridad | Media |
| Estado | Completado |

| | |
|--------------------|---|
| ID | RF7.6 |
| Nombre | Añadir patrocinadores |
| Descripción | Permitir a los administradores añadir a los patrocinadores. |
| Prioridad | Alta |
| Estado | Completado |

| | |
|--------------------|---|
| ID | RF7.7 |
| Nombre | Eliminar patrocinadores |
| Descripción | Permitir a los administradores eliminar a los patrocinadores. |
| Prioridad | Alta |
| Estado | Completado |

| | |
|--------------------|--|
| ID | RF7.8 |
| Nombre | Modificar patrocinadores |
| Descripción | Permitir a los administradores modificar a los patrocinadores. |
| Prioridad | Alta |
| Estado | Completado |

| | |
|--------------------|----------------------------------|
| ID | RF7.9 |
| Nombre | Sección de Contacto |
| Descripción | Mostrar una sección de Contacto. |
| Prioridad | Media |
| Estado | Completado |

4.2.2. Requisitos no funcionales

En este apartado, se detallan los distintos aspectos técnicos, de seguridad, accesibilidad y mantenimiento, garantizando que la plataforma sea eficiente, confiable y fácil de utilizar tanto para los aficionados como para los administradores.

| | |
|--------------------|--|
| ID | RNF1 |
| Nombre | Usabilidad intuitiva |
| Descripción | La interfaz debe ser intuitiva y accesible para usuarios sin conocimientos técnicos. |
| Prioridad | Alta |
| Estado | Completado |

| | |
|--------------------|--|
| ID | RNF2 |
| Nombre | Diseño responsivo |
| Descripción | La aplicación debe contar con un diseño responsivo, adaptándose a dispositivos móviles y tabletas. |
| Prioridad | Media |
| Estado | Completado |

| | |
|--------------------|---|
| ID | RNF3 |
| Nombre | Autenticación segura |
| Descripción | La aplicación debe implementar autenticación segura con cifrado de contraseñas. |
| Prioridad | Alta |
| Estado | Completado |

| | |
|--------------------|--|
| ID | RNF4 |
| Nombre | Protección de datos |
| Descripción | Todos los datos personales deben almacenarse cumpliendo con el Reglamento General de Protección de Datos (GDPR). |
| Prioridad | Alta |
| Estado | Completado |

| | |
|--------------------|--|
| ID | RNF5 |
| Nombre | Arquitectura modular |
| Descripción | El sistema debe estar diseñado con una arquitectura modular para facilitar futuras ampliaciones. |
| Prioridad | Media |
| Estado | Completado |

| | |
|--------------------|---|
| ID | RNF6 |
| Nombre | Copia de seguridad automática |
| Descripción | Se debe realizar una copia de seguridad automática de la base de datos diariamente. |
| Prioridad | Alta |
| Estado | Completado |

| | |
|--------------------|--|
| ID | RNF7 |
| Nombre | Alta disponibilidad |
| Descripción | La plataforma debe estar disponible el 99.9% del tiempo (excluyendo mantenimientos programados). |
| Prioridad | Alta |
| Estado | Completado |

| | |
|--------------------|---|
| ID | RNF8 |
| Nombre | Política de términos y condiciones |
| Descripción | Se debe incluir una política de términos y condiciones accesible para todos los usuarios. |
| Prioridad | Media |
| Estado | Completado |

| | |
|--------------------|---|
| ID | RNF9 |
| Nombre | Escalabilidad |
| Descripción | Este sistema debe ser escalable, permitiendo la adición de nuevos usuarios. |
| Prioridad | Alta |
| Estado | Completado |

4.2.3. Requisitos de información

En este apartado, se detallarán los distintos tipos de información manejados por el sistema, asegurando su correcta estructuración, actualización y disponibilidad.

| | |
|--------------------|---|
| ID | RI1 |
| Nombre | Información sobre Usuarios |
| Descripción | El sistema debe almacenar y gestionar la información de los aficionados (socios y no socios). |
| Contenido | Nombre, apellidos, correo electrónico, fecha de nacimiento y estado de membresía. |

| | |
|--------------------|---|
| ID | RI2 |
| Nombre | Información sobre Abonos |
| Descripción | El sistema debe almacenar y gestionar los datos de los abonos de temporada. |
| Contenido | Tipo de abono, precio, fecha de compra y renovación. |

| | |
|--------------------|---|
| ID | RI3 |
| Nombre | Historial de Compras de Socios |
| Descripción | Se debe registrar un historial de compras y renovaciones de cada socio. |
| Contenido | Fecha de compra, nombre y apellidos. |

| | |
|--------------------|---|
| ID | RI4 |
| Nombre | Catálogo de Productos |
| Descripción | Se debe gestionar un catálogo de productos. |
| Contenido | Nombre, descripción, precio, stock disponible y imágenes. |

| | |
|--------------------|--|
| ID | RI5 |
| Nombre | Pedidos de Usuarios |
| Descripción | El sistema debe almacenar los pedidos realizados por los usuarios. |
| Contenido | Estado del pedido, fecha de compra y método de pago. |

| | |
|--------------------|---|
| ID | RI6 |
| Nombre | Registro de Descuentos y Promociones |
| Descripción | Se debe mantener un registro de descuentos y promociones aplicadas a los productos. |
| Contenido | Código y tiempo de validación. |

| | |
|--------------------|---|
| ID | RI7 |
| Nombre | Información sobre Partidos |
| Descripción | El sistema debe registrar la información de los partidos. |
| Contenido | Fecha, hora, rival, estadio y resultado. |

| | |
|--------------------|---|
| ID | RI8 |
| Nombre | Clasificación de la Liga y Torneos |
| Descripción | Se debe gestionar la clasificación de la liga y torneos en los que participe el club, con actualización automática de puntos. |
| Contenido | Clasificación de liga y torneos. |

| | |
|--------------------|--|
| ID | RI9 |
| Nombre | Repositorio de Noticias |
| Descripción | Se debe gestionar un repositorio de noticias publicadas. |
| Contenido | Título, contenido, fecha de publicación y archivos multimedia. |

4.2.4. Restricciones del sistema

- **RS-1. Alta de administrador:** En el formulario de alta de administrador, se deben realizar las siguientes validaciones:
 - **RS-1.1. Email:** Debe tener un formato válido (por ejemplo, usuario@dominio.com).
 - **RS-1.2. Contraseña:** Debe ser alfanumérica y tener más de 8 caracteres.
 - **RS-1.3. Rol:** El rol del administrador debe ser válido y estar predefinido en el sistema (por ejemplo, “Administrador General” o “Administrador de Contenidos”).
- **RS-2. Alta de usuario (aficionado):** En el formulario de alta de usuario, se deben realizar las siguientes validaciones:
 - **RS-2.1. Email:** Debe tener un formato válido.
 - **RS-2.2. Contraseña:** Debe ser alfanumérica y tener más de 8 caracteres.
 - **RS-2.3. Teléfono:** Debe ser un número de teléfono válido y existir.
 - **RS-2.4. D.N.I.:** Debe tener 8 números y una letra, la cual debe corresponderse con los números según el algoritmo de validación del DNI.
 - **RS-2.5. Fecha de nacimiento:** Debe ser anterior a 12 años y no superar los 110 años respecto a la fecha actual.
- **RS-3. Gestión de abonos:** Para gestionar abonos, se aplican las siguientes restricciones:
 - **RS-3.1. Compra de abono:** El usuario debe estar registrado y no tener un abono activo para poder comprar uno nuevo.
 - **RS-3.2. Renovación de abono:** Solo se permite renovar un abono si está próximo a expirar (por ejemplo, dentro de los últimos 30 días).

- **RS-3.3. Cesión temporal de abono:** El beneficiario de la cesión debe estar registrado en el sistema y no tener un abono activo.
- **RS-4. Gestión de productos en la tienda:** Para gestionar productos en la tienda, se aplican las siguientes restricciones:
 - **RS-4.1. Añadir producto:** El administrador debe proporcionar todos los campos obligatorios (nombre, descripción, precio, stock).
 - **RS-4.2. Modificar producto:** No se puede modificar un producto si está asociado a una compra en proceso.
 - **RS-4.3. Eliminar producto:** No se puede eliminar un producto si hay stock disponible o si está asociado a una compra pendiente.
- **RS-5. Publicación de noticias:** Para publicar noticias, se aplican las siguientes restricciones:
 - **RS-5.1. Título:** El título de la noticia no puede estar vacío y debe tener un máximo de 100 caracteres.
 - **RS-5.2. Contenido:** El contenido de la noticia no puede estar vacío y debe tener un mínimo de 50 caracteres.
 - **RS-5.3. Categoría:** La noticia debe estar asociada a una categoría válida predefinida en el sistema (por ejemplo, “Deportes”, “Eventos”, “Noticias del Club”).
- **RS-6. Gestión de competiciones:** Para gestionar competiciones, se aplican las siguientes restricciones:
 - **RS-6.1. Añadir competición:** El administrador debe proporcionar todos los campos obligatorios (nombre, fecha de inicio, fecha de fin, equipos participantes).
 - **RS-6.2. Modificar competición:** No se puede modificar una competición si ya ha comenzado.
 - **RS-6.3. Eliminar competición:** No se puede eliminar una competición si ya ha comenzado o si tiene partidos asociados.
- **RS-7. Gestión de jugadores:** Para gestionar jugadores, se aplican las siguientes restricciones:
 - **RS-7.1. Añadir jugador:** El administrador debe proporcionar todos los campos obligatorios (nombre, posición, equipo).

- **RS-7.2. Modificar jugador:** No se puede modificar un jugador si está participando en un partido activo.
- **RS-7.3. Eliminar jugador:** No se puede eliminar un jugador si está asociado a un partido o competición.

4.3. Modelo de casos de uso

El Modelo de Casos de Uso es una representación de las principales interacciones entre los usuarios y el sistema. Este modelo permite identificar y describir las funcionalidades clave del sistema, mostrando cómo los diferentes actores interactúan con la plataforma.

4.3.1. Actores del sistema

Aunque este sistema no se caracteriza por la participación de un gran número de actores, es importante detallar aquellos que desempeñan roles específicos dentro de su funcionamiento. Los actores identificados en el sistema son los siguientes:

- **Aficionado No Socio:** Son seguidores del equipo que no cuentan con un abono del club. A pesar de ello, pueden acceder a diversas funcionalidades dentro de la plataforma, como consultar noticias y comunicados oficiales, revisar el calendario de partidos, resultados y clasificaciones, explorar la tienda del club y realizar compras sin acceso a descuentos exclusivos.
- **Aficionado Socio:** Son aquellos usuarios registrados que han adquirido un bono de temporada, lo que les otorga beneficios adicionales dentro del sistema. Además de contar con todas las funcionalidades disponibles para los aficionados no socios, tienen acceso a la gestión de su abono, permitiéndoles comprar, renovar o ceder temporalmente su pase. También disfrutan de descuentos exclusivos en la tienda del club.
- **Administrador:** Es el encargado de la gestión interna del sistema, asegurando que la plataforma funcione correctamente y que la información esté siempre actualizada. Entre sus responsabilidades se encuentran la publicación y gestión de noticias dentro de la aplicación, la administración de los abonos de los socios, la gestión del catálogo de productos de la tienda y el mantenimiento de la información del club. Su rol es clave para garantizar el correcto funcionamiento y la actualización constante del sistema.

Estos actores cumplen funciones diferentes pero interconectadas dentro del sistema, garantizando que las operaciones se lleven a cabo de forma eficiente y sin contratiempos.

4.3.2. Diagramas de casos de uso

Los diagramas UML de casos de uso muestran de forma gráfica los límites del sistema junto con los elementos que lo componen. En este contexto, se representan los actores según el rol que desempeñan (aficionado socio, aficionado no socio, administrador o sistema) y los casos de uso, los cuales describen la secuencia de acciones ejecutadas por el sistema. Para una mejor organización, el sistema se ha estructurado en diferentes subsistemas.

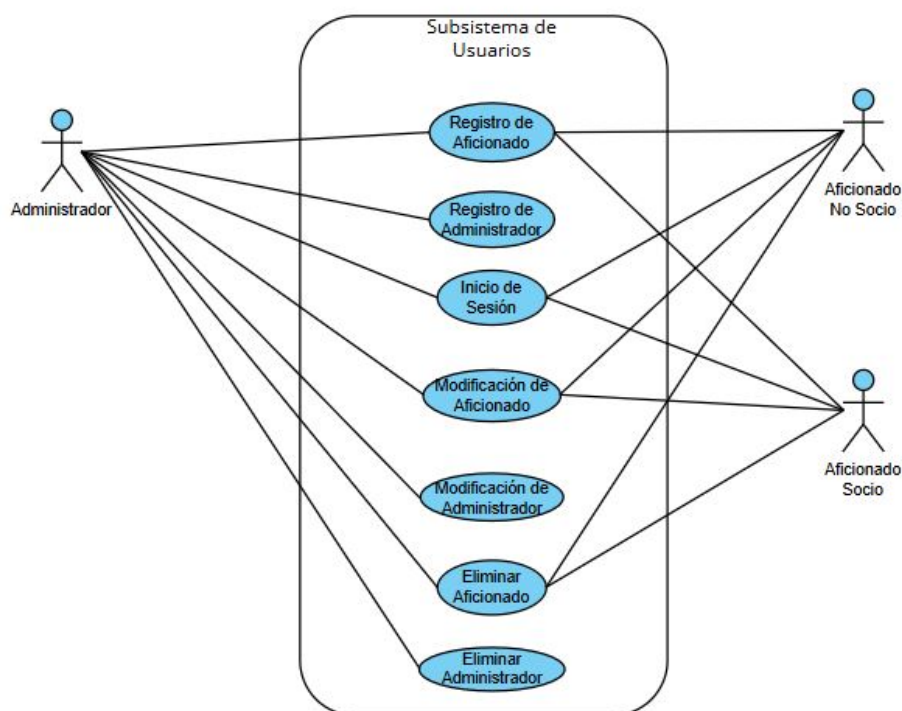


Figura 4.1: Diagrama de caso de uso - Subsistema de usuarios

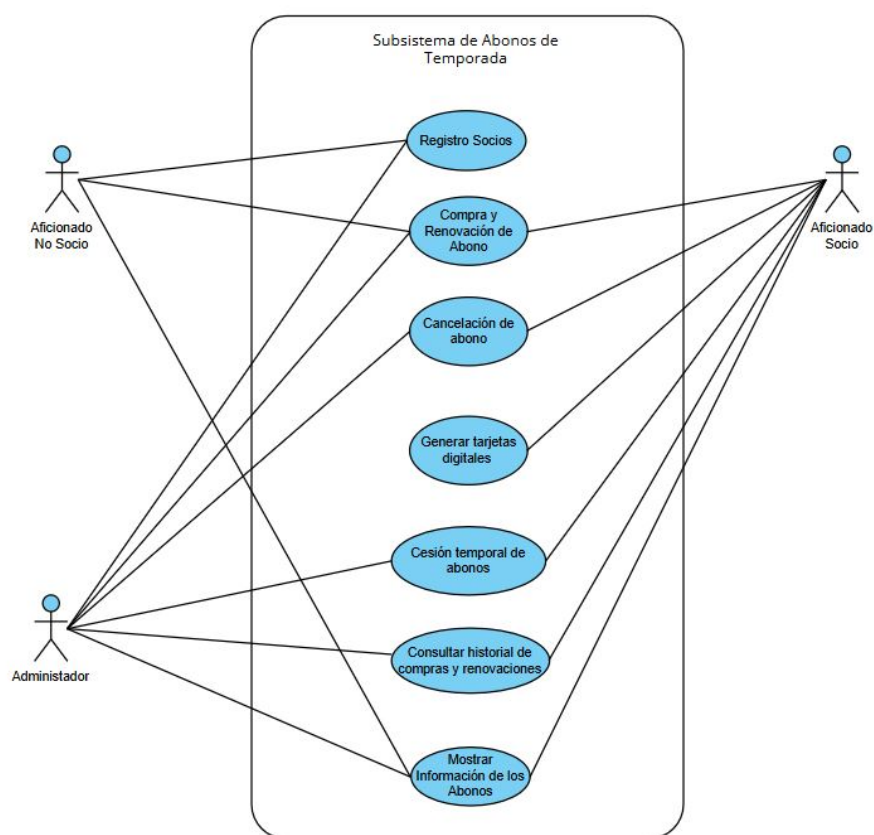


Figura 4.2: Diagrama de caso de uso - Subsistema de abonos de temporada

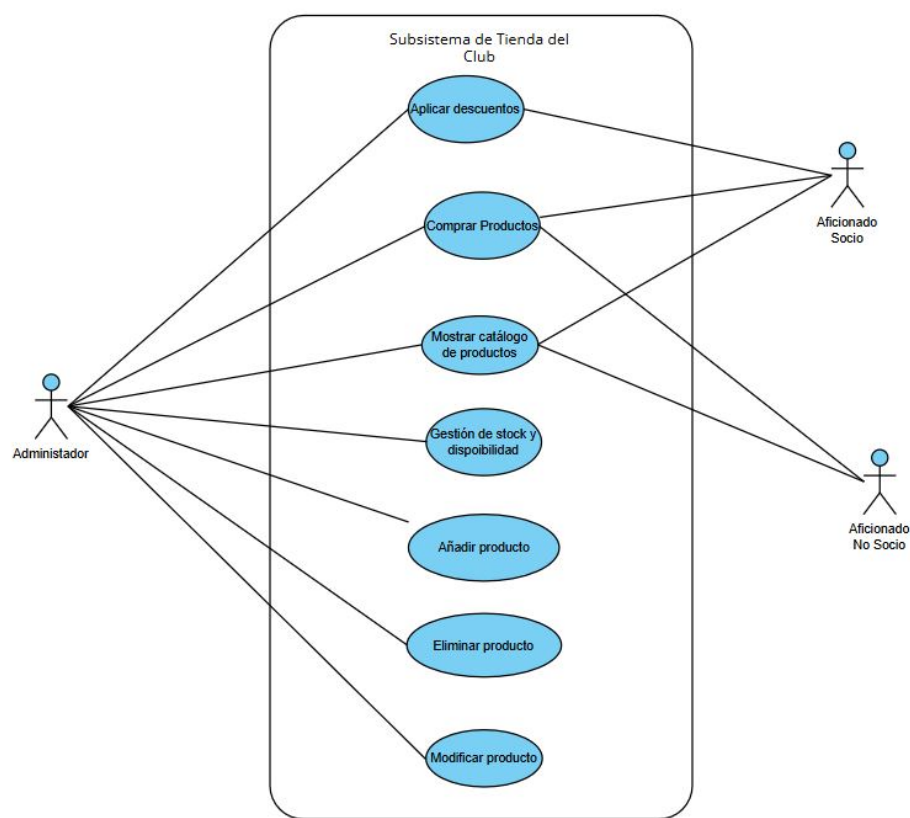


Figura 4.3: Diagrama de caso de uso - Subsistema de tienda del club

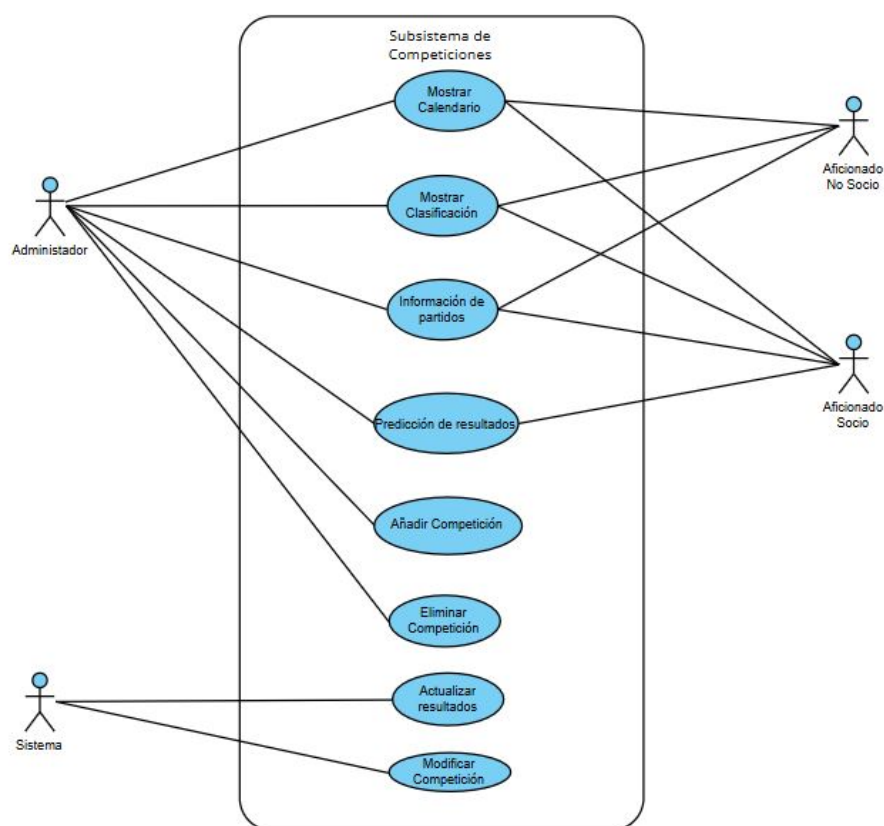


Figura 4.4: Diagrama de caso de uso - Subsistema de competiciones

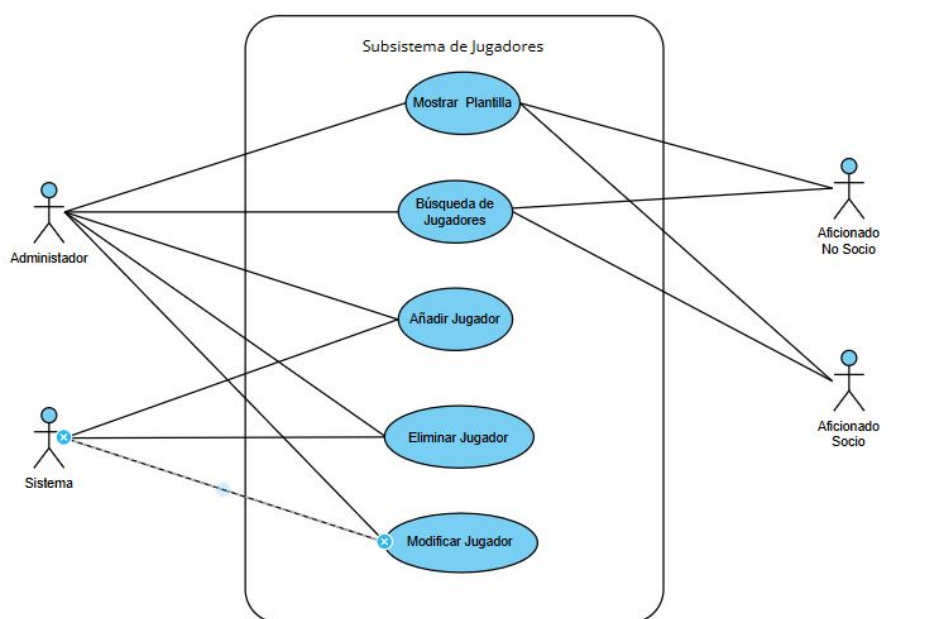
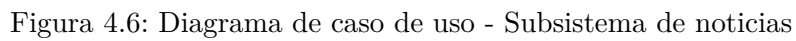


Figura 4.5: Diagrama de caso de uso - Subsistema de jugadores

4.3.3. Descripción de los casos de uso

| | | |
|---|--|------|
| Caso de uso | Permitir el registro de aficionados | CU-1 |
| Actores | Aficionado Socio, Aficionado No Socio y Administrador | |
| Tipo | Primario, básico y esencial | |
| Referencias | RF1.1 | |
| Precondición | El aficionado no debe estar registrado previamente en el sistema. | |
| Poscondición | El aficionado queda registrado en el sistema con sus datos personales. | |
| Propósito | | |
| Permitir a los aficionados registrarse en la plataforma del club para acceder a funcionalidades adicionales. | | |
| Resumen | | |
| El sistema permite a los aficionados proporcionar sus datos personales para registrarse y acceder a las funcionalidades del club. | | |



| | | |
|---|---|------|
| Caso de uso | Permitir el registro de administradores | CU-2 |
| Actores | Administrador | |
| Tipo | Primario, básico y esencial | |
| Referencias | RF1.2 | |
| Precondición | El administrador no debe estar registrado previamente en el sistema. | |
| Poscondición | El administrador queda registrado en el sistema con credenciales verificadas. | |
| Propósito | | |
| Permitir a los administradores registrarse en la plataforma para gestionar el sistema. | | |
| Resumen | | |
| El sistema permite a los administradores registrarse con credenciales verificadas para acceder a las funciones administrativas. | | |

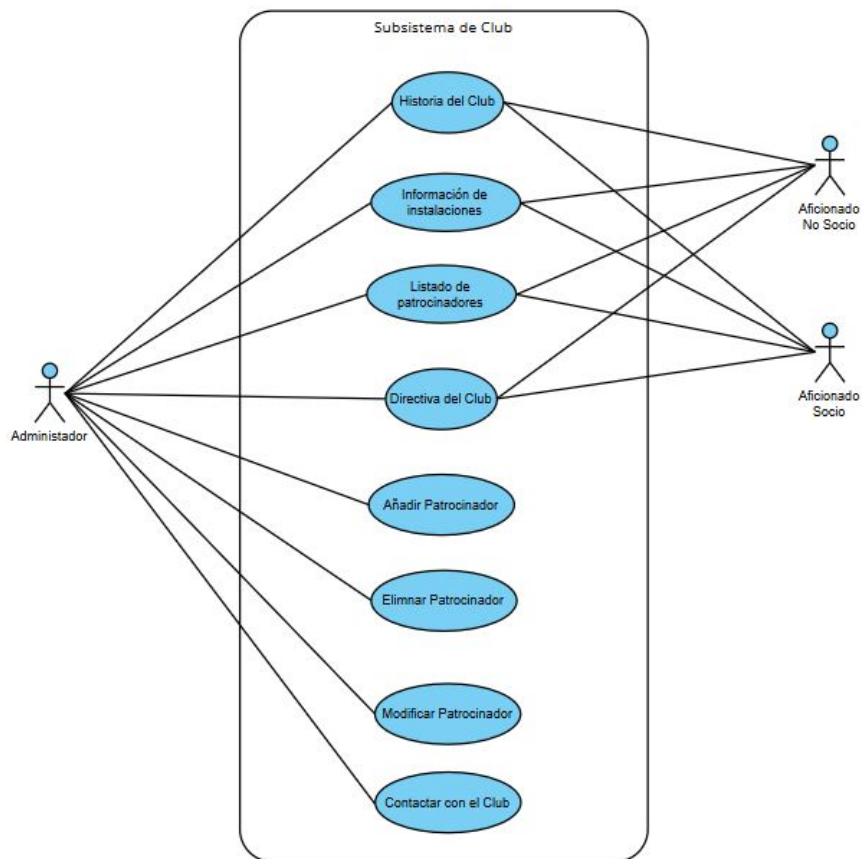


Figura 4.7: Diagrama de caso de uso - Subsistema de club

| | | |
|--|--|------|
| Caso de uso | Permitir el inicio de sesión de aficionados y administradores | CU-3 |
| Actores | Aficionado Socio, Aficionado No Socio y Administrador | |
| Tipo | Primario, básico y esencial | |
| Referencias | RF1.3 | |
| Precondición | El usuario (aficionado o administrador) debe estar registrado en el sistema. | |
| Poscondición | El usuario accede a su cuenta en el sistema. | |
| Propósito | | |
| Permitir a los aficionados y administradores iniciar sesión en la plataforma para acceder a sus funcionalidades correspondientes. | | |
| Resumen | | |
| El sistema permite a los aficionados y administradores iniciar sesión con sus credenciales para acceder a las funcionalidades del sistema. | | |

| | | |
|---|---|------|
| Caso de uso | Permitir la modificación de aficionados | CU-4 |
| Actores | Aficionado Socio, Aficionado No Socio y Administrador | |
| Tipo | Primario, básico y esencial | |
| Referencias | RF1.4 | |
| Precondición | El aficionado debe estar registrado en el sistema. | |
| Poscondición | Los datos del aficionado son modificados en el sistema. | |
| Propósito | | |
| Permitir a los aficionados modificar sus datos personales en el sistema. | | |
| Resumen | | |
| El sistema permite a los aficionados modificar sus datos personales registrados en la plataforma. | | |

| | | |
|---|--|------|
| Caso de uso | Permitir la modificación de administradores | CU-5 |
| Actores | Administrador | |
| Tipo | Primario, básico y esencial | |
| Referencias | RF1.5 | |
| Precondición | El administrador debe estar registrado en el sistema. | |
| Poscondición | Los datos del administrador son modificados en el sistema. | |
| Propósito | | |
| Permitir a los administradores modificar sus datos personales en el sistema. | | |
| Resumen | | |
| El sistema permite a los administradores modificar sus datos personales registrados en la plataforma. | | |

| | | |
|---|---|------|
| Caso de uso | Permitir eliminar la cuenta de aficionado | CU-6 |
| Actores | Aficionado Socio, Aficionado No Socio y Administrador | |
| Tipo | Primario, básico y esencial | |
| Referencias | RF1.6 | |
| Precondición | El aficionado debe estar registrado en el sistema. | |
| Poscondición | La cuenta del aficionado es eliminada del sistema. | |
| Propósito | | |
| Permitir a los aficionados eliminar su cuenta del sistema. | | |
| Resumen | | |
| El sistema permite a los aficionados eliminar su cuenta y todos sus datos asociados de la plataforma. | | |

| | | |
|---|---|------|
| Caso de uso | Permitir eliminar la cuenta de administrador | CU-7 |
| Actores | Administrador | |
| Tipo | Primario, básico y esencial | |
| Referencias | RF1.7 | |
| Precondición | El administrador debe estar registrado en el sistema. | |
| Poscondición | La cuenta del administrador es eliminada del sistema. | |
| Propósito | | |
| Permitir a los administradores eliminar su cuenta del sistema. | | |
| Resumen | | |
| El sistema permite a los administradores eliminar su cuenta y todos sus datos asociados de la plataforma. | | |

| | | |
|---|--|------|
| Caso de uso | Permitir el registro de socios | CU-8 |
| Actores | Aficionado No Socio y Administrador | |
| Tipo | Primario, básico y esencial | |
| Referencias | RF2.1 | |
| Precondición | El aficionado debe estar registrado en el sistema como aficionado no socio. | |
| Poscondición | El aficionado queda registrado como socio y puede acceder a beneficios exclusivos. | |
| Propósito | | |
| Permitir a los aficionados registrarse como socios para acceder a beneficios exclusivos. | | |
| Resumen | | |
| El sistema permite a los aficionados registrarse como socios, lo que les otorga acceso a beneficios exclusivos. | | |

| | | |
|--|--|------|
| Caso de uso | Mostrar información sobre abonos disponibles | CU-9 |
| Actores | Aficionado Socio, Aficionado No Socio y Administrador | |
| Tipo | Primario, básico y esencial | |
| Referencias | RF2.2 | |
| Precondición | El sistema debe tener abonos de temporada disponibles. | |
| Poscondición | El usuario visualiza la información detallada de los abonos disponibles. | |
| Propósito | | |
| Mostrar a los usuarios información detallada sobre los abonos de temporada disponibles. | | |
| Resumen | | |
| El sistema muestra a los usuarios información detallada sobre los distintos abonos de temporada. | | |

| | | |
|--|---|-------|
| Caso de uso | Gestionar la compra y renovación de abonos | CU-10 |
| Actores | Aficionado Socio, Aficionado No Socio y Administrador | |
| Tipo | Primario, básico y esencial | |
| Referencias | RF2.3 | |
| Precondición | El aficionado debe estar registrado como socio. | |
| Poscondición | El aficionado adquiere o renueva un abono de temporada. | |
| Propósito | | |
| Permitir a los socios comprar y renovar abonos de temporada. | | |
| Resumen | | |
| El sistema permite a los socios comprar y renovar abonos de temporada. | | |

| | | |
|---|---|-------|
| Caso de uso | Generar tarjetas digitales para acceso al estadio | CU-11 |
| Actores | Aficionado Socio | |
| Tipo | Primario, básico y esencial | |
| Referencias | RF2.4 | |
| Precondición | El aficionado debe tener un abono de temporada activo. | |
| Poscondición | El aficionado recibe una tarjeta digital para acceder al estadio. | |
| Propósito | | |
| Generar tarjetas digitales para que los socios puedan acceder al estadio sin necesidad de una tarjeta física. | | |
| Resumen | | |
| El sistema genera tarjetas digitales para que los socios puedan acceder al estadio. | | |

| | | |
|--|--|-------|
| Caso de uso | Permitir la cancelación de abono | CU-12 |
| Actores | Aficionado Socio y Administrador | |
| Tipo | Primario, básico y esencial | |
| Referencias | RF2.5 | |
| Precondición | El aficionado debe tener un abono de temporada activo. | |
| Poscondición | El abono del aficionado es cancelado. | |
| Propósito | | |
| Permitir a los socios cancelar su abono de temporada. | | |
| Resumen | | |
| El sistema permite a los socios cancelar su abono de temporada si así lo desean. | | |

| | | |
|---|--|-------|
| Caso de uso | Consultar historial de compras y renovaciones | CU-13 |
| Actores | Aficionado Socio y Administrador | |
| Tipo | Primario, básico y esencial | |
| Referencias | RF2.6 | |
| Precondición | El aficionado debe tener un historial de compras o renovaciones. | |
| Poscondición | El aficionado visualiza su historial de compras y renovaciones. | |
| Propósito | | |
| Permitir a los socios consultar su historial de compras y renovaciones de abonos. | | |
| Resumen | | |
| El sistema permite a los socios consultar su historial de compras y renovaciones de abonos. | | |

| | | |
|---|---|-------|
| Caso de uso | Permitir la cesión temporal de abonos | CU-14 |
| Actores | Aficionado Socio y Administrador | |
| Tipo | Primario, básico y esencial | |
| Referencias | RF2.7 | |
| Precondición | El aficionado debe tener un abono de temporada activo. | |
| Poscondición | El abono es cedido temporalmente a otro usuario registrado. | |
| Propósito | | |
| Permitir a los socios ceder temporalmente su abono a otra persona registrada en el sistema. | | |
| Resumen | | |
| El sistema permite a los socios ceder temporalmente su abono a otro usuario registrado. | | |

| | | |
|--|--|-------|
| Caso de uso | Mostrar catálogo de productos | CU-15 |
| Actores | Aficionado Socio, Aficionado No Socio y Administrador | |
| Tipo | Primario, básico y esencial | |
| Referencias | RF3.1 | |
| Precondición | El sistema debe tener productos disponibles en la tienda. | |
| Poscondición | El usuario visualiza el catálogo de productos con imágenes, descripciones y precios. | |
| Propósito | | |
| Mostrar a los usuarios un catálogo de productos disponibles en la tienda del club. | | |
| Resumen | | |
| El sistema muestra un catálogo de productos con imágenes, descripciones y precios. | | |

| | | |
|---|---|-------|
| Caso de uso | Compra a través de carrito | CU-16 |
| Actores | Aficionado Socio, Aficionado No Socio y Administrador | |
| Tipo | Primario, básico y esencial | |
| Referencias | RF3.2 | |
| Precondición | El usuario debe tener productos en el carrito de compras. | |
| Poscondición | El usuario completa la compra de los productos en el carrito. | |
| Propósito | | |
| Permitir a los usuarios comprar productos a través de un carrito de compras. | | |
| Resumen | | |
| El sistema permite a los usuarios comprar productos utilizando un carrito de compras. | | |

| | | |
|---|---|-------|
| Caso de uso | Gestión de stock y disponibilidad | CU-17 |
| Actores | Administrador | |
| Tipo | Primario, básico y esencial | |
| Referencias | RF3.3 | |
| Precondición | El administrador debe tener acceso al sistema de gestión de la tienda. | |
| Poscondición | El stock y la disponibilidad de los productos son actualizados en el sistema. | |
| Propósito | | |
| Gestionar el stock y la disponibilidad de los productos en la tienda del club. | | |
| Resumen | | |
| El sistema permite a los administradores gestionar el stock y la disponibilidad de los productos. | | |

| | | |
|--|--|-------|
| Caso de uso | Aplicar descuentos y promociones | CU-18 |
| Actores | Aficionado Socio y Administrador | |
| Tipo | Primario, básico y esencial | |
| Referencias | RF3.4 | |
| Precondición | El administrador debe tener acceso al sistema de gestión de la tienda. | |
| Poscondición | Los descuentos y promociones son aplicados a los productos. | |
| Propósito | | |
| Aplicar descuentos y promociones especiales para socios en la tienda del club. | | |
| Resumen | | |
| El sistema permite a los administradores aplicar descuentos y promociones a los productos. | | |

| | | |
|--|--|-------|
| Caso de uso | Añadir producto | CU-19 |
| Actores | Administrador | |
| Tipo | Primario, básico y esencial | |
| Referencias | RF3.5 | |
| Precondición | El administrador debe tener acceso al sistema de gestión de la tienda. | |
| Poscondición | El producto es añadido al catálogo de la tienda. | |
| Propósito | | |
| Permitir a los administradores añadir nuevos productos a la tienda del club. | | |
| Resumen | | |
| El sistema permite a los administradores añadir nuevos productos al catálogo de la tienda. | | |

| | | |
|--|---|-------|
| Caso de uso | Eliminar producto | CU-20 |
| Actores | Administrador | |
| Tipo | Primario, básico y esencial | |
| Referencias | RF3.6 | |
| Precondición | El producto debe existir en el catálogo de la tienda. | |
| Poscondición | El producto es eliminado del catálogo de la tienda. | |
| Propósito | | |
| Permitir a los administradores eliminar productos de la tienda del club. | | |
| Resumen | | |
| El sistema permite a los administradores eliminar productos del catálogo de la tienda. | | |

| | | |
|---|---|-------|
| Caso de uso | Modificar producto | CU-21 |
| Actores | Administrador | |
| Tipo | Primario, básico y esencial | |
| Referencias | RF3.7 | |
| Precondición | El producto debe existir en el catálogo de la tienda. | |
| Poscondición | La información del producto es modificada en el catálogo. | |
| Propósito | | |
| Permitir a los administradores modificar la información de un producto en la tienda del club. | | |
| Resumen | | |
| El sistema permite a los administradores modificar la información de un producto en el catálogo de la tienda. | | |

| | | |
|--|--|-------|
| Caso de uso | Mostrar calendario de partidos | CU-22 |
| Actores | Aficionado Socio, Aficionado No Socio y Administrador | |
| Tipo | Primario, básico y esencial | |
| Referencias | RF4.1 | |
| Precondición | El sistema debe tener partidos programados. | |
| Poscondición | El usuario visualiza el calendario de partidos con fechas, horarios y rivales. | |
| Propósito | | |
| Mostrar a los usuarios el calendario de partidos del equipo. | | |
| Resumen | | |
| El sistema muestra el calendario de partidos con fechas, horarios y rivales. | | |

| | | |
|---|--|-------|
| Caso de uso | Actualizar resultados de partidos | CU-23 |
| Actores | Sistema | |
| Tipo | Primario, básico y esencial | |
| Referencias | RF4.2 | |
| Precondición | El partido debe haber finalizado. | |
| Poscondición | Los resultados del partido son actualizados en el sistema. | |
| Propósito | | |
| Permitir a los administradores actualizar los resultados de los partidos. | | |
| Resumen | | |
| El sistema permite a los administradores actualizar los resultados de los partidos una vez finalizados. | | |

| | | |
|--|--|-------|
| Caso de uso | Mostrar clasificación de la liga | CU-24 |
| Actores | Aficionado Socio, Aficionado No Socio y Administrador | |
| Tipo | Primario, básico y esencial | |
| Referencias | RF4.3 | |
| Precondición | El sistema debe tener datos actualizados de la clasificación de la liga. | |
| Poscondición | El usuario visualiza la clasificación de la liga. | |
| Propósito | | |
| Mostrar a los usuarios la clasificación de la liga y otros torneos en los que participe el equipo. | | |
| Resumen | | |
| El sistema muestra la clasificación de la liga y otros torneos en los que participe el equipo. | | |

| | | |
|--|---|-------|
| Caso de uso | Sistema de predicción de resultados | CU-25 |
| Actores | Aficionado Socio y Administrador | |
| Tipo | Primario, básico y esencial | |
| Referencias | RF4.4 | |
| Precondición | El sistema debe tener datos históricos y estadísticas de partidos. | |
| Poscondición | El usuario visualiza una predicción de resultados basada en estadísticas. | |
| Propósito | | |
| Integrar un sistema de predicción de resultados para los usuarios basado en estadísticas y análisis previos. | | |
| Resumen | | |
| El sistema ofrece una predicción de resultados basada en estadísticas y análisis previos. | | |

| | | |
|--|--|-------|
| Caso de uso | Añadir competición | CU-26 |
| Actores | Administrador | |
| Tipo | Primario, básico y esencial | |
| Referencias | RF4.5 | |
| Precondición | El administrador debe tener acceso al sistema de gestión de competiciones. | |
| Poscondición | La competición es añadida al sistema. | |
| Propósito | | |
| Permitir a los administradores añadir nuevas competiciones al sistema para su seguimiento. | | |
| Resumen | | |
| El sistema permite a los administradores añadir nuevas competiciones al sistema. | | |

| | | |
|---|--|-------|
| Caso de uso | Eliminar competición | CU-27 |
| Actores | Administrador | |
| Tipo | Primario, básico y esencial | |
| Referencias | RF4.6 | |
| Precondición | La competición debe existir en el sistema. | |
| Poscondición | La competición es eliminada del sistema. | |
| Propósito | | |
| Permitir a los administradores eliminar competiciones del sistema cuando ya no sean necesarias. | | |
| Resumen | | |
| El sistema permite a los administradores eliminar competiciones del sistema. | | |

| | | |
|---|---|-------|
| Caso de uso | Modificar competición | CU-28 |
| Actores | Sistema | |
| Tipo | Primario, básico y esencial | |
| Referencias | RF4.7 | |
| Precondición | La competición debe existir en el sistema. | |
| Poscondición | La información de la competición es modificada en el sistema. | |
| Propósito | | |
| Permitir a los administradores modificar la información de una competición existente en el sistema. | | |
| Resumen | | |
| El sistema permite a los administradores modificar la información de una competición existente. | | |

| | | |
|--|--|-------|
| Caso de uso | Mostrar la plantilla completa del equipo | CU-30 |
| Actores | Aficionado Socio, Aficionado No Socio y Administrador | |
| Tipo | Primario, básico y esencial | |
| Referencias | RF5.1 | |
| Precondición | El sistema debe tener registrada la plantilla completa del equipo. | |
| Poscondición | El usuario visualiza la plantilla completa del equipo. | |
| Propósito | | |
| Mostrar a los usuarios la plantilla completa de los equipos (primer equipo masculino, femenino, alevines). | | |
| Resumen | | |
| El sistema muestra la plantilla completa de los equipos, incluyendo el primer equipo masculino, femenino y alevines. | | |

| | | |
|--|---|-------|
| Caso de uso | Búsqueda y filtrado de jugadores | CU-31 |
| Actores | Aficionado Socio, Aficionado No Socio y Administrador | |
| Tipo | Primario, básico y esencial | |
| Referencias | RF5.2 | |
| Precondición | El sistema debe tener jugadores registrados. | |
| Poscondición | El usuario visualiza los resultados de la búsqueda y filtrado de jugadores. | |
| Propósito | | |
| Permitir a los usuarios buscar y filtrar jugadores por diferentes criterios. | | |
| Resumen | | |
| El sistema permite a los usuarios buscar y filtrar jugadores por diferentes criterios, como nombre, posición, equipo, etc. | | |

| | | |
|--|--|-------|
| Caso de uso | Añadir un jugador | CU-32 |
| Actores | Administrador y Sistema | |
| Tipo | Primario, básico y esencial | |
| Referencias | RF5.3 | |
| Precondición | El administrador debe tener acceso al sistema de gestión de jugadores. | |
| Poscondición | El jugador es añadido a la base de datos del sistema. | |
| Propósito | | |
| Permitir a los administradores añadir nuevos jugadores a la base de datos del sistema. | | |
| Resumen | | |
| El sistema permite a los administradores añadir nuevos jugadores a la base de datos. | | |

| | | |
|--|--|-------|
| Caso de uso | Eliminar un jugador | CU-33 |
| Actores | Administrador y Sistema | |
| Tipo | Primario, básico y esencial | |
| Referencias | RF5.4 | |
| Precondición | El jugador debe existir en la base de datos del sistema. | |
| Poscondición | El jugador es eliminado de la base de datos del sistema. | |
| Propósito | | |
| Permitir a los administradores eliminar jugadores de la base de datos del sistema. | | |
| Resumen | | |
| El sistema permite a los administradores eliminar jugadores de la base de datos. | | |

| | | |
|--|---|-------|
| Caso de uso | Modificar un jugador | CU-34 |
| Actores | Administrador y Sistema | |
| Tipo | Primario, básico y esencial | |
| Referencias | RF5.5 | |
| Precondición | El jugador debe existir en la base de datos del sistema. | |
| Poscondición | La información del jugador es modificada en la base de datos. | |
| Propósito | | |
| Permitir a los administradores modificar la información de un jugador existente en la base de datos. | | |
| Resumen | | |
| El sistema permite a los administradores modificar la información de un jugador existente en la base de datos. | | |

| | | |
|--|---|-------|
| Caso de uso | Publicar noticias | CU-35 |
| Actores | Administrador | |
| Tipo | Primario, básico y esencial | |
| Referencias | RF6.1 | |
| Precondición | El administrador debe tener acceso al sistema de gestión de noticias. | |
| Poscondición | La noticia es publicada en el sistema. | |
| Propósito | | |
| Permitir a los administradores publicar noticias con texto, imágenes y vídeos. | | |
| Resumen | | |
| El sistema permite a los administradores publicar noticias en la plataforma. | | |

| | | |
|---|---|-------|
| Caso de uso | Listado de noticias | CU-36 |
| Actores | Aficionado Socio, Aficionado No Socio y Administrador | |
| Tipo | Primario, básico y esencial | |
| Referencias | RF6.2 | |
| Precondición | El sistema debe tener noticias publicadas. | |
| Poscondición | El usuario visualiza un listado de noticias ordenadas por fecha de publicación. | |
| Propósito | | |
| Mostrar a los usuarios un listado de noticias ordenadas por fecha de publicación. | | |
| Resumen | | |
| El sistema muestra un listado de noticias ordenadas por fecha de publicación. | | |

| | | |
|---|---|-------|
| Caso de uso | Clasificación de noticias | CU-37 |
| Actores | Administrador | |
| Tipo | Primario, básico y esencial | |
| Referencias | RF6.4 | |
| Precondición | El administrador debe tener acceso al sistema de gestión de noticias. | |
| Poscondición | Las noticias son clasificadas en diferentes categorías. | |
| Propósito | | |
| Clasificar las noticias en diferentes categorías para facilitar su organización y búsqueda. | | |
| Resumen | | |
| El sistema permite a los administradores clasificar las noticias en diferentes categorías. | | |

| | | |
|---|--|-------|
| Caso de uso | Filtrado de noticias | CU-38 |
| Actores | Aficionado Socio, Aficionado No Socio y Administrador | |
| Tipo | Primario, básico y esencial | |
| Referencias | RF6.6 | |
| Precondición | El sistema debe tener noticias publicadas y categorías definidas. | |
| Poscondición | El usuario visualiza las noticias filtradas por fecha y categoría. | |
| Propósito | | |
| Implementar un sistema de filtrado de noticias por fecha y categoría. | | |
| Resumen | | |
| El sistema permite a los usuarios filtrar noticias por fecha y categoría. | | |

| | | |
|---|--|-------|
| Caso de uso | Edición de noticias | CU-39 |
| Actores | Administrador | |
| Tipo | Primario, básico y esencial | |
| Referencias | RF6.7 | |
| Precondición | La noticia debe estar publicada en el sistema. | |
| Poscondición | La noticia es editada y actualizada en el sistema. | |
| Propósito | | |
| Permitir a los administradores editar noticias ya publicadas. | | |
| Resumen | | |
| El sistema permite a los administradores editar noticias ya publicadas. | | |

| | | |
|---|--|-------|
| Caso de uso | Eliminación de noticias | CU-40 |
| Actores | Administrador | |
| Tipo | Primario, básico y esencial | |
| Referencias | RF6.8 | |
| Precondición | La noticia debe estar publicada en el sistema. | |
| Poscondición | La noticia es eliminada del sistema. | |
| Propósito | | |
| Permitir a los administradores eliminar noticias ya publicadas. | | |
| Resumen | | |
| El sistema permite a los administradores eliminar noticias ya publicadas. | | |

| | | |
|---|---|-------|
| Caso de uso | Listado de patrocinadores | CU-41 |
| Actores | Aficionado Socio, Aficionado No Socio y Administrador | |
| Tipo | Primario, básico y esencial | |
| Referencias | RF7.4 | |
| Precondición | El sistema debe tener un listado de patrocinadores registrados. | |
| Poscondición | El usuario visualiza el listado de patrocinadores. | |
| Propósito | | |
| Mostrar un listado de todos los patrocinadores del club, con su logo y nombre. | | |
| Resumen | | |
| El sistema muestra un listado de todos los patrocinadores del club, con su logo y nombre. | | |

| | | |
|---|---|-------|
| Caso de uso | Añadir patrocinadores | CU-42 |
| Actores | Administrador | |
| Tipo | Primario, básico y esencial | |
| Referencias | RF7.6 | |
| Precondición | El administrador debe tener acceso al sistema de gestión de patrocinadores. | |
| Poscondición | El patrocinador es añadido al listado de patrocinadores del club. | |
| Propósito | | |
| Permitir a los administradores añadir patrocinadores al listado del club. | | |
| Resumen | | |
| El sistema permite a los administradores añadir patrocinadores al listado del club. | | |

| | | |
|--|--|-------|
| Caso de uso | Eliminar patrocinadores | CU-43 |
| Actores | Administrador | |
| Tipo | Primario, básico y esencial | |
| Referencias | RF7.7 | |
| Precondición | El patrocinador debe existir en el listado de patrocinadores del club. | |
| Poscondición | El patrocinador es eliminado del listado de patrocinadores del club. | |
| Propósito | | |
| Permitir a los administradores eliminar patrocinadores del listado del club. | | |
| Resumen | | |
| El sistema permite a los administradores eliminar patrocinadores del listado del club. | | |

4.4. Diagramas de secuencia

Los diagramas de secuencia representan de forma clara la colaboración entre los diferentes objetos del sistema a través del intercambio de mensajes. Su propósito es describir el comportamiento del sistema indicando qué hace, sin enfocarse en cómo lo hace.

Los diagramas presentados a continuación corresponden a la representación gráfica de uno o varios casos de uso, dependiendo de la situación.

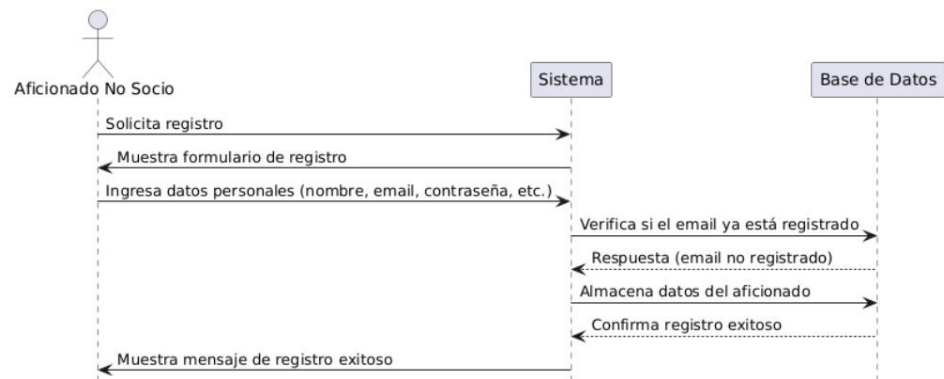


Figura 4.8: Diagrama de secuencia - Registro Aficionado

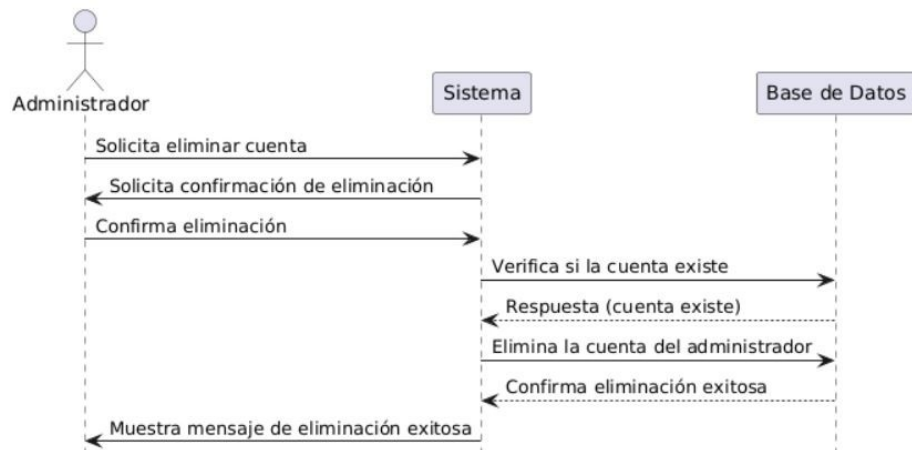


Figura 4.9: Diagrama de secuencia - Eliminar cuenta

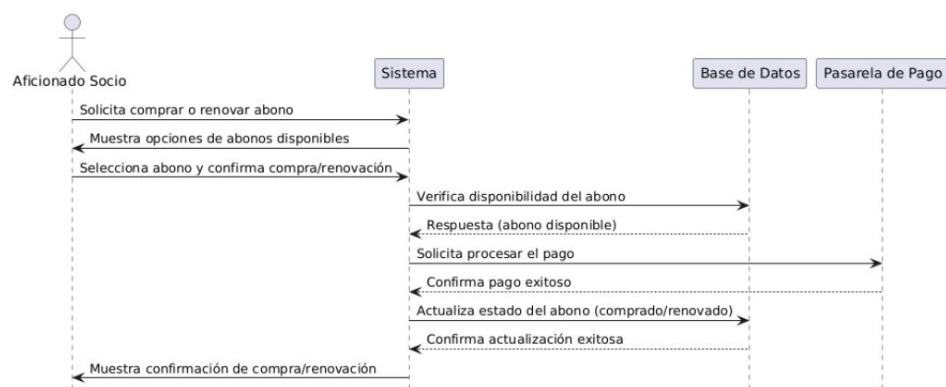


Figura 4.10: Diagrama de secuencia - Comprar abono

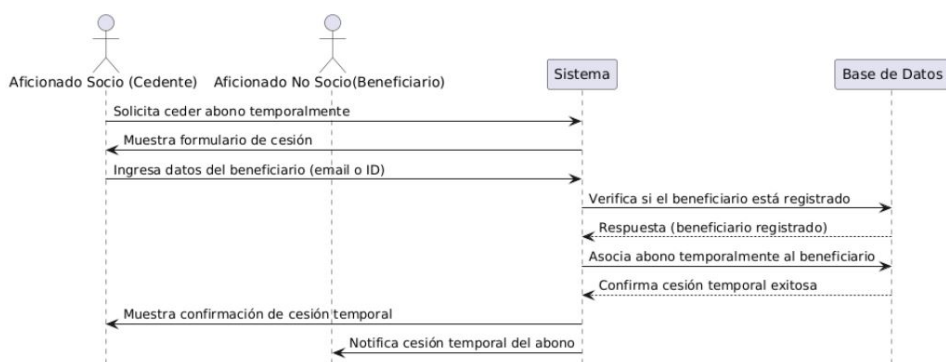


Figura 4.11: Diagrama de secuencia - Cesión de abono

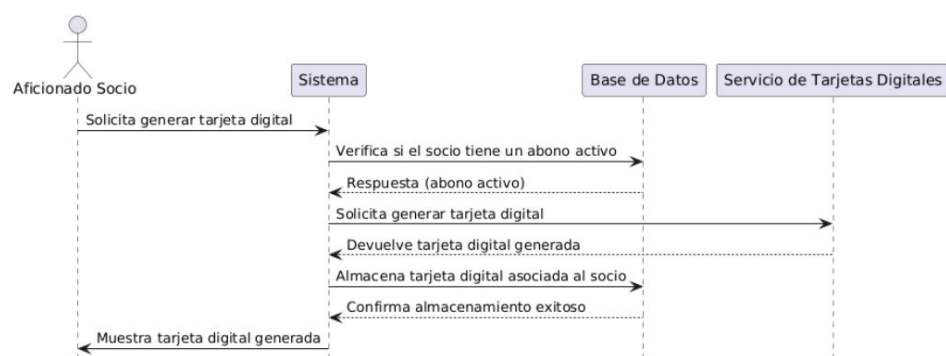


Figura 4.12: Diagrama de secuencia - Generar abono digital

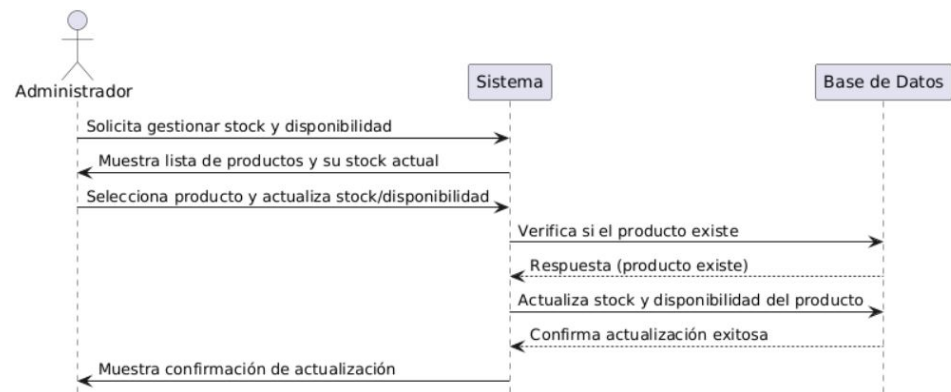


Figura 4.13: Diagrama de secuencia - Administrar stock

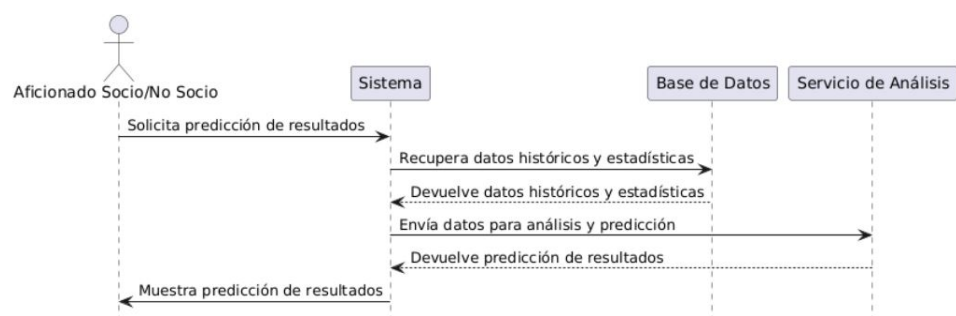


Figura 4.14: Diagrama de secuencia - Predicción de resultado

Capítulo 5

Diseño

5.1. Diseño de la base de datos

En este apartado se describe la estructura de la base de datos utilizada en la aplicación web, explicando el diseño de cada tabla, sus atributos y la justificación detrás de su implementación. La base de datos final es el resultado del trabajo previo de análisis y diseño, asegurando que la información necesaria esté correctamente organizada y accesible.

Uno de los aspectos fundamentales en la construcción de la base de datos ha sido la normalización , con el objetivo de minimizar la redundancia y maximizar la eficiencia en la gestión de datos. Por ello, las tablas han sido diseñadas siguiendo el principio de la tercera forma normal (3FN) , garantizando que cada atributo no clave dependa únicamente de la clave primaria y no de otros atributos no clave.

Además, se ha puesto especial énfasis en la definición adecuada de las relaciones entre entidades , ya que una correcta estructuración facilita la consulta de datos y evita redundancias o ciclos innecesarios. Otro factor clave ha sido el uso de índices optimizados para mejorar el rendimiento de las consultas, considerando en qué casos es más conveniente emplear claves primarias autogeneradas y cuándo es necesario definirlas manualmente para aportar mayor información.

5.1.1. Modelo Entidad-Relación

En este apartado, se presenta el diagrama entidad-relación de la aplicación web del club Campillo del Río CF, destacando las entidades clave y las relaciones que existen entre ellas. Además, se explica cómo este diseño contribuye a la integridad y optimización de la base de datos, garantizando un acceso rápido y seguro a la información.

La base de datos del sistema está compuesta por aficionados socios, aficionados no socios y administradores. Para su implementación, se ha optado por un modelo de entidades específicas, en el cual *Usuario* actúa como tabla principal, mientras que *Socio* y *Administrador* funcionan como tablas derivadas. Esta decisión se fundamenta en la necesidad de lograr un equilibrio entre normalización, claridad y eficiencia, permitiendo una gestión estructurada y optimizada de los diferentes tipos de usuarios dentro del sistema.

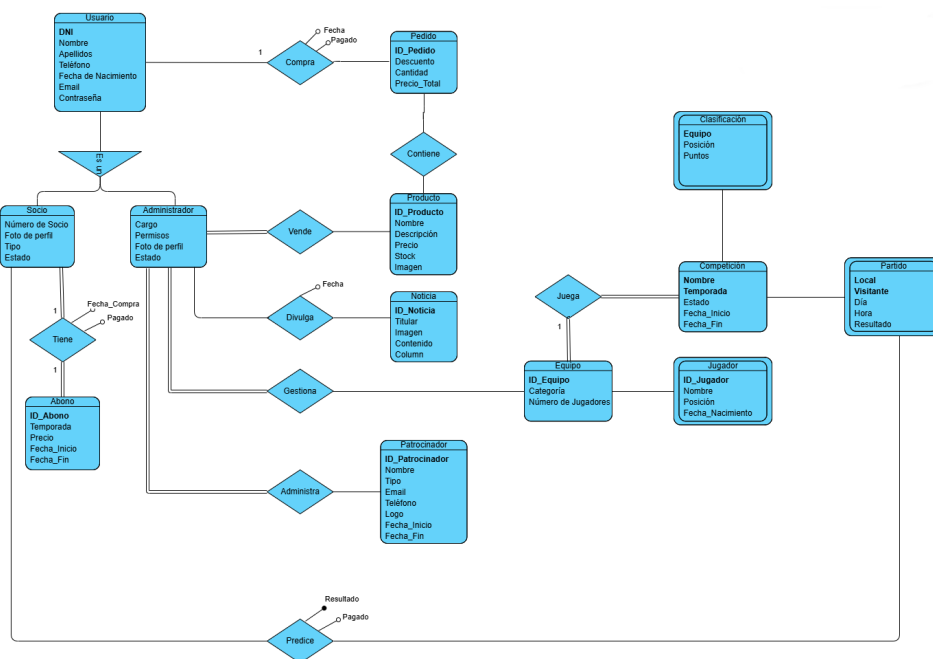


Figura 5.1: Diagrama Entidad-Relación

5.1.2. Paso a tablas y fusión

Una vez obtenido el diagrama entidad-relación, el siguiente paso consiste en transformarlo en tablas concretas. Este proceso comienza organizando todas las entidades y relaciones del modelo, asignando sus correspondientes claves primarias.

Posteriormente, se procede a combinar tablas cuando sea posible, con el objetivo de optimizar la estructura. Esta fusión debe cumplir dos condiciones esenciales: no puede generar pérdida de datos y debe contribuir a mejorar la eficiencia de almacenamiento y el rendimiento del sistema. Para que esta combinación sea viable, las tablas involucradas deben compartir la misma clave primaria y no deben derivar de relaciones de herencia.

- 1: Usuario(DNI, Nombre, Apellidos, Teléfono, Fecha_Nacimiento, Email, Contraseña)
CP CC
- 2: Socio(DNI, Num_Socio, Foto_Perfil, Tipo, Estado)
CP CC
- 3: Administrador(DNI, Cargo, Permisos, Foto_Perfil, Estado)
CP
- 4: Abono(ID_Abono, Temporada, Precio, Fecha_Inicio, Fecha_Fin)
CP
- 5: Post_Foro(ID_Post, Contenido, Moderado, Tipo, Fecha)
CP
- 6: Pedido(ID_Pedido, Descuento, Cantidad, Precio_Total)
CP
- 7: Producto(ID_Producto, Nombre, Descripción, Precio, Stock, Imagen)
CP CC
- 8: Noticia(ID_Noticia, Titular, Imagen, Contenido)
CP CC
- 9: Patrocinador(ID_Patrocinador, Nombre, Tipo, Email, Teléfono, Logo, Fecha_Inicio, Fecha_Fin)
CP CC
- 10: Equipo(ID_Equipo, Categoría, Num_Jugadores)
CP
- 11: Jugador(ID_Equipo, ID_Jugador, Nombre, Posición, Fecha_Nacimiento)
CP
- 12: Competición(Nombre, Temporada, Estado, Fecha_Inicio, Fecha_Fin)
CP

- $\frac{CE(12)}{CP}$
 13: Partido(Nombre, Temporada, Local, Visitante, Día, Hora, Resultado)
- $\frac{CE(1) \quad CE(4)}{CP}$
 14: Tiene(DNI, ID_Abono, Fecha, Compra, Pagado)
- $\frac{CE(1) \quad CE(5)}{CP}$
~~15: Publica(DNI, ID_Post)~~
- $\frac{CE(1) \quad CE(6)}{CP}$
 16: Compra(DNI, ID_Pedido, Fecha, Pagado)
- $\frac{CE(6) \quad CE(7)}{CP}$
~~17: Contiene(ID_Pedido, ID_Producto)~~
- $\frac{CE(1) \quad CE(7)}{CP}$
~~18: Vende(DNI, ID_Producto)~~
- $\frac{CE(1) \quad CE(8)}{CP}$
~~19: Divulga(DNI, ID_Noticia)~~
- $\frac{CE(1) \quad CE(10)}{CP}$
~~20: Gestiona(DNI, ID_Equipo)~~
- $\frac{CE(1) \quad CE(9)}{CP}$
~~21: Administra(DNI, ID_Patrocinador)~~
- $\frac{CE(1) \quad CE(13)}{CP}$
 22: Predice(DNI, Nombre, Temporada, Local, Visitante, Resultado, Pagado)
- $\frac{CE(10) \quad CE(12)}{CP}$
~~23: Juega(ID_Equipo, Nombre, Temporada)~~

5.1.3. Normalización de la base de datos

La normalización de bases de datos es un proceso que garantiza que el diseño lógico de una base de datos relacional esté libre de anomalías al momento de realizar operaciones como inserciones, actualizaciones y eliminaciones de datos.

En este caso, la base de datos cumple con la **primera forma normal (1FN)**, ya que no existen valores repetidos ni grupos de repetición dentro de las tablas; es decir, cada campo almacena datos atómicos. Además, se encuentra en **segunda forma normal (2FN)**, dado que ya cumple con la 1FN y todos los atributos no clave dependen completamente de la clave candidata, evitando dependencias parciales en claves compuestas.

Finalmente, la base de datos también cumple con la **tercera forma normal (3FN)**, ya que, además de estar en 2FN, no presenta dependencias transitivas, lo que significa que ningún atributo no clave depende de otro atributo no clave. Esto asegura un diseño más eficiente y libre de redundancias innecesarias.

5.2. Diseño de la arquitectura del sistema

El diseño arquitectónico más adecuado para este proyecto es, sin duda, el Modelo Vista-Controlador (MVC). Este enfoque es ideal para desarrollos de software en los que la interfaz visual desempeña un papel fundamental, además de permitir una clara separación entre la lógica del sistema y la información con la que interactúa el usuario, como se ilustra en la siguiente figura.

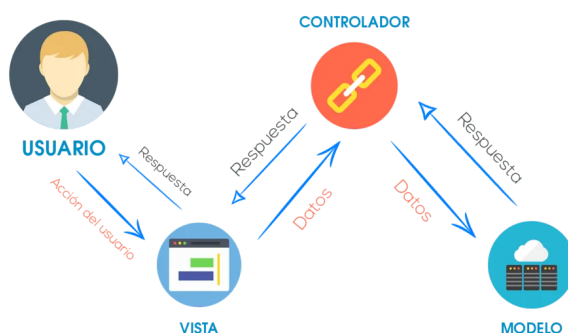


Figura 5.2: Modelo-Vista-Controlador (MVC)

5.2.1. Estructura de MVC

El Modelo Vista-Controlador divide la aplicación en tres componentes principales. Esta separación de responsabilidades facilita el mantenimiento y la escalabilidad del sistema, permitiendo modificaciones en una parte sin afectar las demás.

- **Vista:** Es el componente encargado de la presentación de la información al usuario, es decir existe una interacción directa con el usuario. Recibe datos del modelo y los muestra de forma adecuada, reflejando cualquier cambio en tiempo real.

En el contexto de un equipo de fútbol, las vistas se utilizan para mostrar información como la clasificación, el calendario de partidos y perfiles de jugadores.

- **Controlador:** Es el componente que actúa como intermediario entre el modelo y la vista. Maneja las entradas del usuario, procesa las solicitudes y determina la respuesta apropiada, actualizando el modelo o la vista según corresponda.

En la aplicación del equipo de fútbol, el controlador podría procesar solicitudes como registrar un nuevo jugador o actualizar los resultados de un partido.

- **Modelo:** Es el componente encargado de la gestión los datos y la lógica del sistema. Es responsable de acceder a la información, procesarla y responder a las solicitudes del controlador. Contiene las reglas y operaciones fundamentales del sistema, gestionando la manipulación y validación de datos.

En una aplicación para un equipo de fútbol, el modelo gestionaría información como jugadores, partidos, productos, etc.

5.2.2. Ventajas para la aplicación web

El uso del Modelo Vista-Controlador (MVC) [27] en una aplicación web para un equipo de fútbol ofrece diferentes ventajas, tanto en términos de organización del código como en la eficiencia del desarrollo y mantenimiento del software.

- **Alta cohesión:** Los componentes relacionados (ej: operaciones de "socio") se agrupan en un mismo lugar.
- **Bajo acoplamiento:** Las vistas no dependen directamente de la base de datos, ni los modelos de la interfaz.

- **Facilidad de modificación:** La separación de capas permite hacer cambios sin romper todo el sistema.
- **Compatibilidad con *frameworks*:** En la actualidad, la mayoría de *frameworks* webs son compatibles con el MVC.
- **Reutilización del código:** El modelo y el controlador pueden ser reutilizados en diferentes vistas sin necesidad de duplicar código.
- **Depuración y testeo simplificados:** La separación ayuda a la solución de posibles errores, ya que es más fácil la localización de dichos errores.
- **Mejor experiencia del usuario:** Al tener una estructura organizada, es posible desarrollar interfaces más dinámicas y eficientes, permitiendo tiempos de carga más rápidos y una mejor presentación de los datos.

5.3. Diseño de la interfaz de usuario

Antes de iniciar la implementación del proyecto, es fundamental crear bocetos de la interfaz de usuario. Esto es especialmente relevante en una aplicación web, donde la disposición de los elementos juega un papel clave para garantizar una experiencia visual atractiva y una navegación fluida [23]. La utilización de MockUps o bocetos permite diseñar y prever cómo será el resultado final, facilitando ajustes y mejoras antes del desarrollo definitivo.

5.3.1. Mockups de la aplicación web

Al entrar en la **página principal del club**, el usuario se encuentra con la página de inicio. Aquí lo reciben las últimas noticias, los próximos partidos y accesos rápidos a las secciones más importantes.

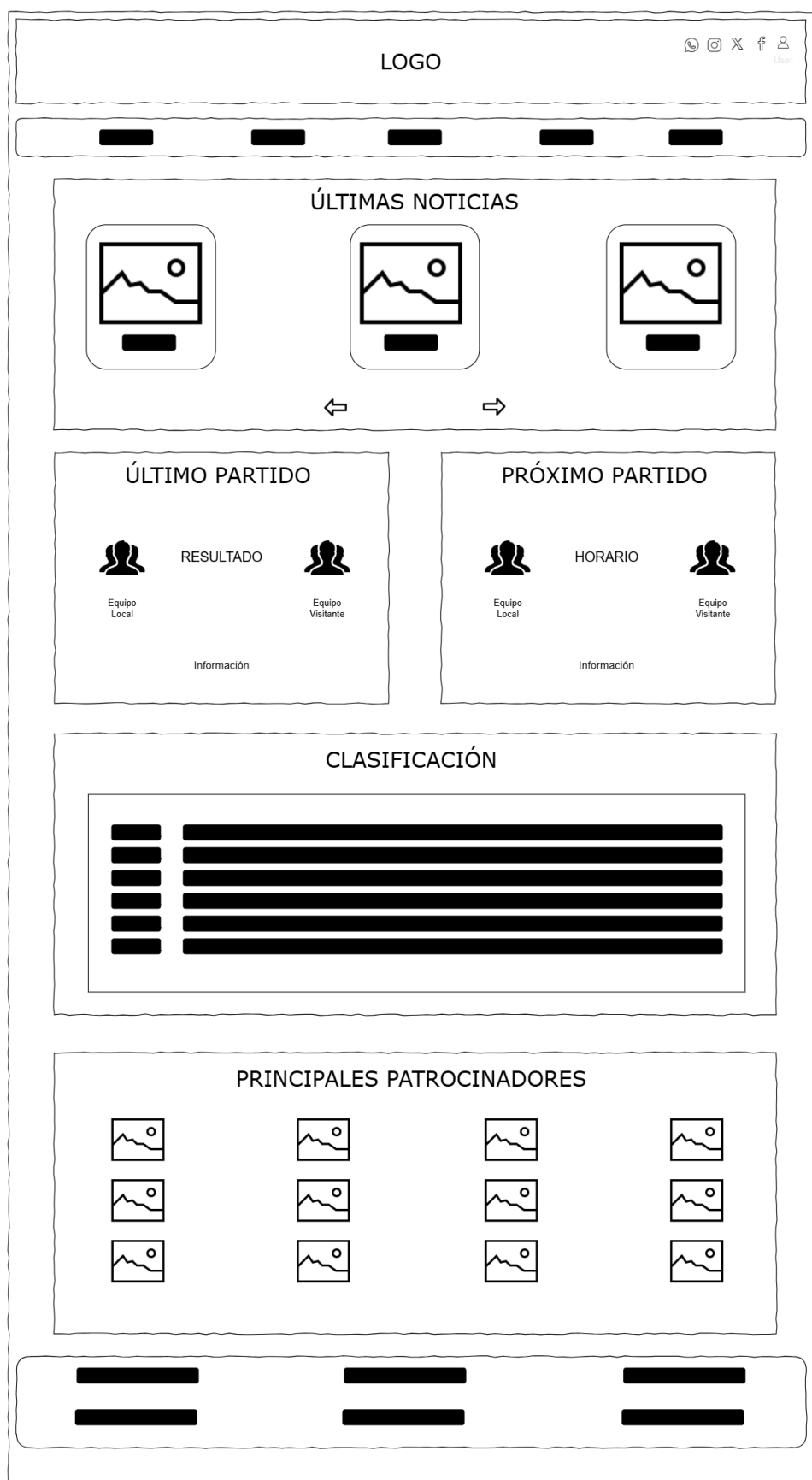


Figura 5.3: Mockups de inicio

Ahora navega hacia la **sección de Plantillas y Noticias**. En esta página, puede explorar la lista de jugadores con sus fotos, junto con una sección de noticias destacadas que muestran imágenes y titulares.

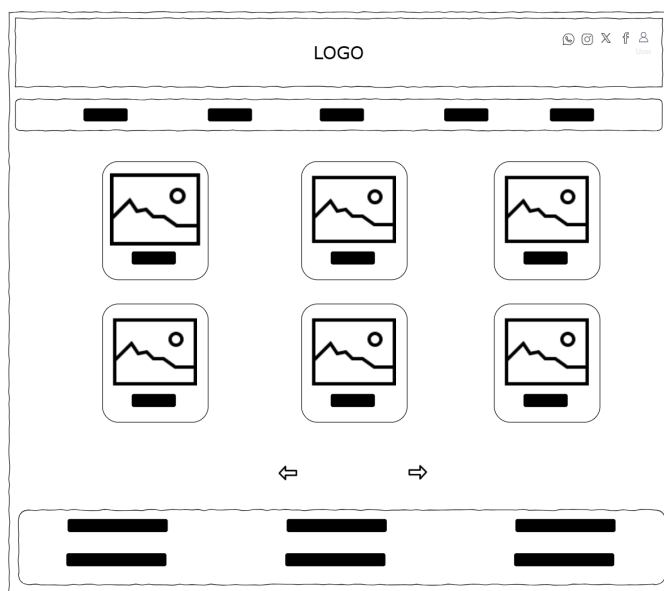


Figura 5.4: Mockups de plantilla y noticias

Si hace clic en un jugador, accede a su ficha personal. De igual manera, al seleccionar una noticia específica.

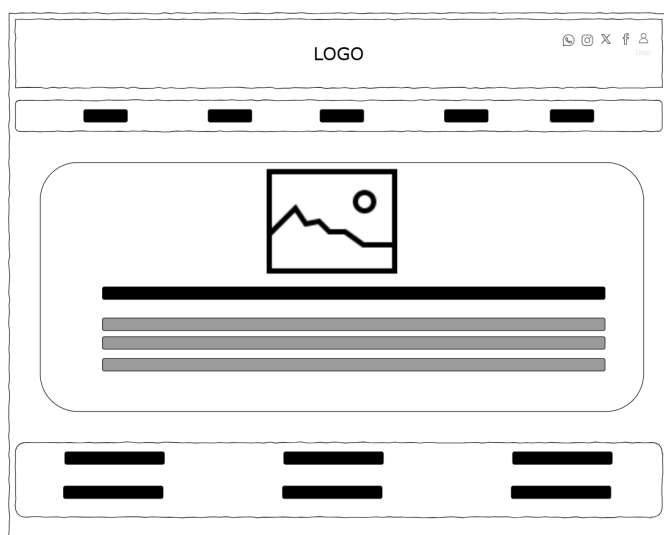


Figura 5.5: Mockups de jugador y noticia

A continuación, visita la **tienda oficial**. Si un artículo le llama la atención, puede acceder a su **página de producto**.

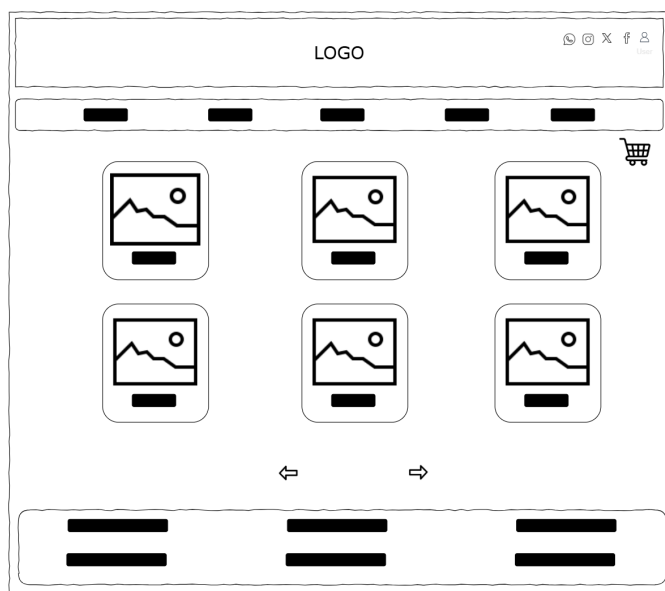


Figura 5.6: Mockup de tienda

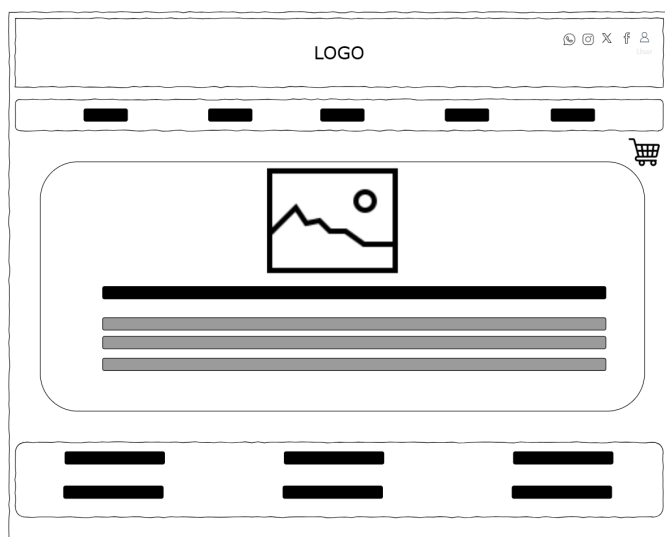
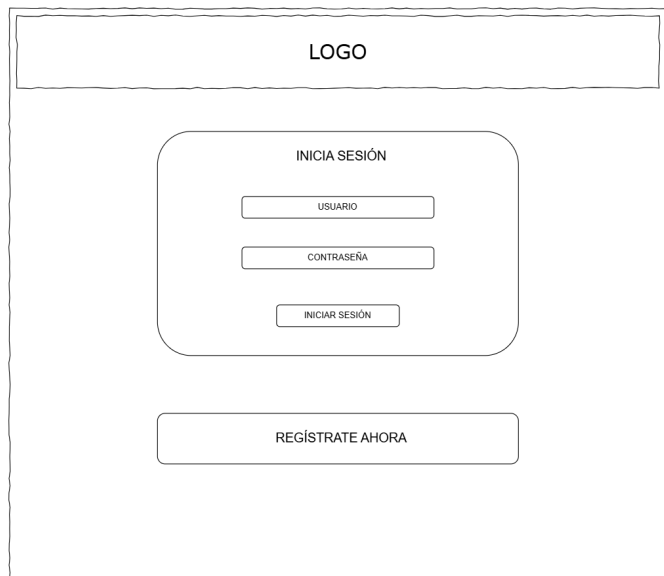


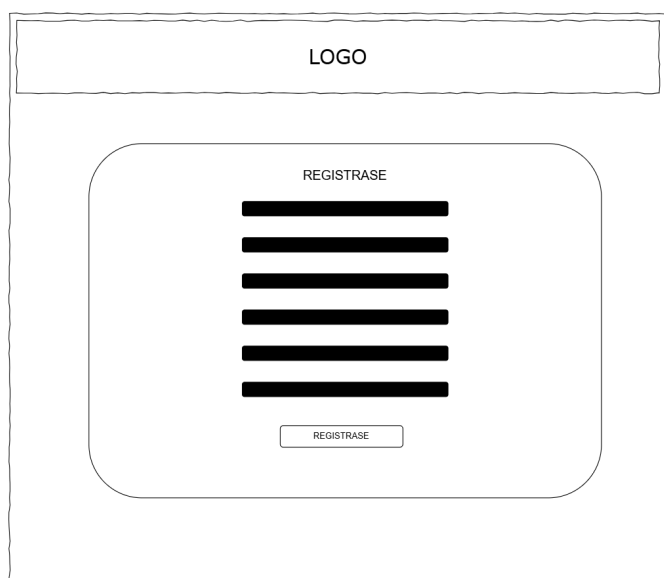
Figura 5.7: Mockup de producto

Para completar la compra, necesita iniciar sesión. Se dirige a la **página de inicio de sesión**. En caso de no tener una cuenta, puede registrarse fácilmente desde la **página de registro**.



The mockup shows a login interface. At the top, there is a rectangular box labeled "LOGO". Below this, centered, is a rounded rectangular container labeled "INICIA SESIÓN". Inside this container, there are three input fields: the first is labeled "USUARIO", the second is labeled "CONTRASEÑA", and below them is a button labeled "INICIAR SESIÓN". Below the "INICIA SESIÓN" container, there is a separate rectangular button labeled "REGÍSTRATE AHORA".

Figura 5.8: Mockups de inicio de sesión



The mockup shows a registration interface. At the top, there is a rectangular box labeled "LOGO". Below this, centered, is a rounded rectangular container labeled "REGISTRASE". Inside this container, there are six horizontal black bars representing input fields for registration details. Below these bars is a button labeled "REGISTRASE".

Figura 5.9: Mockups de registro

Si es socio o administrador, tiene acceso a un **página con las funciones de específicas de administrador o socio.**

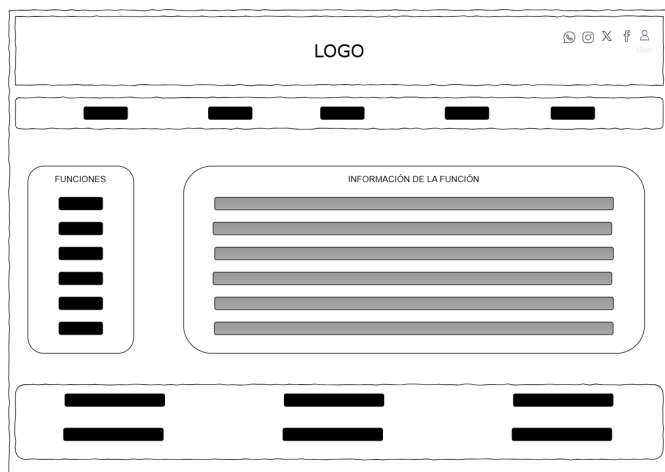


Figura 5.10: Mockup de socio y administrador

Finalmente, el usuario accede a una página estática de información, y según la página puede variar ligeramente el diseño.

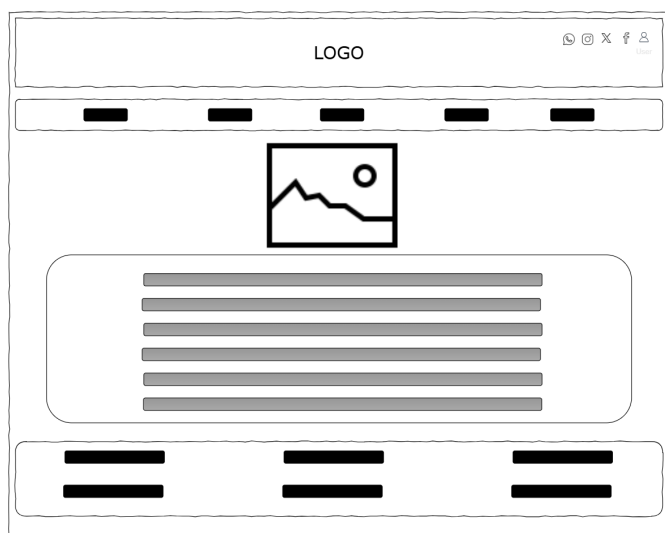


Figura 5.11: Mockup de página estática

Capítulo 6

Implementación

En este capítulo se expone detalladamente el proceso de implementación de la solución propuesta, abordando las herramientas, lenguajes y tecnologías empleadas, así como la organización general del sistema. A lo largo del capítulo se desglosan los distintos componentes que conforman la solución, desde el entorno de desarrollo hasta el despliegue final, permitiendo comprender cómo se ha construido e integrado cada parte del sistema [16].

Se inicia con una descripción del entorno de trabajo y los lenguajes de programación utilizados, diferenciando claramente entre los componentes del *backend* y del *frontend* debido a sus características y necesidades específicas. Posteriormente, se abordan por separado la implementación de la base de datos, del *backend* y del *frontend*, explicando la estructura modular adoptada, las dependencias utilizadas y las principales clases y servicios desarrollados.

Finalmente, el capítulo concluye con el proceso de despliegue de la aplicación, tanto del *backend* y la base de datos como del *frontend*, explicando cómo se ha realizado la puesta en producción del sistema.

Este análisis integral permite comprender en profundidad la implementación técnica del proyecto, asegurando la cohesión y eficiencia entre sus diferentes componentes, y sentando las bases para una solución escalable y robusta. Para una revisión más detallada del código fuente, se recomienda consultar el repositorio oficial de GitHub del proyecto (https://github.com/aplicacionwebcampillo/Aplicacion_Web).

6.1. Entorno y lenguajes utilizados

En este apartado se detallan los lenguajes de programación, herramientas, bibliotecas, *frameworks* y entornos empleados en el desarrollo del sistema, distinguiendo entre los componentes de *backend* y *frontend*. La elección de cada tecnología se basó en criterios como la eficiencia, escalabilidad, facilidad de uso, soporte comunitario y compatibilidad entre los distintos módulos del sistema.

6.1.1. Lenguajes de programación

El sistema fue desarrollado principalmente utilizando dos lenguajes de programación:

- **Python:** fue utilizado para la implementación del *backend*. Este lenguaje se eligió por su sintaxis clara, su amplia adopción en el desarrollo de APIs web modernas, y por contar con una rica colección de bibliotecas orientadas al desarrollo *backend*, validación de datos y manejo de bases de datos relacionales.
- **TypeScript + JSX (archivos .tsx):** en el desarrollo del *frontend* se utilizó el lenguaje TypeScript junto con la sintaxis JSX. Esta combinación, común en proyectos React, permite una programación tipada y segura en entornos complejos de interfaces de usuario. También se emplearon tecnologías como HTML y CSS [10] [9], siendo estos los lenguajes base para la estructura y estilo de las páginas web.

6.1.2. Frameworks y bibliotecas utilizadas

Backend

El desarrollo del *backend* se realizó utilizando el framework FastAPI, el cual permite crear aplicaciones web y APIs de alto rendimiento haciendo uso de asincronía nativa con `async/await`. FastAPI destaca por su velocidad, su integración automática y su sistema de validación de datos mediante anotaciones de tipo.

A continuación, se listan las bibliotecas y herramientas principales empleadas:

- **FastAPI:** framework principal para la construcción de la API.
- **SQLAlchemy:** herramienta ORM (Object-Relational Mapping) utilizada para definir, mapear y manipular las entidades de la base de datos de manera programática mediante clases Python.

- **Pydantic:** se utilizó para la validación y serialización de datos. Permite definir esquemas de datos fuertemente tipados que FastAPI utiliza internamente para manejar peticiones y respuestas de forma segura.

Frontend

El *frontend* se construyó utilizando la biblioteca React [32], una de las tecnologías más populares para el desarrollo de interfaces de usuario dinámicas y basadas en componentes reutilizables. Para mejorar el rendimiento y la experiencia de desarrollo, se utilizó Vite como herramienta de construcción y servidor de desarrollo. Vite ofrece tiempos de recarga muy reducidos y configuración mínima.

Además, se incorporaron tecnologías auxiliares como:

- **Tailwind CSS:** un framework CSS utilitario que permite aplicar estilos directamente en las clases HTML, lo que facilita una maquetación rápida, coherente y altamente personalizable sin necesidad de escribir CSS desde cero.

6.1.3. Base de datos

Para el almacenamiento persistente de datos se utilizó el sistema gestor de bases de datos PostgreSQL [29], una solución robusta, escalable y ampliamente utilizada en entornos de producción. PostgreSQL ofrece soporte para tipos de datos avanzados, integridad referencial, y transacciones ACID.

El diseño del esquema de base de datos y la gestión de las relaciones entre entidades se realizaron directamente mediante código en SQLAlchemy [1], sin el uso de herramientas gráficas ni *frameworks* de migraciones. Esta aproximación permitió mantener un control detallado del modelo de datos, asegurando consistencia entre el modelo lógico y la implementación.

6.1.4. Entorno de desarrollo

El entorno de desarrollo utilizado fue el siguiente:

- **Sistema operativo:** Ubuntu (basado en Linux), elegido por su estabilidad, eficiencia en el manejo de recursos y compatibilidad con herramientas de desarrollo *backend*.
- **Editor de código:** Visual Studio Code (VS Code), un editor ligero, extensible y con excelente soporte para múltiples lenguajes y *frameworks*, incluyendo Python y React.

- **Entorno virtual:** para el desarrollo del *backend*, se utilizó un entorno virtual gestionado con `venv`, lo que permitió aislar las dependencias del proyecto y evitar conflictos con otras instalaciones de Python en el sistema.

6.1.5. Control de versiones y flujo de trabajo

Durante todo el desarrollo del sistema se empleó Git [30] como sistema de control de versiones, con el código fuente alojado en un repositorio de GitHub. Esta herramienta permitió llevar un control riguroso de los cambios realizados, facilitando la organización del código, la trazabilidad de errores y la posibilidad de volver a versiones anteriores en caso necesario.

El desarrollo se realizó de manera individual, siguiendo una estrategia secuencial compuesta por las siguientes etapas:

- Diseño e implementación de la base de datos.
- Desarrollo del *backend* y creación de los *endpoints* necesarios.
- Construcción del *frontend* y conexión con la API.
- Pruebas y validaciones funcionales.
- Despliegue del sistema completo.

6.1.6. Gestión de dependencias y herramientas adicionales

Para la gestión de dependencias del proyecto, en específico las dependencias del *backend*, se utilizó un archivo `requirements.txt`, en el cual se declararon todas las bibliotecas necesarias para la ejecución del proyecto. Este archivo permite una fácil instalación del entorno en otros sistemas mediante el uso de herramientas como `pip`.

Durante la etapa de validación, se utilizaron herramientas como Postman para realizar pruebas manuales sobre los *endpoints* del *backend*, permitiendo verificar el correcto funcionamiento de las rutas y la validez de las respuestas. Las pruebas de interfaz se realizaron manualmente desde el navegador, evaluando la interacción entre el usuario y el sistema.

Finalmente, una vez concluido el desarrollo, se procedió al despliegue completo del sistema en la plataforma Render.com [15]. Esta plataforma permitió alojar y ejecutar tanto la base de datos como los componentes de *backend* y *frontend* en un entorno accesible vía web, facilitando su puesta en producción y evaluación remota.

6.2. Implementación de la base de datos

La implementación de la base de datos se llevó a cabo combinando el uso de scripts SQL tradicionales para la creación de tablas con el empleo de SQLAlchemy en el *backend*, lo que permitió una integración eficiente con la lógica de la aplicación desarrollada en Python [11] [20].

Durante la fase de desarrollo, se optó por utilizar PostgreSQL como sistema de gestión de base de datos, debido a su robustez, rendimiento y cumplimiento con el estándar SQL. La instancia de PostgreSQL utilizada fue completamente local, lo que facilitó el control del entorno y permitió un desarrollo más directo, sin dependencias externas durante las etapas iniciales del proyecto.

6.2.1. Creación y estructura de la base de datos

La base de datos fue inicialmente construida mediante scripts SQL manuales, lo cual permitió definir de forma precisa la estructura inicial del esquema, incluyendo la creación de tablas, restricciones, claves primarias, claves foráneas y relaciones entre entidades.

Además de la estructura básica, se desarrollaron múltiples funciones y triggers SQL para implementar las reglas de validación y lógica de negocio directamente en la base de datos.

Estas funciones y disparadores (triggers) permitieron garantizar la integridad de los datos, aplicar restricciones adicionales antes de insertar, modificar o eliminar registros, y asegurar que todas las operaciones cumplieran las condiciones necesarias para un funcionamiento coherente del sistema. Esta lógica embebida en el motor de base de datos actúa como una capa de validación adicional, complementando las verificaciones realizadas en el *backend*.

Una vez establecida la estructura de la base de datos, se procedió a la definición de los modelos correspondientes en el *backend* utilizando SQLAlchemy. Esta biblioteca ORM permitió mapear cada tabla a una clase de Python, lo que facilitó una interacción más fluida entre la lógica de negocio y la base de datos subyacente. A través de estas clases, fue posible realizar operaciones CRUD (Create, Read, Update, Delete) de forma eficiente y segura, delegando en SQLAlchemy el manejo de las consultas SQL internas.

Las relaciones entre las distintas entidades del modelo fueron definidas mediante las capacidades declarativas de SQLAlchemy, haciendo uso de atributos como `ForeignKey` y mecanismos de relación como `relationship`. Esto permitió reflejar adecuadamente las asociaciones entre entidades, dentro de la estructura orientada a objetos del código.

6.2.2. Conexión con el *backend*

La conexión entre el *backend* y la base de datos se estableció utilizando SQLAlchemy, definiendo una cadena de conexión conforme al formato estándar de PostgreSQL. Esta conexión se mantuvo encapsulada en un módulo específico encargado de gestionar las sesiones y asegurar que las operaciones sobre la base de datos se realizaran de forma controlada y consistente.

El uso de entornos virtuales (*venv*) en el desarrollo del *backend* permitió aislar las dependencias y garantizar que las bibliotecas necesarias, incluida SQLAlchemy, estuvieran correctamente instaladas y configuradas.

6.2.3. Evolución y mantenimiento del esquema

Aunque la estructura inicial de la base de datos fue definida desde el comienzo, el desarrollo iterativo del proyecto llevó a la necesidad de realizar algunos ajustes menores. Estos cambios, como la inclusión de nuevos campos, el refinamiento de relaciones entre entidades o la modificación de funciones y triggers, fueron gestionados de manera controlada mediante la combinación de Alembic y scripts SQL.

Este enfoque permitió mantener la coherencia entre los modelos definidos en el código, el esquema de la base de datos y la lógica de validación embebida, asegurando así la estabilidad y fiabilidad del sistema durante todo el ciclo de desarrollo.

6.3. Implementación del *backend*

El desarrollo del *backend* se llevó a cabo utilizando el framework **FastAPI** [22], una herramienta moderna y eficiente para construir APIs web en Python, con soporte para programación asíncrona y validación automática de datos. La estructura del proyecto responde a una arquitectura en capas, lo que permite mantener una separación clara entre la lógica de negocio, la gestión de datos, la validación y la presentación de la API.

6.3.1. Estructura general del proyecto

El *backend* se organiza dentro de la carpeta principal **app/**, que contiene los diferentes módulos funcionales distribuidos por responsabilidades:

routers/: contiene los *endpoints* organizados por módulo funcional (por ejemplo, **usuario.py**, **socio.py**, entre otros). Estos ficheros definen las rutas HTTP de la API y son el punto de entrada para las operaciones disponibles.

models/: almacena los modelos ORM definidos con SQLAlchemy, los cuales representan las tablas y relaciones de la base de datos.

schemas/: incluye los esquemas definidos con Pydantic, utilizados para validar y estructurar los datos de entrada y salida de los *endpoints*.

crud/: aquí se implementan las funciones responsables de la lógica de negocio y acceso a datos, como inserciones, actualizaciones, consultas o validaciones específicas.

database.py: módulo que gestiona la conexión con la base de datos PostgreSQL, configurando la sesión mediante SQLAlchemy.

main.py: punto de entrada de la aplicación FastAPI. En este archivo se inicializa la aplicación, se conectan las rutas y se configura el middleware básico.

Esta organización permite una clara separación de responsabilidades, facilita la mantenibilidad del código y permite una escalabilidad futura del sistema.

6.3.2. *Endpoints* y funcionalidades de la API

El *backend* expone entre 30 y 40 *endpoints* RESTful, organizados por entidad funcional. Se implementan operaciones CRUD completas para diversas entidades como: usuarios, socios y abonos, carrito de compras, productos, pedidos y pagos y predicciones en competiciones

Además de los endpoints estándar (GET, POST, PUT, DELETE, PATCH), también se implementaron rutas personalizadas con lógica específica, como: `validar_pago_socio_abono`, `/confirmar_pedido`, `/aplicar_descuento`, entre otros

Gracias al uso de FastAPI, cada endpoint está debidamente documentado y es accesible desde la interfaz automática generada en `/docs`.

6.3.3. Validación de datos

La validación de datos se realiza principalmente mediante el uso de Pydantic [8], que permite definir esquemas de entrada y salida. Estos esquemas garantizan que los datos recibidos por la API cumplan con el formato y tipo esperados, ofreciendo una capa automática de seguridad y control.

Ejemplos de esquemas utilizados incluyen:

`UsuarioCreate`, `SocioAbonoCreate`, `CarritoUpdate`, etc.

Además, se aplican validaciones manuales específicas en el código dentro de las funciones `crud`, como por ejemplo: verificar si un usuario o socio

ya existe, validar el formato y unicidad de un DNI, confirmar fechas válidas en registros de abonos y comprobar la existencia de registros antes de modificarlos.

6.3.4. Gestión de errores

La gestión de errores se realiza mediante el uso de la clase `HTTPException` de FastAPI, lo que permite devolver respuestas personalizadas según el tipo de error detectado. Entre los códigos de estado más utilizados se encuentran:

404 `Not Found`: cuando un recurso no existe.

400 `Bad Request`: para errores de validación lógica o conflictos de datos (por ejemplo, registros duplicados).

6.3.5. Autenticación y autorización

La autenticación de usuarios se implementa mediante el sistema de OAuth2 con PasswordBearer [18], utilizando el formulario `OAuth2PasswordRequestForm` proporcionado por FastAPI. Este mecanismo permite la autenticación mediante nombre de usuario y contraseña, devolviendo un token de acceso en formato JSON Web Token (JWT) [19].

Este sistema proporciona una base sólida para la protección de *endpoints*, permitiendo autenticar a los usuarios antes de acceder a recursos sensibles del sistema.

6.3.6. Conexión con la base de datos

La conexión al sistema gestor de base de datos PostgreSQL se realiza utilizando SQLAlchemy, y se configura a través de una cadena de conexión definida en el módulo `database.py`. Se gestionan sesiones por petición, siguiendo las buenas prácticas recomendadas por FastAPI para evitar conflictos de concurrencia y mantener las transacciones controladas.

6.3.7. Servicios externos o integraciones

Aunque el *backend* se comunica principalmente con la base de datos y con el *frontend*, también integra varios servicios externos que amplían sus funcionalidades:

Cloudinary: se utiliza como servicio externo para el almacenamiento de imágenes. Las imágenes cargadas por los usuarios (por ejemplo, productos, perfiles, etc.) se suben a la plataforma de Cloudinary [7], y en la base de

datos solo se almacena la URL generada. Esto reduce la carga del servidor y mejora el rendimiento.

FastAPI Mail: se utiliza la librería FastMail para el envío automático de correos electrónicos. Se envían notificaciones tanto a los administradores como a los clientes en eventos clave como: confirmación de una compra y validación de pagos.

La configuración del sistema de correo se realiza mediante Connection-Config, utilizando parámetros seguros definidos en variables de entorno.

6.3.8. Integración mediante Web Scraping para datos deportivos

Para las tablas relacionadas con **competición**, **clasificación** y **partido**, se implementó un proceso automatizado de obtención de datos mediante *web scraping* a la página oficial de la Real Federación Andaluza de Fútbol (RFAF). Este enfoque garantiza que la información mostrada en el sistema sea precisa y esté siempre actualizada, sin depender únicamente de fuentes manuales o APIs externas no oficiales.

La extracción de datos se realizó con las siguientes tecnologías:

Python: lenguaje principal para la implementación de los scripts.

Playwright: biblioteca para automatización de navegadores web que permite navegar, interactuar y extraer datos de páginas dinámicas con JavaScript [21].

Asyncio: módulo de Python para programación asíncrona, utilizado para optimizar el proceso de scraping, permitiendo manejar múltiples tareas concurrentemente y mejorar la eficiencia y velocidad de obtención de la información.

Este sistema de scraping se ejecuta periódicamente para actualizar las tablas correspondientes en la base de datos, asegurando que las clasificaciones, resultados y detalles de partidos reflejen siempre la información oficial disponible en la RFAF.

6.3.9. Otros aspectos técnicos

Se utilizaron dependencias inyectadas mediante el decorador `@Depends` para el acceso a la base de datos en los *endpoints*.

Las configuraciones del entorno, como la URL de conexión a la base de datos o claves sensibles, se manejan mediante variables de entorno almacenadas en un archivo `.env`.

6.4. Implementación del *frontend*

La implementación del *frontend* se realizó utilizando React junto con Vite [31] como herramienta de construcción y desarrollo, lo que garantiza una experiencia de desarrollo rápida y eficiente. A continuación se describen los aspectos más relevantes de su estructura, tecnologías empleadas, comunicación con el *backend*, manejo de autenticación, diseño y funcionalidades principales.

6.4.1. Estructura del proyecto

El proyecto *frontend* sigue una organización clara y tradicional en React, basada en la separación de responsabilidades. La estructura principal se encuentra en la carpeta `src/`, donde destacan las siguientes subcarpetas:

- `pages/`: contiene las páginas principales de la aplicación, tales como `Carrito.tsx`, `Login.tsx`, entre otras.
- `components/`: componentes reutilizables de la interfaz, como barras de navegación (`Navbar`), tarjetas de productos (`ProductoCard`), y otros elementos UI.
- `context/`: gestión del estado global mediante React Context API, incluyendo contextos como `CarritoContext` y `useAuth` para autenticación.
- `hooks/`: para hooks personalizados que encapsulan lógica reutilizable.
- `assets/`: archivos estáticos, como imágenes o íconos.
- `App.tsx` y `main.tsx`: punto de entrada y composición general de la aplicación.

6.4.2. Tecnologías y herramientas

Para el desarrollo del *frontend* se usaron las siguientes tecnologías y librerías principales:

- **React** junto con **Vite**, que provee un entorno moderno y eficiente para el desarrollo *frontend*.
- **Tailwind CSS**, usado para los estilos, lo que permite un diseño altamente modular y responsivo basado en clases utilitarias.
- **React Router**, para la gestión de rutas y navegación entre páginas.

- **Context API**, que gestiona el estado global de la aplicación, principalmente para el carrito de compras y la autenticación del usuario.

6.4.3. Comunicación con el *backend*

Las peticiones HTTP hacia el *backend* se realizan utilizando la API nativa `fetch` de JavaScript. Actualmente, estas llamadas se encuentran distribuidas directamente dentro de los componentes que requieren la información, sin una capa intermedia centralizada de servicios o API. El manejo de errores y estados de carga se implementa de forma local mediante bloques `try/catch`.

6.4.4. Autenticación en el *frontend*

La autenticación del usuario se maneja mediante tokens JWT que se almacenan en `localStorage` tras un login exitoso. El contexto de autenticación, proporcionado por `useAuth` mediante React Context, permite acceder al estado de usuario y controlar la visibilidad de componentes y funcionalidades. Las rutas protegidas se controlan a nivel de componentes, mostrando alertas o restringiendo contenido si no existe una sesión activa.

6.4.5. Diseño y experiencia de usuario

Se aplicó un diseño responsivo mediante las clases de Tailwind CSS [33], usando los breakpoints estándar `sm:`, `md:`, `lg:` para garantizar una experiencia adecuada en distintos dispositivos.

6.4.6. Funcionalidades principales

La aplicación *frontend* cuenta con diversas vistas y funcionalidades que se organizan en páginas estáticas y dinámicas para ofrecer una experiencia completa al usuario:

- **Home**: incluye secciones variadas como últimas noticias, información sobre el último y próximo partido, la clasificación actual del equipo y un listado de patrocinadores. Esta página sirve como punto de entrada para que los usuarios accedan a la información más relevante y actualizada del club.
- **Noticias**: página dedicada a la publicación y visualización de noticias relacionadas con el club y sus actividades.

- **Plantilla y calendario:** secciones específicas para mostrar la plantilla de jugadores y el calendario de partidos, proporcionando información dinámica y actualizada.
- **Tienda:** espacio destinado a la venta de productos oficiales del club, con funcionalidades para agregar artículos al carrito y realizar compras.
- **Páginas estáticas:** contienen información institucional del club, como historia, contacto, y otras secciones informativas no sujetas a cambios frecuentes.
- **Gestión de usuarios:** páginas que permiten a los usuarios registrarse, iniciar sesión y administrar su perfil personal.
- **Gestión de socios:** funcionalidades específicas para los socios, incluyendo la gestión de abonos, pagos y beneficios asociados.
- **Panel administrativo:** espacio reservado para administradores del sistema, desde donde se gestionan contenidos, usuarios, productos y otras funcionalidades propias del *backend*.

Estas funcionalidades permiten cubrir las necesidades tanto de usuarios regulares como de socios y administradores, integrando información dinámica y estática para una experiencia completa y personalizada.

6.5. Despliegue

El despliegue del sistema se ha realizado de manera completamente *on-line*, utilizando los servicios en la nube que ofrece la plataforma Render.com tanto para el *backend* como para la base de datos y el *frontend*. A continuación, se detalla el proceso seguido en cada componente del sistema.

6.5.1. Despliegue del *backend* y base de datos

El *backend* de la aplicación, desarrollado con FastAPI, se encuentra desplegado en Render.com mediante un servicio web. Este servicio se configura para tomar el código fuente desde un repositorio de GitHub, lo que permite automatizar el proceso de despliegue cada vez que se actualiza el código.

Para desplegar el *backend* se siguieron los siguientes pasos:

- Se subió el código fuente del *backend* al repositorio en GitHub.
- Se vinculó el repositorio a un nuevo servicio web en Render.com.

- Se configuraron las variables de entorno necesarias para el correcto funcionamiento del *backend*, tales como la URL de conexión a la base de datos, claves secretas y otros parámetros sensibles.
- Se definió el comando de instalación de dependencias: `pip install -r requirements.txt`.
- Se estableció el comando de inicio del servidor: `uvicorn main:app --host 0.0.0.0 --port $PORT`.
- Finalmente, se monitorearon los registros (logs) de Render para verificar que el servicio se levante correctamente.

La base de datos PostgreSQL también se encuentra alojada en Render.com, haciendo uso de su servicio de bases de datos gestionadas.

6.5.2. Despliegue del *frontend*

El *frontend*, desarrollado con React y Vite, también ha sido desplegado mediante un servicio web en Render.com. El proceso de despliegue fue igualmente automatizado a través de la vinculación con su repositorio en GitHub.

El flujo de despliegue incluyó los siguientes pasos:

- Vinculación del repositorio del *frontend* a un nuevo servicio web en Render.com.
- Configuración del entorno de instalación con el comando `npm install`.
- Definición del comando de construcción del proyecto: `npm run build`.
- Establecimiento del directorio de salida de la aplicación construida: `dist`.
- Render.com se encarga automáticamente de construir y desplegar la aplicación al detectar cambios en el repositorio.

En cuanto a la conexión con el *backend*, esta se realiza manualmente dentro del código del *frontend*, configurando las URLs de las peticiones API a la dirección pública del *backend* desplegado. Esta URL puede definirse mediante variables de entorno o archivos de configuración específicos.

Finalmente, se ha configurado un dominio personalizado para el *frontend*, adquirido a través de Hostinger. Render.com permite asociar dicho dominio fácilmente, así como habilitar automáticamente HTTPS para asegurar la conexión de los usuarios al sitio web.

Capítulo 7

Experimentos y resultados

En este capítulo se presentan las pruebas realizadas sobre el sistema implementado, con el objetivo de verificar su correcto funcionamiento, estabilidad y cobertura de las funcionalidades esperadas. Se realizaron distintos tipos de pruebas, incluyendo pruebas unitarias automatizadas, pruebas de integración, pruebas manuales y pruebas de navegación enfocadas en la experiencia del usuario [13] [2].

7.1. Pruebas Unitarias

Para la verificación de los distintos *endpoints* del *backend*, se empleó Postman [26] junto con Newman [25] para automatizar la ejecución de pruebas unitarias sobre la API REST desarrollada en FastAPI. Las pruebas se enfocaron en operaciones CRUD y funcionalidades específicas del sistema.

Las pruebas cubrieron un total de:

- **104 solicitudes** realizadas a los diferentes *endpoints* del *backend*.
- **0 fallos detectados**, lo que indica un funcionamiento estable en todos los casos evaluados.
- **0 tests omitidos**.
- Tiempo total de ejecución: **35.5 segundos**.
- Tiempo medio de respuesta por solicitud: **318 ms**.

Además, se comprobó la respuesta esperada de cada petición, incluyendo los códigos de estado HTTP como 200 OK, 404 Not Found, y 500 Internal Server Error, según el caso probado.

Estas pruebas permiten asegurar que los *endpoints* implementados responden correctamente bajo condiciones normales, y que errores como recursos no encontrados o entradas inválidas son tratados adecuadamente.

7.2. Pruebas Manuales

También se realizaron pruebas manuales a través del navegador para verificar:

- El correcto funcionamiento de los formularios de registro, login y edición de perfil.
- Visualización de productos, noticias, calendario y plantilla.
- Respuesta del sistema ante entradas incorrectas o malformadas.
- Gestión del carrito y flujos de compra completos desde el *frontend*.
- Comprobación visual del diseño responsive y la interacción con dispositivos móviles.

Estas pruebas fueron útiles para detectar errores en la interfaz y validar que los flujos de usuario funcionaran como se espera desde el punto de vista funcional y visual.

7.3. Pruebas de Navegación y Flujo del Usuario

Para evaluar la experiencia del usuario final, se realizaron recorridos completos por las funcionalidades del sistema, tanto para usuarios anónimos como para usuarios autenticados (clientes, socios y administradores). Se prestó especial atención a:

- Accesibilidad a secciones clave desde la navegación principal.
- Fluidez y lógica en la secuencia de páginas y acciones (login → tienda → carrito → compra).
- Visibilidad de errores, mensajes de éxito y confirmaciones.
- Control de acceso a rutas protegidas en función del estado de sesión.

Estas pruebas contribuyeron a optimizar la usabilidad general del sistema y ajustar comportamientos que pudieran generar confusión en el flujo de interacción.

Capítulo 8

Conclusiones y trabajos futuros

8.1. Objetivos alcanzados

El objetivo principal de este Trabajo Fin de Grado era desarrollar una aplicación web funcional, escalable y adaptada a las necesidades reales de un club deportivo. Este objetivo ha sido alcanzado con éxito, permitiendo:

- Diseñar e implementar una plataforma web moderna y completa para la gestión integral de un equipo de fútbol, desde la administración interna hasta la interacción con sus aficionados.
- Ofrecer funcionalidades como gestión de usuarios (aficionados, socios y administradores), venta de productos mediante una tienda *online*, sistema de abonos, seguimiento de competiciones, clasificación, calendario y visualización de noticias.
- Incorporar medidas de seguridad y buenas prácticas de desarrollo tanto en el *backend* (con autenticación OAuth2 y uso de roles) como en el *frontend* (control de rutas, validación de sesión, persistencia con JWT).
- Asegurar un diseño responsive, limpio y funcional gracias al uso de Tailwind CSS, garantizando una experiencia óptima tanto en escritorio como en dispositivos móviles.
- Desplegar correctamente el sistema en la nube mediante Render.com, incluyendo el *backend*, la base de datos y el *frontend*, con integración de un dominio personalizado y uso de servicios externos como Cloudinary (para imágenes) y envío de correos electrónicos.

- Validar el sistema mediante pruebas automatizadas (con Postman/-Newman) y manuales, comprobando su estabilidad y funcionalidad completa.

En definitiva, todos los objetivos específicos establecidos al inicio del proyecto han sido cumplidos, y la aplicación se encuentra plenamente operativa para su uso en un entorno real.

8.2. Aprendizaje y reflexión personal

Este proyecto ha supuesto un reto técnico y personal de gran valor. A lo largo del desarrollo se han consolidado numerosos conocimientos adquiridos durante la carrera, tanto a nivel teórico como práctico. En concreto, se destacan los siguientes aprendizajes:

- Comprensión profunda del desarrollo web full-stack, integrando de forma efectiva *backend* (FastAPI, SQLAlchemy, PostgreSQL) y *frontend* (React, Vite, Tailwind CSS).
- Mejora en la planificación y organización del trabajo, especialmente a través de metodologías ágiles y el uso de sprints para avanzar iterativamente.
- Experiencia real con herramientas de despliegue en la nube (Render), control de versiones con Git y GitHub, y servicios complementarios como Cloudinary o FastAPI-Mail.
- Enfrentamiento a problemas reales de integración, depuración de errores, y adaptación continua del diseño a nuevas necesidades o descubrimientos durante el desarrollo.
- Valoración del testing como proceso clave para asegurar la calidad y estabilidad del sistema.

A nivel personal, este proyecto también ha demostrado la importancia de la perseverancia, la gestión del tiempo y la capacidad de autogestión para afrontar un proyecto de gran envergadura en solitario. Además, ha sido muy gratificante poder aplicar los conocimientos en un contexto práctico, con un caso de uso real y potencialmente útil para una entidad deportiva.

8.3. Trabajos futuros

Si bien la aplicación desarrollada cumple satisfactoriamente con los objetivos iniciales, existen múltiples vías de mejora y expansión en futuros

trabajos. Algunas de las propuestas más relevantes son:

- **Implementación de un sistema de roles y permisos:** aunque actualmente se diferencian usuarios por tipo, se puede profundizar en la lógica de autorización para restringir accesos de forma más granular.
- **Mejoras en la experiencia de usuario:** incorporar animaciones con Framer Motion, optimización de tiempos de carga y mayor personalización del diseño.
- **Centralización de servicios y peticiones en el *frontend*:** crear una capa de servicios unificada que maneje todas las llamadas a la API y permita una gestión global de errores y estados.
- **Panel de administración más avanzado:** con estadísticas, gráficos y funcionalidades de gestión más completas para administradores.
- **Desarrollo de una app móvil:** que permita a los aficionados interactuar desde sus smartphones con una experiencia aún más adaptada.
- **Internacionalización:** incluir soporte multilingüe para llegar a un público más amplio.
- **Sistema de migraciones y testing más robusto:** implementar automatización con Alembic y Pytest para mejorar el ciclo de vida del software.

Estas mejoras permitirían no solo ampliar las funcionalidades del sistema, sino también profesionalizar aún más la solución y convertirla en un producto totalmente adaptable a otros clubes o entidades similares.

Bibliografía

- [1] Mike Bayer. Ssqlalchemy documentation, 2024.
- [2] Kent Beck. *Test Driven Development: By Example*. Addison-Wesley, 2003.
- [3] Tim Berners-Lee, Mark Fischetti, and Michael L. Dertouzos. *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by Its Inventor*. HarperOne, 1999.
- [4] Bob Boiko. *Content Management Bible*. Wiley, 2005.
- [5] David Rowe Brett Hutchins. *Sport Beyond Television: The Internet, Digital Media and the Rise of Networked Media Sport*. Routledge, 2012.
- [6] Scott; Parker Craig Burgess, Stephen; Bingley. The value of local sporting clubs' websites. *ScienceDirect*, 2021.
- [7] Cloudinary Ltd. Cloudinary - image and video management, 2024.
- [8] Samuel Colvin. Pydantic - data validation and settings management using python type annotations, 2024.
- [9] Mozilla Corporation. Css: Cascading style sheets - mdn web docs. <https://developer.mozilla.org/en-US/docs/Web/CSS>, 2024.
- [10] Mozilla Corporation. Html: Hypertext markup language - mdn web docs. <https://developer.mozilla.org/en-US/docs/Web/HTML>, 2024.
- [11] Brian K. Jones David Beazley. *Python Cookbook: Recipes for Mastering Python 3*. O'Reilly Media, 2013.
- [12] Eric Freeman and Elisabeth Freeman. *Head First Design Patterns*. O'Reilly Media, 2007.
- [13] Steve Freeman and Nat Pryce. *Growing Object-Oriented Software, Guided by Tests*. Addison-Wesley, 2010.

- [14] James Gillies and Robert Cailliau. *How the Web Was Born: The Story of the World Wide Web*. Oxford University Press, 2000.
- [15] Anurag Goel. Render - the modern cloud provider for all your apps and websites. <https://render.com/>, 2024. Sitio oficial de Render.com.
- [16] Miguel Grinberg. *Flask Web Development: Developing Web Applications with Python*. O'Reilly Media, 2018.
- [17] Eugenio; Fernández Luna Juan Manuel; García Sánchez Pablo; Noguera García Manuel; Rodríguez Fórtiz María José; Romero Zaliz Rocio Celeste Guillén Perales, Alberto; Martínez Cámara. *Cómo escribir la memoria de tu TFG del Grado en Ingeniería Informática y presentarlo sin morir en el intento*. Universidad de Granada, 2025.
- [18] D. Hardt. The oauth 2.0 authorization framework. <https://datatracker.ietf.org/doc/html/rfc6749>, 2012.
- [19] Internet Engineering Task Force (IETF). Json web token, 2010.
- [20] Eric Matthes. *Python Crash Course: A Hands-On, Project-Based Introduction to Programming*. No Starch Press, 2019.
- [21] Microsoft Corporation. Playwright - fast and reliable end-to-end testing, 2024.
- [22] Sebastián Ramírez Montaña. Fastapi - titulo oficial. <https://fastapi.tiangolo.com/>, 2024. Documentación oficial.
- [23] Jakob Nielsen. *Designing Web Usability: The Practice of Simplicity*. New Riders, 1999.
- [24] A. O'Cass and J. Carlson. Examining the effects of website-induced flow in professional sporting team websites. *Internet Research*, 2010.
- [25] Postman, Inc. Newman - postman's cli collection runner, 2024.
- [26] Postman, Inc. Postman api platform, 2024.
- [27] Precognis. Modelo vista controlador (mvc). <https://www.precognis.com/blog/modelo-vista-controlador/>.
- [28] Roger S. Pressman and Bruce R. Maxim. *Software Engineering: A Practitioner's Approach*. McGraw-Hill Education, 8th edition, 2014.
- [29] Michael Stonebrake. PostgreSQL: The world's most advanced open source relational database. <https://www.postgresql.org/>. Último acceso: 2 junio 2025.

-
- [30] Linus Torvalds. Introduction to git and github. <https://docs.github.com/en/get-started>, 2024. GitHub Documentation.
 - [31] van You. Vite – next generation frontend tooling. <https://vitejs.dev/>, 2024. Documentación oficial de Vite.
 - [32] Jordan Walke. React router – declarative routing for react. <https://reactrouter.com/en/main>, 2024. React Router Documentation.
 - [33] Adam Wathan y Steve Schoger. Tailwind css - rapidly build modern websites without ever leaving your html. <https://tailwindcss.com/>, 2024. Sitio oficial de Tailwind CSS.