ESCUELA DE INGENIERÍA INFORMÁTICA (SG)

Grado en Ingeniería Informática de Servicios y Aplicaciones

LinkByBall: la red de contactos para el fútbol español

Autor: Adrián Contreras Sanz
Tutor: Fernando Díaz Gómez

Agradecimientos

Quiero expresar mi más sincero agradecimiento a mis padres, hermanos y pareja, por su apoyo constante y su confianza en mí durante todo este proceso académico.

A mi tutor del TFG, Fernando Díaz, por su guía, paciencia y dedicación, que han sido clave para la realización de este trabajo.

También deseo agradecer a la Universidad de Valladolid, al Campus María Zambrano de Segovia y a todos los profesores que me acompañaron en este camino, por proporcionarme una formación de calidad y los recursos necesarios para alcanzar mis objetivos.

Finalmente, a mis amigos y compañeros de trabajo, por estar a mi lado en los momentos difíciles y ayudarme con sus respectivas experiencias personales a realizar este proyecto.

Resumen

Cada año, con la llegada del verano, los equipos de fútbol concluyen la temporada, generando gran incertidumbre entre todos los involucrados sobre lo que les deparará el futuro en los meses estivales. Mientras algunos cuentan con el respaldo de representantes que gestionan su próximo destino, la mayoría se enfrenta a la necesidad de contactar directamente con los clubes o esperar con incertidumbre a que suene el teléfono con una propuesta atractiva de un club que pueda ofrecer una buena categoría, acompañada de buenas condiciones.

El objetivo principal de este proyecto es desarrollar una aplicación web que sirva como puente para conectar clubes de fútbol de categorías amateur y semiprofesional con jugadores y entrenadores que estén en busca de un nuevo equipo. La plataforma tiene como propósito facilitar la interacción entre ambas partes, optimizando los procesos de búsqueda, selección y contratación.

A través de la aplicación, los clubes podrán publicar ofertas específicas para cubrir vacantes en su equipo, detallando el perfil deseado para cada posición, ya sea como futbolista o miembro del cuerpo técnico. Por su parte, los jugadores y entrenadores tendrán la posibilidad de crear perfiles profesionales completos. Cuando una oferta resulte atractiva, los usuarios podrán aplicar a ella, dejando al club la tarea de evaluar si encajan con las necesidades del puesto.

Palabras clave: Aplicación web, fútbol, conectar, ofertas, jugadores, cuerpo técnico, clubes, puesto, categorías.

Abstract

Every year, with the arrival of summer, football teams conclude the season, generating great uncertainty among all those involved about what the future holds during the summer months. While some rely on agents to manage their next destination, most are left with the task of contacting clubs directly or waiting with uncertainty for the phone to ring with an attractive offer from a club that can provide a good category accompanied by favorable conditions.

The main objective of this project is to develop a web application that acts as a bridge to connect amateur and semi-professional football clubs with players and coaches looking for a new team. The platform aims to facilitate interaction between both parties, optimizing the processes of searching, selection, and recruitment.

Through the application, clubs will be able to post specific offers to fill vacancies in their teams, detailing the desired profile for each position, whether it is a football player or a member of the coaching staff. Players and coaches, in turn, will have the opportunity to create complete professional profiles. When an offer appears attractive, users will be able to apply, leaving the club to evaluate whether they meet the requirements for the position.

Keywords: Web application, football, connect, offers, players, coaching staff, clubs, position, categories.

Índice general

Índice de imágenes

Índice de tablas

1. Introducción

Para los futbolistas de categorías semi profesionales o amateurs, la llegada del final de temporada puede ser un auténtico quebradero de cabeza. Hoy en día, encontrar oportunidades en el mundo del fútbol sigue siendo un proceso complicado y poco accesible. Muchos futbolistas y entrenadores dependen de contactos personales, representantes o incluso la suerte para contactar con clubes que busquen nuevos talentos. Por otro lado, los clubes tienen dificultades para encontrar candidatos adecuados sin pasar por procesos largos e ineficientes, lo que provoca retrasos entre ambas partes, futbolistas y clubes, a la hora de formalizar la plantilla.

La mayoría de las plataformas actuales están enfocadas en el fútbol profesional de alto nivel, dando de lado a miles de jugadores y entrenadores que buscan oportunidades en categorías inferiores. Además, la comunicación entre clubes y candidatos sigue siendo poco estructurada. Muchas veces para poder conseguir el contacto de alguien que pertenece a un club específico, tienes que hacer varias llamadas a personas que van redirigiendo hasta la persona de contacto final.

Para solucionar esto, LinkByBall nace como una plataforma que funciona como un punto de encuentro entre futbolistas, entrenadores y clubes. La idea es que cualquier persona que quiera crecer en el mundo del fútbol tenga un lugar donde pueda mostrar su perfil, postularse a vacantes y conectarse con clubes sin necesidad de intermediarios.

Con LinkByBall, los clubes pueden publicar vacantes de forma rápida y sencilla, mientras que los futbolistas y entrenadores pueden aplicar directamente y comunicarse con ellos a través de un chat integrado. Además, cuenta con un sistema de favoritos y notificaciones que permite a los usuarios estar al tanto de cambios en los perfiles de interés.

Esta plataforma está pensada para:

- Futbolistas que buscan equipos en cualquier categoría.
- Entrenadores que quieren encontrar nuevas oportunidades laborales.
- Clubes deportivos que necesitan encontrar jugadores o entrenadores de manera eficiente.

El objetivo de LinkByBall es hacer más fácil y accesible el proceso de búsqueda de oportunidades en el fútbol, eliminando barreras y facilitando la conexión entre personas con un mismo propósito: hacer crecer sus carreras en este deporte.

1.1 Motivación

Desde que tengo memoria, el fútbol ha sido mi mayor pasión y una parte fundamental de mi vida. Desde bien pequeño he jugado en clubes con la finalidad de competir y rendir al más alto nivel, viviendo de primera mano lo difícil que puede ser encontrar la mejor

oportunidad que combinara una buena categoría junto con la importancia de tener un hueco en el equipo y no estar la mitad de los partidos en el banquillo o en la grada. He visto a compañeros con mucho talento quedarse sin equipo simplemente porque no tenían los contactos adecuados, por el contrario, he visto a compañeros con menos talento alcanzar clubes con categorías más altas simplemente por saber a quién llamar o conocer a alguien que les pudiera dar esa oportunidad. También he podido conocer a entrenadores con experiencia y buena formación, sin opciones claras de seguir escalando, y clubes que por no saber dónde buscar, se conformaban con lo que tenían.

En mi propia experiencia, encontrar oportunidades ha sido más cuestión de conocer a la persona correcta en el momento adecuado, que de contar con un sistema transparente y accesible. En un mundo donde la tecnología ha facilitado la búsqueda de empleo en casi cualquier sector, el fútbol sigue funcionando de manera demasiado tradicional, con recomendaciones de boca en boca, agentes o redes de contactos cerradas.

Este proyecto surge de una necesidad que he vivido en primera persona y que sé que muchos jugadores, entrenadores y clubes también enfrentan. Si LinkedIn ha cambiado la forma en la que las empresas encuentran talento, ¿por qué no hacer lo mismo con el fútbol? LinkByBall busca ser esa herramienta que conecte a las personas adecuadas en el momento adecuado, abriendo puertas y permitiendo que el talento sea el verdadero protagonista.

1.2 Objetivos y alcance del proyecto

LinkByBall es una plataforma digital diseñada para conectar futbolistas, miembros del cuerpo técnico y clubes en un entorno profesional y accesible. Su propósito es facilitar la búsqueda de oportunidades en el mundo del fútbol que agilicen la publicación y postulación a vacantes, la gestión de nuevos contactos y la comunicación entre ellos.

El desarrollo de este proyecto abarcará los siguientes aspectos:

- Un *frontend* desarrollado con angular, con una interfaz intuitiva y vistosa para generar la atracción de nuevos usuarios.
- Un *backend* robusto construido con Node.js y Express, asegurando una gestión eficiente de la información junto con una comunicación segura a la base de datos.
- Una base de datos en MongoDB que sirva para almacenar los datos sobre los diferentes perfiles de usuarios, vacantes, solicitudes, mensajes, avisos y conversaciones.
- Autenticación segura a través del uso de JWT (*JSON Web Tokens*), utilizado para restringir la información dependiendo de qué tipo de usuario ha iniciado sesión en la aplicación.
- Almacenamiento de imágenes en Cloudinary para la gestión de las imágenes de perfil.

La primera versión de LinkByBall incluirá las siguientes características principales:

- Habrá cuatro tipos de perfiles de usuario diferentes en toda la aplicación:

    - **Futbolistas**: aquellos usuarios que se registren como futbolistas deberán completar su formulario correspondiente indicando su nombre, apellidos, correo electrónico, contraseña, edad, posición en la que juega (portero, central, lateral, interior, mediocentro defensivo, media punta, carrilero, extremo, delantero centro), pierna dominante y país de nacimiento. Como campos opcionales estará el club y la categoría en la que juega en el momento de registrase y una imagen de perfil.
    - **Entrenadores**: este tipo de usuarios deberán indicar el nombre, apellidos, correo electrónico, contraseña, edad, país de nacimiento y su especialidad dentro de un cuerpo técnico (primer entrenador, segundo entrenador, preparador físico, entrenador de porteros, analista táctico, fisioterapeuta, utillero, nutricionista, psicólogo deportivo). Como campos opcionales estará el número de teléfono, el club y categoría en el momento del registro y una imagen de perfil.
    - **Clubes**: los clubes deberán cumplimentar el formulario con el nombre del club, la categoría que disputa, correo electrónico, teléfono, comunidad autónoma, provincia, contraseña e imagen con el logotipo del club. Para que los clubes aparezcan y puedan realizar las operaciones de gestión que ofrece la herramienta deberán ser previamente autorizados por un usuario de tipo administrador.
    - **Administradores**: estos usuarios no deberán realizar ningún tipo de registro, serán introducidos manualmente en la base de datos con un correo electrónico y una contraseña. Su labor es autorizar y desautorizar a aquellos clubes de nuevo ingreso en la aplicación.
- En cuanto a las funcionalidades de usuario, la aplicación contará con una **gestión de perfil**, permitiendo a los usuarios modificar datos específicos o eliminar su cuenta de la plataforma. Además, dispondrá de un **buscador avanzado**, diseñado para filtrar perfiles según el tipo de usuario y otros criterios específicos, mejorando la experiencia de búsqueda y facilitando la localización de perfiles con características comunes.

  Para fomentar la interacción entre los usuarios, la plataforma incorporará un sistema de mensajería en tiempo real, permitiendo el envío y recepción de mensajes instantáneos a través de un **chat integrado**. Asimismo, los usuarios podrán **guardar perfiles como favoritos**, disponiendo de un apartado específico donde podrán acceder rápidamente a aquellos contactos de mayor interés.

  En lo que respecta a la gestión de vacantes, la aplicación ofrecerá un **sistema de seguimiento de solicitudes**, donde los usuarios podrán visualizar el estado de sus postulaciones. Cada vez que se produzca un cambio en el estado de una solicitud, se enviará una **notificación automática** a los usuarios involucrados, garantizando así un seguimiento eficiente de las oportunidades disponibles.
- Los futbolistas y entrenadores compartirán la mayoría de las funcionalidades dentro de la plataforma. Ambos contarán con un ***dashboard* personal**, donde podrán visualizar información relevante de su perfil, incluyendo:
    - **Nacionalidad**,
    - **Club actual** (si aplica),

    - **Edad**,
    - **Posiciones o puestos desempeñados**,
    - **Vacantes recomendadas** en función de su desempeño,
    - **Últimas tres notificaciones recibidas**,
    - **Solicitudes enviadas y su estado**.

  Además, tendrán la posibilidad de **postularse a cualquier vacante disponible**, realizando una búsqueda personalizada que se ajuste a sus necesidades y preferencias. Dentro de este apartado, podrán **consultar el estado de sus solicitudes**, lo que les permitirá conocer en tiempo real si han sido aceptadas, rechazadas o se encuentran en proceso de revisión.
- Los clubes contarán con un ***dashboard* general** que incluirá estadísticas clave para la gestión de su plantilla y vacantes. En este espacio podrán visualizar:
    - **Número de vacantes activas**,
    - **Número de solicitudes recibidas**,
    - **Número de miembros en plantilla**,
    - **Últimas seis vacantes publicadas**,
    - **Últimos avisos recibidos**,
    - **Futbolistas y entrenadores actuales en el club**.
    - Asimismo, dispondrán de un apartado dedicado a la **gestión de vacantes**, donde podrán **visualizar, editar y añadir nuevas oportunidades** mediante un formulario que incluirá:
    - **Tipo de usuario** (futbolista o entrenador),
    - **Puesto ofertado**,
    - **Salario (opcional)**,
    - **Breve descripción de la vacante**.

  Dentro de cada vacante, los clubes podrán acceder a todas las **solicitudes recibidas** y gestionar su estado mediante cuatro categorías:
    - **Pendiente**: cuando la solicitud no ha sido revisada,
    - **En proceso**: cuando el club está evaluando al candidato,
    - **Aceptada**,
    - **Rechazada**.

  Este sistema permitirá a los clubes gestionar sus procesos de selección de manera estructurada y eficiente, optimizando la captación de talento y asegurando una comunicación clara con los postulantes.
- En cuanto a la alimentación de la base de datos, se hará una carga de información no verídica para presentar el prototipo de la aplicación. Ya más adelante se empezará a alimentar de datos reales.

Para futuras versiones:

- El sistema deberá estar organizado para que más adelante se piense en meter LinkByBall en las diferentes tiendas de aplicaciones como AppleStore o Play Store. Esto se podrá realizar fácilmente mediante herramientas como Ionic o Capacitor, las cuales soportan angular para el desarrollo del apk.
- Almacenar más información sobre diferentes usuarios como por ejemplo partidos jugados, partidos ganados/empatados/perdidos, partidos como titular/suplente, número de lesiones sufridas, número de fichajes por año, etc. Todo ello para después crear algoritmos que deduzcan el futuro comportamiento de un futbolista o entrenador si se realiza su fichaje, algoritmos que traten de encontrar un club que se ajuste a tus características personales, etc.

1.3 Organización del documento

Este documento se organiza en diferentes secciones, cada una enfocada en un aspecto clave del proyecto, con el objetivo de proporcionar una visión clara y detallada de su desarrollo y ejecución. A continuación, se describe el contenido de cada capítulo:

Capítulo 1: Introducción

Este capítulo presenta el contexto y la motivación del proyecto, explicando las razones que impulsaron su desarrollo. Además, se establecen los objetivos principales y el alcance del trabajo, definiendo los límites y expectativas del proyecto.

Sección 2: Estado del Arte

Se realiza un análisis de las soluciones existentes en el ámbito del proyecto, comparando diferentes alternativas para destacar las características innovadoras de esta propuesta. También se describen las herramientas y tecnologías utilizadas durante el desarrollo, proporcionando un marco de referencia sobre el entorno técnico en el que se llevó a cabo.

Capítulo 3: Gestión del Proyecto

En este apartado se detalla la metodología empleada para la planificación y ejecución del proyecto, incluyendo cronogramas, distribución de tareas y estrategias de trabajo. También se incorpora un análisis de viabilidad financiera, con una estimación del presupuesto y recursos necesarios. Finalmente, se realiza una evaluación del progreso, comparando los resultados obtenidos con los objetivos planteados.

Capítulo 4: Análisis

Se describen los distintos actores que interactúan con la plataforma, incluyendo usuarios y otras entidades relevantes. Además, se especifican los requisitos funcionales y no funcionales del sistema, estableciendo las características esenciales que debe cumplir. También se analizan las restricciones y limitaciones del proyecto, así como la información gestionada por la plataforma.

Capítulo 5: Diseño

Este capítulo aborda la estructura conceptual del sistema, explicando tanto su arquitectura lógica como su configuración física. Se incluyen diagramas de secuencia que ilustran la interacción entre los distintos componentes, el diseño de la interfaz de usuario y la organización de los datos en la base de datos.

Capítulo 6: Implementación

Aquí se detalla la estructura del código y la organización de los diferentes módulos del sistema. Se describe el proceso técnico de implementación y se explican las decisiones adoptadas durante el desarrollo para convertir el diseño en una solución funcional.

Capítulo 7: Pruebas

Este apartado recoge las estrategias de prueba aplicadas al sistema para garantizar su correcto funcionamiento. Se explican las pruebas de caja negra, que evalúan el comportamiento de la aplicación sin conocer su estructura interna, así como otros procedimientos adicionales utilizados para verificar la calidad del producto final.

Capítulo 8: Documentación de usuario

Se incluye el manual de usuario, con instrucciones detalladas sobre el uso de la plataforma, y el manual de despliegue, que proporciona las directrices necesarias para la instalación y configuración del sistema.

Capítulo 9: Conclusiones y trabajo futuro

En el último capítulo, se presenta una reflexión sobre los resultados alcanzados y el proceso de desarrollo del proyecto. Además, se sugieren posibles mejoras y líneas de trabajo futuro que permitirían ampliar o perfeccionar la plataforma en versiones posteriores.

2. Estado del arte

En este capítulo se realiza un análisis del estado actual de las soluciones existentes en el ámbito de la búsqueda de oportunidades deportivas, con especial énfasis en plataformas similares a **LinkByBall**. El objetivo es identificar las fortalezas y limitaciones de las alternativas disponibles en el mercado, permitiendo justificar la necesidad y relevancia del presente proyecto.

A lo largo de este apartado, se compararán diferentes herramientas y aplicaciones utilizadas en el sector del fútbol y el reclutamiento deportivo, evaluando aspectos como su funcionalidad, accesibilidad y grado de especialización. Se analizarán plataformas de empleo tradicionales, redes profesionales y otros sistemas que faciliten la conexión entre jugadores, entrenadores y clubes, identificando qué aspectos aún presentan oportunidades de mejora.

Además, este capítulo describe el **entorno de trabajo** en el que se desarrolla la plataforma, detallando las herramientas y tecnologías seleccionadas para su implementación. Se presentarán los criterios que han guiado la elección de ***frameworks,* lenguajes de programación, bases de datos y servicios en la nube**, justificando las razones por las cuales me he decantado por cada una de ellas.

El análisis del estado del arte y del entorno de trabajo no solo contextualiza el proyecto dentro de un marco tecnológico y competitivo, sino que también permite comprender las ventajas diferenciales que aporta **LinkByBall** frente a las soluciones ya existentes.

2.1 Propuestas software existentes

A continuación, se detallarán varios sistemas con softwares similares o con funcionalidades parecidas, ya que esta es una idea que no se ha desarrollado demasiadas veces y no hay aplicaciones del todo iguales.

2.1.1 LinkedIn

LinkedIn es actualmente la red profesional más grande del mundo, con más de 900 millones de usuarios registrados en más de 200 países [1]. Su principal propósito es facilitar la conexión entre profesionales y empresas, permitiendo la creación de perfiles detallados con información sobre experiencia laboral, educación, habilidades y certificaciones.

A lo largo de los años, LinkedIn ha evolucionado hacia una plataforma de reclutamiento y *networking* que ofrece herramientas avanzadas tanto para candidatos como para empresas. Funcionalidades como la **búsqueda de empleo, publicación de vacantes, recomendaciones personalizadas y herramientas de mensajería profesional** han convertido a LinkedIn en un estándar dentro del ámbito del empleo digital [1], [2].

Si bien LinkedIn ha revolucionado la forma en que los profesionales buscan empleo, su enfoque sigue estando mayormente orientado a sectores tradicionales como la ingeniería, negocios y tecnología. En el ámbito deportivo, y en especial en el **fútbol**, no existe una plataforma específica con el mismo nivel de alcance y especialización.

Actualmente, la búsqueda de oportunidades para futbolistas y entrenadores sigue dependiendo de **contactos personales, agentes e intermediarios**, lo que genera un acceso desigual a oportunidades y limita la visibilidad del talento emergente. **LinkByBall** toma como referencia el modelo de LinkedIn, pero lo adapta a las necesidades del fútbol, ofreciendo una plataforma especializada donde los jugadores y entrenadores pueden conectarse directamente con clubes y oportunidades relevantes.

2.1.2 FutbolJobs

Es una plataforma de empleo especializada en el ámbito del fútbol. Su propósito es conectar a **jugadores, entrenadores, preparadores físicos, analistas y otros profesionales del deporte** con clubes y entidades deportivas que buscan talento [3].

A diferencia de las plataformas generalistas de empleo, FutbolJobs centra su actividad exclusivamente en ofertas laborales del sector futbolístico, permitiendo a los usuarios **registrarse, crear un perfil y postularse a ofertas de trabajo** dentro de clubes y academias deportivas. La plataforma ha logrado consolidarse como un referente en la búsqueda de empleo dentro del ámbito del fútbol, facilitando la contratación de profesionales a nivel nacional e internacional [4].

Uno de los aspectos diferenciadores de **LinkByBall** frente a **FutbolJobs** es la posibilidad de realizar **seguimiento en tiempo real de las solicitudes**, recibir **notificaciones automáticas sobre cambios en vacantes** y establecer una **red de contactos** dentro del sector.

Además, el modelo de negocio de FutbolJobs se basa en **suscripciones de pago**, lo que puede limitar el acceso a jugadores y entrenadores que no puedan permitirse una membresía premium. **LinkByBall** se presenta como una alternativa más accesible, en la que los usuarios pueden interactuar y postularse sin necesidad de pagar por servicios básicos.

2.1.3 Jobs in Football

Jobs in Football es una plataforma de empleo especializada en la industria del fútbol. Su objetivo es conectar a jugadores, entrenadores, analistas, directores deportivos y otros profesionales del sector con oportunidades laborales en clubes, academias y organizaciones deportivas a nivel internacional[5].

A diferencia de los portales de empleo convencionales, Jobs in Football se enfoca exclusivamente en el ámbito futbolístico, ofreciendo una base de datos con vacantes en **dirección técnica, *scouting*, marketing deportivo y gestión de clubes**, entre otros. La

plataforma proporciona acceso a recursos educativos y consejos para mejorar la empleabilidad de los aspirantes[6].

Uno de los principales desafíos de Jobs in Football es su modelo de acceso, en el que algunos contenidos y vacantes están restringidos a suscriptores premium, lo que puede limitar el acceso a ciertas oportunidades laborales.

Por otro lado, la falta de interacción directa entre los usuarios en Jobs in Football impide que jugadores, entrenadores y clubes puedan comunicarse de manera ágil y establecer relaciones a largo plazo. **LinkByBall** resuelve esta limitación mediante un sistema de chat en tiempo real, un apartado de favoritos y notificaciones automáticas, ofreciendo una experiencia más fluida y colaborativa.

2.2 Comparativa

	LinkByBall	LinkedIn	FutbolJobs	JobsInFootball
Aplicación web	X	X	X	X
Aplicación en marketplace		X	X	X
Enfocado al sector del deporte	X		X	X
Varios Idiomas		X		X
Funcionalidades premium		X	X	X
Chat integrado	X	X		
Control y gestión de perfiles registrados	X	X	X	X
Seguimiento de favoritos	X	X		X
Búsqueda por filtros	X	X	X	X
Recomendaciones personalizadas	X	X	X	X

Tabla 1: Comparativa softwares similares

2.3 Herramientas y tecnologías utilizadas

A continuación, se desarrollará un listado con todas las herramientas, tecnologías y softwares utilizados durante el desarrollo de esta aplicación.

2.3.1 Microsoft Word

Microsoft Word es un software de procesamiento de texto desarrollado por Microsoft, ampliamente utilizado en entornos académicos, profesionales y personales para la creación, edición y formato de documentos. Como herramienta principal en la elaboración de esta memoria de Trabajo de Fin de Grado (TFG), Word ofrece funcionalidades avanzadas que facilitan la redacción, organización y presentación de contenidos. Entre sus características más relevantes se encuentran la posibilidad de estructurar documentos mediante encabezados, tablas de contenido automáticas, inserción de citas y bibliografías, así como herramientas de revisión ortográfica y gramatical que garantizan la calidad del texto [7]. Además, su compatibilidad con otros programas de Microsoft Office, como Excel o PowerPoint, y su capacidad para exportar documentos en formatos como PDF, lo convierten en una opción versátil y eficiente para la gestión de proyectos académicos.

Ilustración 1: Microsoft word

2.3.2 Mendeley

Mendeley es un gestor de referencias bibliográficas desarrollado por Elsevier que facilita la organización, almacenamiento y citación de fuentes académicas [8] en la elaboración de este Trabajo de Fin de Grado (TFG). Utilizado para gestionar la bibliografía, permite importar referencias desde bases de datos, generar citas automáticas en diversos estilos (como IEEE) y crear listas de referencias, optimizando el proceso de documentación académica gracias a su integración con procesadores de texto como Microsoft Word y su funcionalidad de sincronización en la nube.

Ilustración 2: Mendeley

2.3.3 Github

GitHub es una plataforma en línea diseñada para el control de versiones y el desarrollo colaborativo de software, basada en el sistema de control de versiones Git. Permite a múltiples usuarios trabajar simultáneamente en proyectos, gestionar cambios

en el código fuente y realizar un seguimiento detallado de las modificaciones a través de repositorios. Entre sus características principales se incluyen la gestión de repositorios públicos y privados, herramientas de colaboración como solicitudes de extracción (pull requests) y revisiones de código, integración con servicios de despliegue continuo (CI/CD), y un entorno social que fomenta la contribución a proyectos de código abierto. GitHub es ampliamente utilizado tanto por desarrolladores individuales como por empresas y comunidades académicas para alojar, compartir y documentar proyectos de software [9].

Ilustración 3: Github

2.3.4 Visual Studio Code

Visual Studio Code (VS Code) es un editor de código fuente gratuito y de código abierto desarrollado por Microsoft, lanzado en 2015. Está diseñado para ser altamente personalizable, ligero y compatible con una amplia variedad de lenguajes de programación. VS Code utiliza un modelo basado en extensiones, lo que permite a los usuarios ampliar sus funcionalidades mediante la instalación de complementos desde un marketplace integrado, como soporte para lenguajes específicos, herramientas de depuración o integración con sistemas de control de versiones como Git. Entre sus características destacadas se incluyen el resaltado de sintaxis, autocompletado inteligente (IntelliSense), capacidades de refactorización y un terminal integrado.

VS Code se ha convertido en una herramienta popular tanto para desarrolladores profesionales como para estudiantes y académicos debido a su facilidad de uso, rendimiento y soporte para flujos de trabajo modernos, como el desarrollo colaborativo a través de extensiones como Live Share. Además, su integración con plataformas como GitHub lo hace ideal para proyectos que requieren colaboración en tiempo real y control de versiones. En el ámbito académico, VS Code es ampliamente utilizado para la enseñanza de programación, la edición de scripts científicos y la gestión de proyectos de software. Gracias a las extensiones, VS Code se puede customizar con iconos, colores para diferenciar mejor el código y ayudas de autocompletado a la hora del desarrollo [10].

Ilustración 4: Visual Studio Code

2.3.5 Node js

Node.js es un entorno de ejecución de JavaScript de código abierto, basado en el motor V8 de Google Chrome, que permite ejecutar código JavaScript en el lado del servidor. Lanzado en 2009 por Ryan Dahl, Node.js está diseñado para construir aplicaciones web escalables y de alto rendimiento, utilizando un modelo de entrada/salida asíncrono y no bloqueante basado en eventos. Esto lo hace especialmente adecuado para aplicaciones en tiempo real, como chats, servicios de *streaming* o APIs RESTful. Node.js incluye un gestor de paquetes integrado, npm (*Node Package Manager*), que facilita la instalación y gestión de bibliotecas y dependencias, lo que ha contribuido a su amplio ecosistema de módulos de código abierto.

Entre sus características clave se encuentran su arquitectura ligera, su capacidad para manejar múltiples conexiones simultáneamente y su compatibilidad con JavaScript, un lenguaje ya familiar para los desarrolladores web. Node.js ha transformado el desarrollo *backend* al permitir que JavaScript se utilice tanto en el cliente como en el servidor, simplificando la creación de aplicaciones *full-stack*. En el ámbito académico y profesional, Node.js es utilizado para proyectos que requieren alta concurrencia, así como en la enseñanza de tecnologías modernas de desarrollo web [11].

Ilustración 5: Node js

2.3.6 Express

Express.js, comúnmente conocido como Express, es un *framework* minimalista y flexible de código abierto para Node.js, diseñado para facilitar la creación de aplicaciones web y APIs. Lanzado en 2010 por TJ Holowaychuk, Express proporciona una capa de abstracción sobre las funcionalidades nativas de Node.js, simplificando la gestión de rutas, middleware y respuestas HTTP. Su diseño ligero permite a los desarrolladores estructurar aplicaciones backend de manera eficiente, mientras que su sistema de middleware modular soporta funcionalidades como autenticación, manejo de sesiones y procesamiento de solicitudes. Express es ampliamente reconocido por su simplicidad y

su enfoque en la rapidez de desarrollo, lo que lo hace ideal para proyectos de pequeña y gran escala [12].

Ilustración 6: Express js

2.3.7 Angular

Angular es un *framework* de código abierto desarrollado por Google para la creación de aplicaciones web dinámicas y de una sola página (SPA, *Single Page Applications*). Lanzado inicialmente como AngularJS en 2010, fue completamente reescrito y relanzado como Angular (versión 2 y superiores) en 2016, utilizando TypeScript como lenguaje principal. Angular ofrece un enfoque basado en componentes, donde la interfaz de usuario se construye mediante bloques reutilizables y modulares, facilitando la escalabilidad y el mantenimiento del código. Entre sus características clave se incluyen el enlace de datos bidireccional (en AngularJS) y unidireccional (en versiones posteriores), la inyección de dependencias, un sistema de plantillas avanzado y herramientas integradas como Angular CLI para la generación y gestión de proyectos [13].

Ilustración 7: Angular

2.3.8 TypeScript (TS)

TypeScript es un lenguaje de programación de código abierto desarrollado por Microsoft, lanzado en 2012 como una extensión de JavaScript que incorpora tipado estático opcional. TypeScript se compila a JavaScript puro, lo que lo hace compatible con cualquier entorno donde JavaScript pueda ejecutarse, como navegadores web o Node.js. Su principal objetivo es mejorar la escalabilidad y mantenibilidad del código en proyectos grandes mediante la introducción de tipos, interfaces y otras características de programación orientada a objetos, como clases y módulos. Esto permite a los desarrolladores detectar errores en tiempo de compilación en lugar de en tiempo de ejecución, lo que aumenta la robustez de las aplicaciones.

TypeScript ha ganado gran popularidad en el desarrollo web moderno, especialmente en *frameworks* como Angular, que lo adoptó como lenguaje principal

desde su versión 2. También es ampliamente utilizado en proyectos con React, Vue.js y Node.js gracias a su flexibilidad y herramientas como el compilador TSC (*TypeScript Compiler*) y el soporte en editores como Visual Studio Code. En el ámbito académico, TypeScript se emplea para enseñar conceptos avanzados de programación y para desarrollar proyectos que requieren un alto grado de fiabilidad y estructura [14].

Ilustración 8: TypeScript

2.3.9 Figma

Figma es una herramienta de diseño colaborativo que permite a los usuarios crear interfaces de usuario, prototipos interactivos y diagramas como wireframes, flujos de usuario y mapas de sitio, entre otros. Funciona completamente en línea a través de un navegador web, aunque también dispone de una aplicación de escritorio. Se integra con diversas plataformas de almacenamiento en la nube y herramientas de gestión de proyectos como Google Drive, Slack y Jira. Su interfaz intuitiva basada en arrastrar y soltar, junto con una amplia biblioteca de componentes reutilizables y funciones avanzadas como el diseño en tiempo real con múltiples colaboradores, la hacen ideal tanto para diseñadores principiantes como para profesionales. Figma opera bajo un modelo basado en la nube, lo que permite la sincronización instantánea de cambios y facilita el trabajo en equipo sin necesidad de instalaciones locales, priorizando la accesibilidad y la eficiencia.

Ilustración 9: Figma

2.3.10 MongoDB

MongoDB es un sistema de base de datos NoSQL de código abierto, orientado a documentos, diseñado para manejar grandes volúmenes de datos no estructurados o semiestructurados. Lanzado en 2009 por 10gen (ahora MongoDB Inc.), utiliza un modelo de almacenamiento basado en documentos BSON (Binary JSON), que permite representar datos en estructuras flexibles similares a JSON, en lugar de tablas relacionales como en las bases de datos SQL tradicionales. Esta flexibilidad lo hace ideal para

aplicaciones modernas como plataformas web, sistemas de análisis en tiempo real y aplicaciones móviles, donde los esquemas de datos pueden cambiar con frecuencia [15].

Ilustración 10: MongoDB

2.3.11 Cloudinary

Cloudinary es una plataforma en la nube diseñada para la gestión, transformación y entrega de activos multimedia, como imágenes, videos y otros archivos digitales. Fundada en 2011, ofrece un conjunto de herramientas y APIs que permiten a los desarrolladores cargar, almacenar, optimizar y distribuir contenido multimedia de manera eficiente. Cloudinary utiliza una arquitectura basada en la nube que incluye servidores de transformación y un sistema de entrega mediante redes de distribución de contenido (CDN), lo que asegura un acceso rápido y escalable a los recursos. Entre sus funcionalidades clave se encuentran la optimización automática de imágenes y videos (como ajustes de formato y compresión), transformaciones dinámicas (recorte, redimensionamiento, aplicación de efectos), y soporte para la carga masiva de archivos.

La plataforma es especialmente útil en aplicaciones web y móviles donde el rendimiento y la experiencia del usuario son críticos, ya que permite adaptar contenido multimedia a diferentes dispositivos y contextos sin necesidad de procesamiento manual. Cloudinary también ofrece integraciones con *framework*s populares y herramientas de desarrollo, así como capacidades avanzadas como reconocimiento facial y análisis basado en inteligencia artificial. En el ámbito académico y profesional, es una solución destacada para proyectos que requieren manejo eficiente de activos digitales y entrega optimizada.

Ilustración 11: Cloudinary

3. Gestión del proyecto

3.1 Metodología de Desarrollo

Enfoque Iterativo e Incremental Personalizado:

Para el desarrollo de este proyecto, implementaré una metodología propia basada en los principios del desarrollo iterativo e incremental, adaptada a las necesidades específicas de un proyecto individual. Este enfoque me permitirá construir el sistema de manera progresiva, añadiendo funcionalidades en ciclos cortos y obteniendo un producto funcional desde las primeras etapas. Estas son las principales características de la metodología a utilizar:

- **Ciclos de desarrollo cortos**: Dividiré el desarrollo en iteraciones de 1-2 semanas, cada una con objetivos claros y entregables definidos.
- **Desarrollo incremental**: Cada ciclo añadirá nuevas funcionalidades al producto existente, aumentando gradualmente el valor y la completitud del sistema.
- **Enfoque en funcionalidades prioritarias**: Comenzaré por implementar las funcionalidades *core* del sistema, estableciendo una base sólida sobre la que construir características adicionales.
- **Revisión constante**: Al final de cada iteración, realizaré una evaluación del trabajo completado, identificando posibles mejoras y ajustando la planificación según sea necesario.
- **Flexibilidad ante cambios**: La metodología me permitirá adaptar el desarrollo a nuevos requisitos o modificaciones sin comprometer el avance general del proyecto, sobre todo porque el alcance puede sufrir modificaciones.

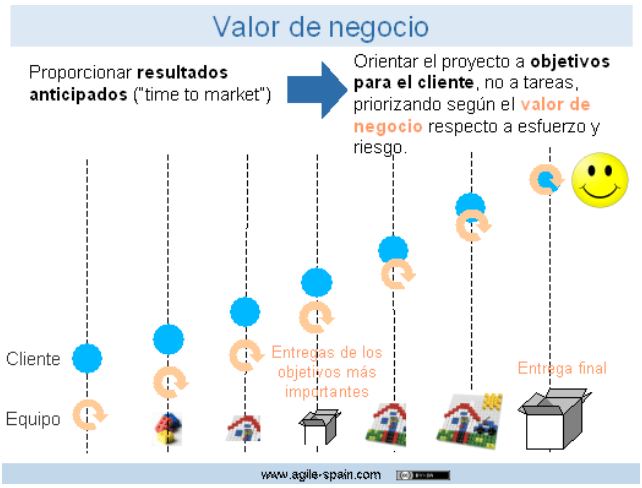


Ilustración 12: Imagen proceso iterativo e incremental [16]

Proceso de desarrollo de cada iteración:

1. **Planificación de la iteración**:

- Selección de funcionalidades a implementar
- Establecimiento de objetivos concretos
- Estimación de tiempo necesario

2. **Desarrollo**:
   - Implementación de las funcionalidades planificadas
   - Documentación del código
   - Pruebas unitarias durante el desarrollo
3. **Pruebas**:
   - Verificación del funcionamiento correcto
   - Validación con los requisitos establecidos
   - Identificación de errores o mejoras necesarias

Herramientas de apoyo:

- **Control de versiones**: Implementaré Git para gestionar los cambios en el código y mantener un historial de versiones
- **Documentación**: Mantendré un diario de desarrollo para registrar decisiones importantes y problemas encontrados

Este enfoque metodológico personalizado me permitirá mantener un ritmo constante de desarrollo, adaptarme a los desafíos que encuentre durante el proceso y garantizar un producto final que cumpla con los requisitos establecidos, todo ello optimizando los recursos disponibles al ser un proyecto desarrollado por una sola persona.

3.2 Planificación temporal

La planificación temporal del proyecto se estructurará en fases bien definidas con una duración total estimada de 16 semanas. A continuación, se presenta el cronograma general del proyecto:

- **Fase 1: Análisis y Diseño (3 semanas)**

  Semana 1-2: Análisis de requisitos y especificaciones.

  Semana 2-3: Diseño de arquitectura y modelado de datos.

- **Fase 2: Desarrollo Iterativo (10 semanas)**

  Iteración 1 (Semanas 4-5): Implementación del núcleo básico.

Iteración 2 (Semanas 6-7): Desarrollo de funcionalidades principales.

Iteración 3 (Semanas 8-9): Implementación de características secundarias.

Iteración 4 (Semanas 10-11): Desarrollo de interfaz de usuario.

Iteración 5 (Semanas 12-13): Integración y refinamiento.

- **Fase 3: Pruebas y Optimización (2 semanas)**

  Semana 14: Pruebas integrales y corrección de errores.

  Semana 15: Optimización de rendimiento y experiencia de usuario.

- **Fase 4: Documentación y Entrega (1 semana)**

  Semana 16: Finalización de la documentación y preparación para la entrega.

3.2.1 Planificación Detallada por Iteraciones

- **Iteración 1: Núcleo Básico (2 semanas):**

  Configuración del entorno de desarrollo.

  Implementación de la estructura básica del *backend* (Node.js/Express).

  Creación del esquema de base de datos (MongoDB).

  Desarrollo de funcionalidades CRUD básicas.

Resultados esperados: Aplicación con funcionalidades básicas operativas.

Tiempo estimado: 40 horas.

- **Iteración 2: Funcionalidades Principales (2 semanas):**

  Desarrollo de la lógica de negocio central.

  Implementación de autenticación y autorización.

  Creación de *endpoints* de API esenciales.

  Inicio del desarrollo del *frontend* (Angular 18).

Resultados esperados: Sistema con funcionalidades principales operativas.

Tiempo estimado: 40 horas.

- **Iteración 3: Características Secundarias (2 semanas):**

  Implementación de funcionalidades secundarias.

Desarrollo de reportes y visualizaciones.

Mejora de la interacción entre *frontend* y *backend.*

Implementación de validaciones y manejo de errores.

Resultados esperados: Aplicación con funcionalidad completa a nivel lógico.

Tiempo estimado: 40 horas.

- **Iteración 4: Interfaz de Usuario (2 semanas)**

  Diseño e implementación de componentes UI.

  Desarrollo de formularios y validaciones en cliente.

  Implementación de navegación y flujos de usuario.

  Resultados esperados: Interfaz de usuario completa y funcional.

  Tiempo estimado: 40 horas.

- **Iteración 5: Integración y Refinamiento (2 semanas)**

  Integración de módulos desarrollados.

  Refinamiento de la experiencia de usuario.

  Implementación de funcionalidades avanzadas.

  Optimización inicial de rendimiento.

  Resultados esperados: Sistema completamente integrado y refinado.

  Tiempo estimado: 40 horas.

Hitos Principales

1. **Fin de análisis y diseño** (Semana 3): Documento de especificaciones y arquitectura completado.
2. **Prototipo funcional** (Semana 7): Primera versión con funcionalidades básicas operativas.
3. **Versión beta** (Semana 13): Sistema con todas las funcionalidades implementadas.
4. **Entrega final** (Semana 16): Producto completado.

Esta planificación temporal permitirá gestionar eficientemente el desarrollo del proyecto, estableciendo objetivos claros para cada fase e iteración, y facilitando el seguimiento del progreso a lo largo del tiempo.

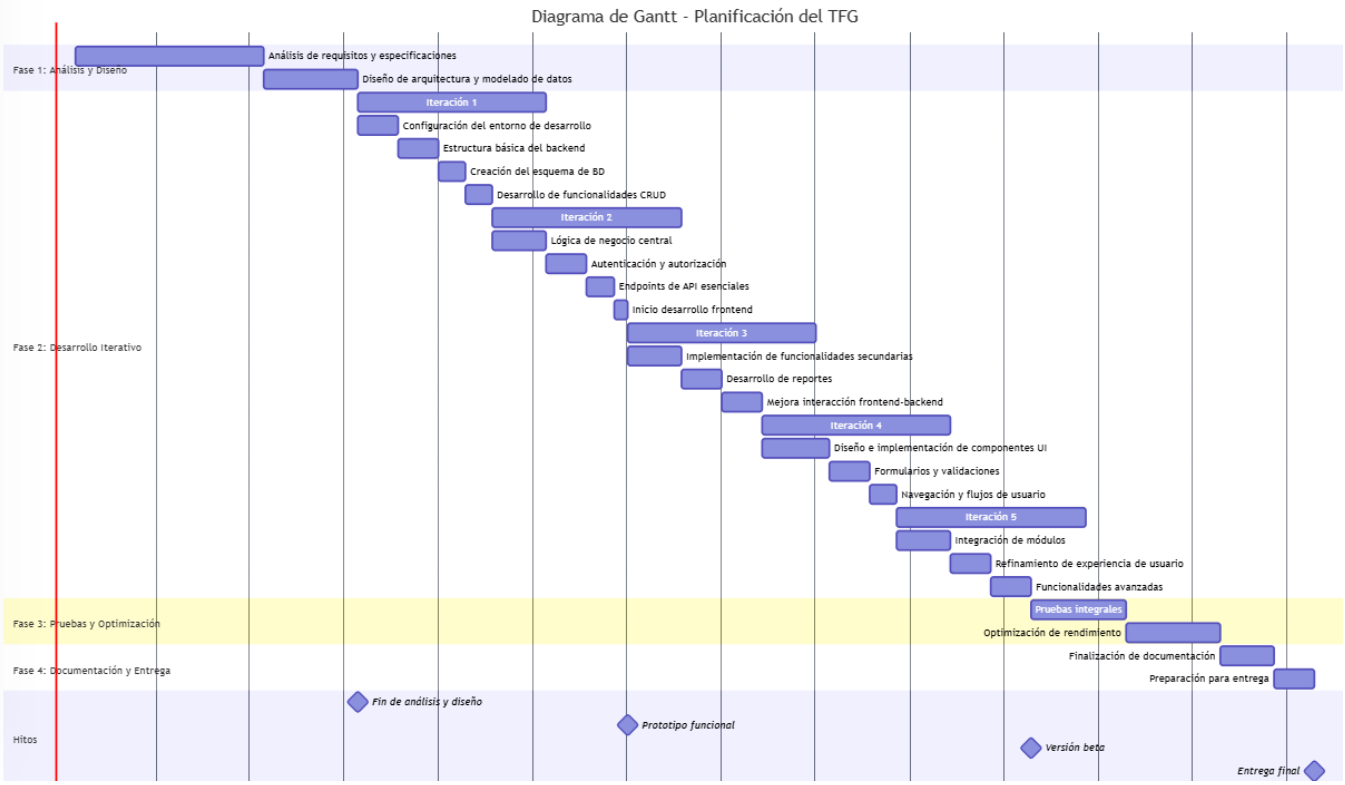


Ilustración 13: Diagrama de Gantt

3.3 Gestión de recursos

Al ser un proyecto individual, la gestión de recursos humanos se centrará en la optimización de mi propio tiempo y capacidades (autogestión del tiempo):

- **Horario de trabajo**: Dedicaré 20 horas semanales al desarrollo del proyecto, distribuidas de la siguiente manera:

  14 horas en días laborables (aproximadamente 3 horas diarias).

  6 horas en fines de semana (3 horas cada día).

- **Gestión de interrupciones**: Estableceré bloques de tiempo ininterrumpido para tareas que requieran alta concentración, así como si por causas personales he de interrumpir el desarrollo lo podré hacer.

- **Desarrollo de capacidades:**
  - **Recursos de formación**: Utilizaré documentación oficial, tutoriales y cursos en línea específicos para Angular 18, Node.js y MongoDB.
  - **Asesoramiento**: Programaré consultas puntuales con el tutor en momentos clave del desarrollo.

3.4 Gestión de riesgos

La gestión de riesgos es un aspecto fundamental para garantizar el éxito del proyecto. Se han identificado los principales riesgos potenciales y establecido estrategias para mitigarlos:

Tabla 2: Gestión de riesgos

Riesgo	Probabilidad	Impacto	Valor de riesgo	Estrategia de mitigación
Cambios en los requisitos	Media	Alto	Alto	Metodología iterativa que permite adaptación.
Problemas técnicos con las tecnologías seleccionadas	Media	Alto	Alto	Investigación previa de las tecnologías. Alternativas tecnológicas identificadas
Retrasos en la planificación	Alta	Medio	Alto	Priorización clara de funcionalidades. Incluir márgenes de tiempo en las iteraciones.
Limitaciones de rendimiento	Media	Medio	Medio	Desarrollo incremental con pruebas de integración frecuentes.
Indisponibilidad por motivos personales	Alta	Alto	Alto	Planificación con margen de tiempo. Documentación para facilitar la reanudación del trabajo.

3.4.1 Estrategia control de versiones

Para mantener un desarrollo organizado y facilitar la gestión de cambios, implementaré una estrategia de control de versiones utilizando Git con el siguiente enfoque:

Estructura de ramas

- **main**: Rama principal que contiene el código estable y probado. Solo se fusionan cambios completamente verificados.
- **deve**: Rama de desarrollo donde se integran las funcionalidades completadas.
- **feature/[nombre-funcionalidad]**: Ramas temporales para el desarrollo de funcionalidades específicas.

Políticas de commits

- Commits atómicos y frecuentes (idealmente al finalizar cada unidad lógica de trabajo).
- Mensajes de commits descriptivos siguiendo el formato: tipo(alcance): descripción corta
    - Tipos: feat (funcionalidad), fix (corrección), docs (documentación), style (formato), refactor, test, chore (tareas).
    - Ejemplo: feat(auth): implementar sistema de autenticación básico
- Mensajes en castellano para mantener consistencia con la documentación.

Flujo de trabajo

1. Crear rama de funcionalidad desde dev
2. Desarrollar con commits frecuentes
3. Realizar pruebas unitarias locales
4. Solicitar revisión personal (auto-revisión) antes de fusionar
5. Fusionar a dev cuando la funcionalidad esté completa
6. Fusionar a main al finalizar cada iteración significativa

Este enfoque me permitirá mantener un historial claro del desarrollo, facilitar la identificación y corrección de errores, y disponer de versiones estables en cualquier momento del proyecto.

3.5 Presupuesto económico

Para realizar esta estimación, se considerarán las herramientas de hardware y software utilizadas, así como los factores de impacto relacionados con la duración del proyecto y los gastos derivados del personal. Esta evaluación se basará en la estimación de horas planificadas y en los roles específicos asignados a cada tarea.

El costo total del proyecto se dividirá en cuatro categorías principales: hardware, licencias de software, otros costes y costos de personal.

Tabla coste *hardware*:

Componentes	Coste (€)	Uso (%)	Total (€)
Ordenador portátil	750,00 €	10%	75,00€
Ratón	10,00€	5%	0,50€
Monitor	110,00€	3%	3,30€
Teclado	40,00€	5%	2,00€
TOTAL			80,80€

Tabla 3: Costes hardware

Tabla coste *software*:

Licencia	Coste (€)	Uso (%)	Total (€)
MongoDB Atlas	0,00 €		0,00€
Visual Studio Code	0,00 €		0,00€
Figma	288,00 €	33,3%	95,04€
Github	0,00 €		0,00€
Brave	0,00 €		0,00€
Windows 11	145,00€	33,3%	47,85€
Microsoft Office	69,00€	33,3%	22,77€
Mendeley	0,00€		0,00€
TOTAL			165,66€

Tabla 4: Coste software

Para calcular el coste de recursos humanos en este proyecto, es necesario considerar las tarifas estándar del mercado actual. Se han establecido dos perfiles profesionales para el desarrollo: Desarrollador Junior y Analista de Sistemas. Según las referencias consultadas en el sector tecnológico español, incluyendo datos del observatorio salarial de asociaciones profesionales informáticas y estudios de mercado recientes, un Desarrollador Junior percibe aproximadamente 22.800€ anuales, mientras que un Analista puede recibir en torno a 35.500€ anuales.

El proyecto requiere una dedicación total de 320 horas, distribuidas a lo largo de 16 semanas con una intensidad de 20 horas semanales. Esta carga de trabajo se ha repartido entre ambos perfiles de la siguiente manera:

1. **Perfil técnico (Desarrollador)**: Este rol asumirá el 81,25% del esfuerzo total, lo que representa 260 horas de trabajo.
2. **Perfil analítico**: Corresponde al 18,75% restante, equivalente a 60 horas de dedicación.

Para determinar el coste horario de cada perfil, se ha aplicado la siguiente fórmula:

Tarifa horaria = Salario anual ÷ Horas laborables anuales

Considerando que un profesional trabaja aproximadamente **173,33 horas mensuales** (40 horas semanales), el cómputo anual asciende a **2.080 horas**. Así, obtenemos:

- Tarifa Desarrollador = 22.800€ ÷ 2.080 horas = 10,96€/hora
- Tarifa Analista = 35.500€ ÷ 2.080 horas = 17,07€/hora

Estos valores servirán como base para calcular el coste total de personal del proyecto.

Tabla coste personal:

Rol	Hora (h)	Coste/Hora (h/€)	Total (€)
Desarrollador	260	10,96€	2.849,60€
Analista	60	17,07€	1024,20€
TOTAL			3873,80€

Tabla 5: Coste personal

Tabla otros costes:

Servicio	Coste (€) / mes	Tiempo (mes)	Total (€)
Costes eléctricos	30,00 €	4	120,00€
Conexión a internet	45,00€	4	180,00€
TOTAL			300,00€

Tabla 6: Otros costes

Tras haber calculado todos los costes por separado, realizamos una tabla que recoge el total de todos los costes:

Componentes	Total (€)
Hardware	80,80€
Software	165,66€
Recursos humanos	3873,80€
Otros costes	300,00€
TOTAL	**4420,26€**

Tabla 7: Presupuesto total

3.5.1 Presupuesto con contingencia

Considerando la naturaleza del proyecto y los riesgos identificados, es prudente incluir un margen de contingencia en el presupuesto total. Esta reserva económica permitirá abordar situaciones imprevistas sin comprometer la viabilidad del proyecto.

Se aplicará un porcentaje de contingencia diferenciado según la naturaleza de cada categoría de costes:

Componentes	Presupuesto base (€)	Contingencia (%)	Total (€)
Hardware	80,80€	5%	84,84€
Software	165,66€	10%	182,23€
Recursos humanos	3873,80€	15%	4.454,87€
Otros costes	300,00€	10%	330,00€
TOTAL	**4420,26€**	**14,3%**	**5.051,94€**

Tabla 8: Presupuesto con contingencia

- Hardware (5%): Bajo riesgo de variaciones significativas en costes o necesidades adicionales.

- **Software (10%)**: Posibilidad de requerir herramientas adicionales no previstas inicialmente.

- Recursos humanos (15%): Mayor incertidumbre debido a la complejidad técnica y posibles retrasos en el desarrollo.

- **Otros costes (10%)**: Margen para cubrir variaciones en consumos o servicios adicionales.

El presupuesto total con contingencia asciende a **5.051,94€**, lo que representa un incremento del 14,3% sobre el presupuesto base. Esta reserva proporcionará un margen de seguridad financiera adecuado para afrontar imprevistos durante la ejecución del proyecto.

4. Análisis

Este capítulo recopila toda la información obtenida durante la fase de análisis, cuyo objetivo es comprender en detalle los requerimientos del cliente, en este caso, los actores involucrados en la plataforma LinkByBall. En primer lugar, se describen las características del sistema y los distintos tipos de usuarios que interactuarán con la aplicación. Posteriormente, se presentan los requisitos de usuario junto con un modelo de Casos de Uso que especifica cada funcionalidad clave. A partir de estos, se derivan los requisitos funcionales y se detallan los requisitos no funcionales. Finalmente, se incluye un modelo Entidad-Relación que representa la estructura de la información en el sistema, seguido de las principales restricciones que deben considerarse en su desarrollo.

4.1 Características del sistema

El objetivo es proporcionar una visión general de cómo el sistema está diseñado para cumplir con sus objetivos funcionales y no funcionales, asegurando que sea eficiente, seguro, accesible y adaptable a futuras mejoras o ampliaciones. Para ello se detallan las principales características del sistema junto con una representación visual mediante un diagrama (árbol de características):

1. Autenticación y Gestión de Usuarios

- Registro de usuarios (club, entrenador, futbolista).
- Gestión de sesión
    - Iniciar sesión
    - Cerrar sesión
    - Autenticar sesión
- Edición de perfil.
- Eliminación de cuenta.
- Búsqueda personalizada de perfiles

2. Gestión de Vacantes

- Creación de vacantes por parte de los clubes.
- Búsqueda y filtrado de vacantes.
- Solicitud y postulación de futbolistas y entrenadores.
    - Modificar estado
    - Visualización de solicitudes

3. Sistema de Favoritos y Avisos

- Gestión de perfiles favoritos.
    - Añadir perfil a favoritos
    - Eliminar perfil de favoritos
    - Contactar perfil
- Gestión de Avisos y notificaciones
    - Aviso modificación de perfil favorito
    - Aviso cambio de estado en solicitud
    - Aviso nueva solicitud a vacante

4. Comunicación y Chat

- Mensajería en tiempo real entre usuarios.
- Historial de conversaciones.
- Notificaciones de nuevos mensajes.

4.1.1 Árbol de características

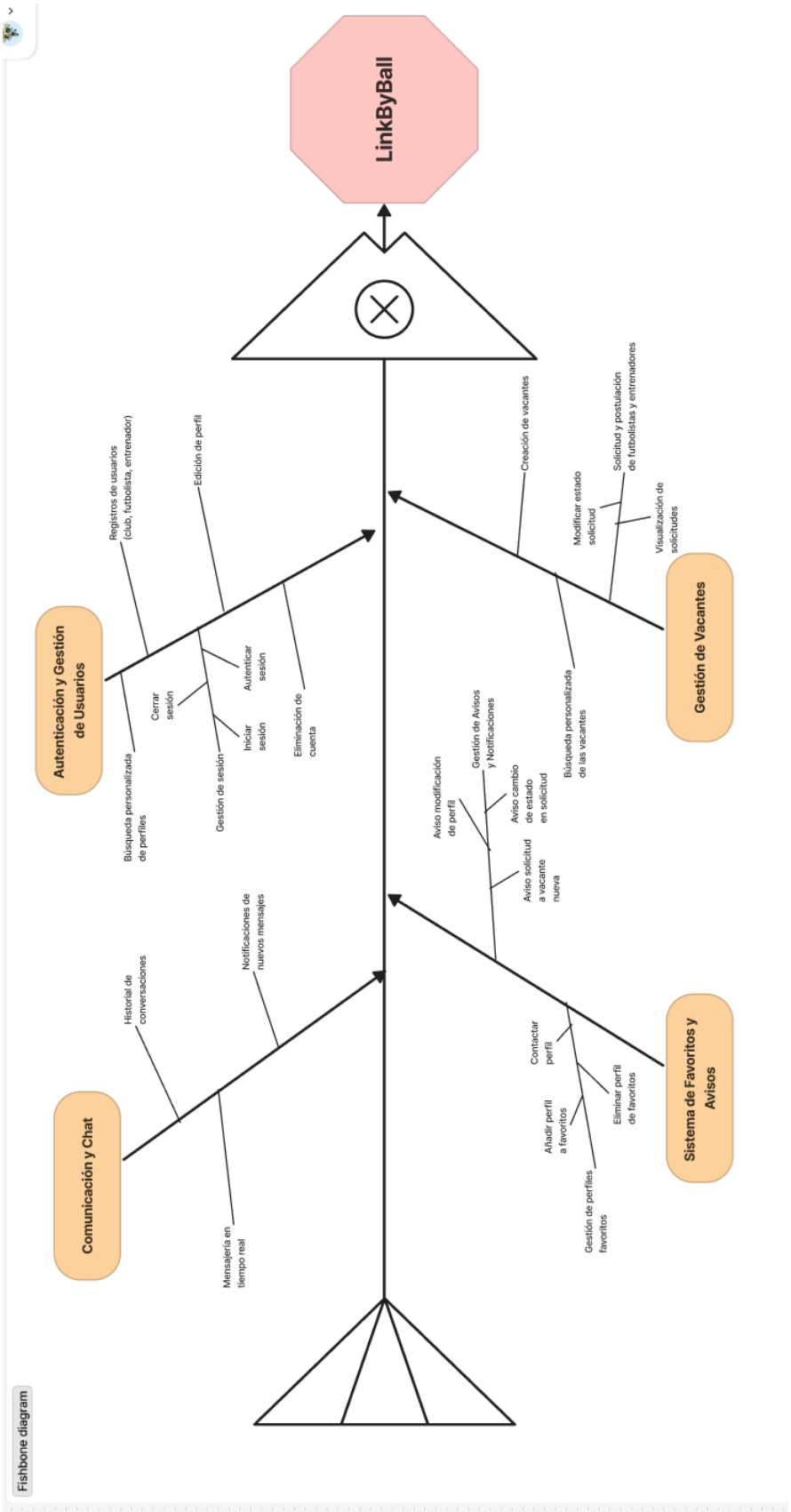


Ilustración 14: Diagrama de características

4.2 Requisitos de Usuario

Los requisitos de usuario describen las necesidades y expectativas de los usuarios finales con respecto al sistema. Se centran en cómo interactúan con la aplicación y qué funcionalidades esperan encontrar para alcanzar sus objetivos de manera eficiente.

A continuación, se detallan los requisitos de usuario para el sistema LinkByBall:

RU-01: El usuario debe poder registrarse en la plataforma proporcionando su correo electrónico, nombre, rol (futbolista, entrenador o club) y una contraseña segura.

RU-02: El usuario debe poder iniciar sesión con su correo y contraseña.

RU-03: El usuario debe poder editar su perfil, actualizando información como foto, experiencia, posición, descripción...

RU-04: Los futbolistas y entrenadores deben poder buscar vacantes publicadas por los clubes.

RU-05: Los clubes deben poder publicar vacantes indicando detalles como posición requerida, nivel y descripción, opcionalmente el salario.

RU-06: Los futbolistas y entrenadores deben poder postularse a una vacante haciendo una solicitud.

RU-07: Los clubes deben poder ver la lista de postulantes a sus vacantes.

RU-08: El usuario debe poder agregar a otros usuarios a su lista de favoritos para hacer seguimiento de sus perfiles.

RU-09: El usuario debe recibir notificaciones cuando alguien de su lista de favoritos actualice su perfil.

RU-10: Se debe permitir la comunicación entre usuarios a través de un chat integrado en la plataforma.

RU-11: El usuario debe poder buscar perfiles de otros usuarios utilizando filtros como rol, ubicación, experiencia…

RU-12: El usuario debe poder eliminar su cuenta en cualquier momento si así lo desea.

RU-13: El usuario de tipo administrador debe poder verificar a los clubes, cerciorándose que no hay duplicados y de que es un perfil verdadero.

RU-14: La plataforma debe mostrar un *dashboard* personalizado según el tipo de usuario (club, entrenador o futbolista).

RU-15: Los clubes deben poder gestionar las solicitudes recibidas, aceptando o rechazando postulantes.

RU-16: Los usuarios deben poder cerrar sesión de forma segura en cualquier momento.

4.2.1 Diagramas de casos de uso

Se muestran los diagramas de casos de uso:

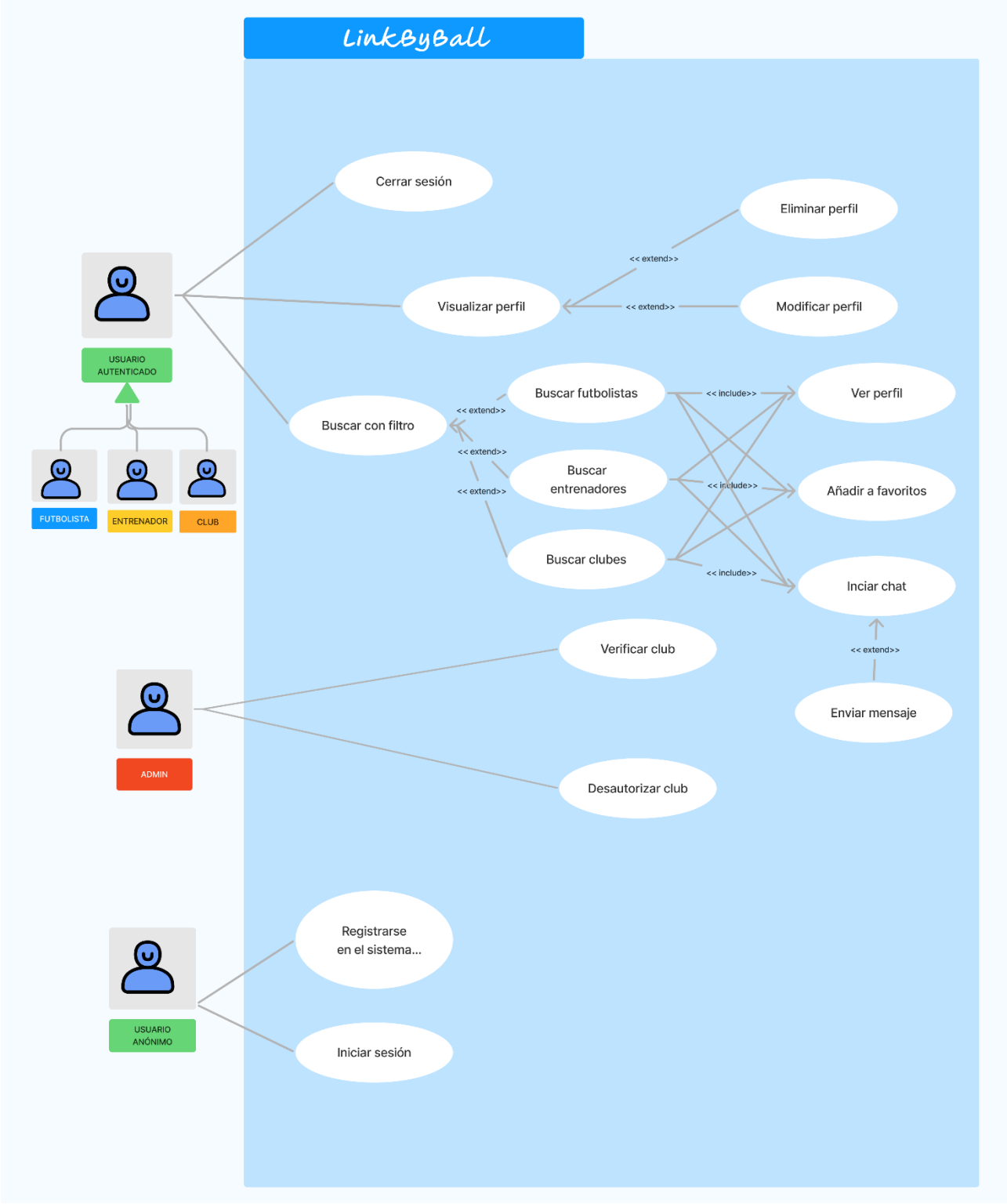


Ilustración 15: Diagrama caso de uso

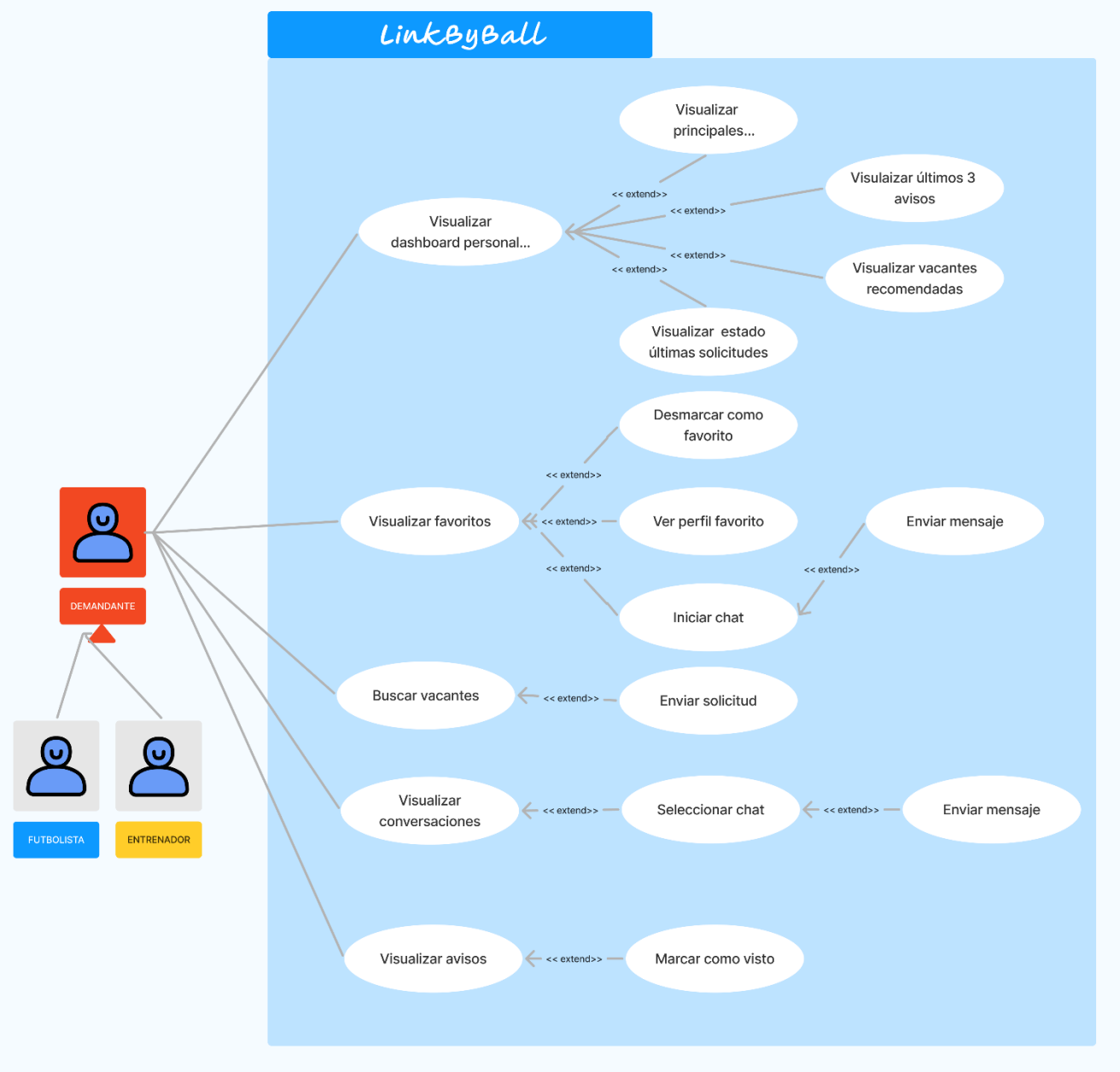


Ilustración 16:Diagrama caso de uso (futbolista, entrenador)

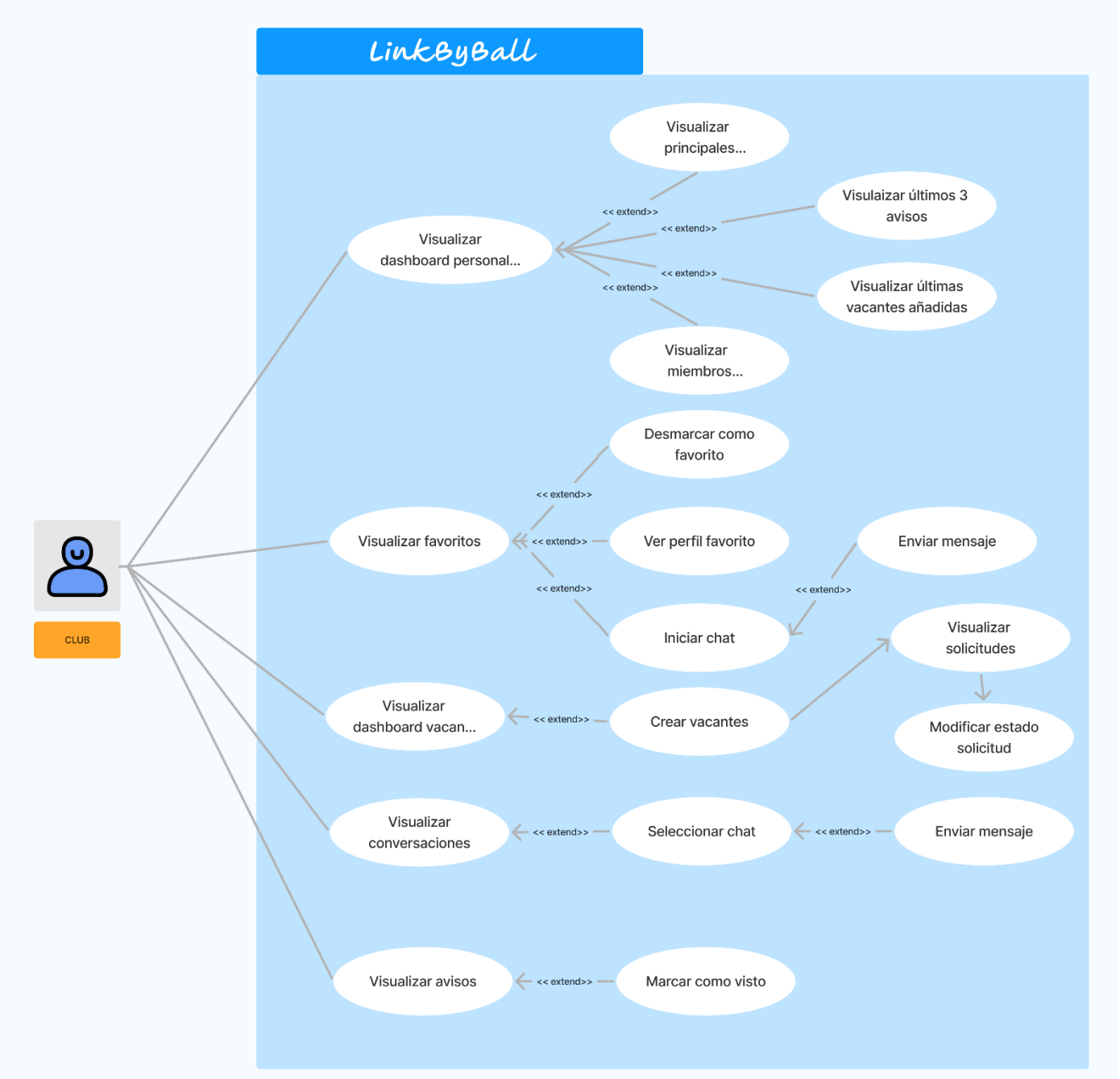


Ilustración 17: Diagrama caso de uso (club)

4.2.2 Tablas casos de uso

RU-01	Registro en el sistema
Versión	1.0.0
Actores	Futbolista, Entrenador, Club
Descripción	El usuario que desea crearse una cuenta introduce los datos del formulario correspondiente
Precondiciones	El usuario debe seleccionar qué tipo de usuario es antes de rellenar uno de los tres formularios
Secuencia normal	1- El usuario elige una opción de registro de entre futbolista, entrenador o club.
	2- El usuario rellena el formulario correspondiente aportando los datos obligatorios para el registro exitoso.
	3- El sistema inicia sesión automáticamente una vez han sido validados los datos.
Postcondiciones	El usuario se ha registrado exitosamente y ahora podrá iniciar sesión con su email y contraseña.
Excepciones	1- El usuario no ha completado el formulario con los campos obligatorios.
	2- El usuario ha introducido un email ya registrado en la base de datos.
Frecuencia	Solo será necesario realizarlo una única vez.
Importancia	Alta

Tabla 9: Requisito de usuario 1

RU-02	Inicio de sesión
Versión	1.0.0
Actores	Futbolista, Entrenador, Club, Administrador
Descripción	El usuario iniciará sesión en el sistema aportando el correo electrónico y contraseña.
Precondiciones	El usuario deberá estar registrado en la base de datos.
Secuencia normal	1- El usuario completa los campos correo electrónico y contraseña.
	2- El sistema comprueba si el usuario está registrado.
	3- Se inicia sesión y se enruta con la página principal del tipo de usuario que ha iniciado sesión.
Postcondiciones	El usuario podrá acceder únicamente a las funcionalidades de su tipo de usuario (futbolista, entrenador, club, administrador). Es importante destacar que la sesión (*token*) solo es válido durante treinta minutos.
Excepciones	1- El usuario no ha introducido un correo o contraseña válidos.
	2- El usuario no está registrado en el sistema.
Frecuencia	La sesión será válida solo durante treinta minutos por lo que se deberá iniciar sesión nuevamente.
Importancia	Alta

Tabla 10: Requisito de usuario 2

RU-03	Modificar perfil
Versión	1.0.0
Actores	Futbolista, Entrenador, Club
Descripción	El usuario podrá editar su perfil actualizando información como foto, experiencia, posición o especialidad, descripción…
Precondiciones	El usuario debe haber iniciado sesión en el sistema.
Secuencia normal	1- El usuario accede a la sección de edición de perfil.
	2- Modifica los campos deseados.
	3- El sistema valida los datos, guarda los cambios y actualiza la información.
Postcondiciones	La información del perfil se actualiza y es visible para otros usuarios.
Excepciones	1- El usuario intenta guardar sin completar los campos obligatorios.
	2- Error en la carga de la foto de perfil.
Frecuencia	Los usuarios pueden editar su perfil en cualquier momento.
Importancia	Alta

Tabla 11: Requisito de usuario 3

RU-04	Búsqueda de vacantes
Versión	1.0.0
Actores	Futbolista, Entrenador
Descripción	Los futbolistas y entrenadores podrán buscar vacantes publicadas por los clubes.
Precondiciones	Debe haber vacantes publicadas en la plataforma.
Secuencia normal	1- El usuario accede a la sección de búsqueda de vacantes.
	2- El usuario aplica filtros según sus intereses.
	3- El usuario visualiza los resultados y selecciona una vacante para ver los detalles.
Postcondiciones	El usuario obtiene una lista de vacantes que coinciden con su búsqueda.
Excepciones	1- No hay vacantes disponibles.
	2- Error en la carga de los resultados.
Frecuencia	Según la necesidad del usuario.
Importancia	Media

Tabla 12: Requisito de usuario 4

RU-05	Publicación de vacantes
Versión	1.0.0
Actores	Club
Descripción	Los clubes pueden publicar vacantes indicando posición requerida o especialidad en caso de ser entrenador, descripción y opcionalmente el salario.
Precondiciones	El club debe haber iniciado sesión en la plataforma.
Secuencia normal	1- El club accede a la sección de publicación de vacantes.
	2- El club completa los detalles de la vacante.
	3- El sistema guarda y publica la vacante.
Postcondiciones	La vacante es visible para futbolistas en caso de ser solo para futbolista y lo mismo en el caso de ser para entrenadores o miembros del cuerpo técnico.
Excepciones	1- No se completan los campos obligatorios.
	2- Error en la publicación de la vacante.
Frecuencia	Cuando el club necesite nuevos jugadores o entrenadores.
Importancia	Alta

Tabla 13: Requisito de usuario 5

RU-06	Solicitud a vacantes
Versión	1.0.0
Actores	Futbolista, Entrenador
Descripción	Los futbolistas y entrenadores podrán postularse a una vacante enviando una solicitud.
Precondiciones	Debe haber vacantes disponibles y el usuario debe haber iniciado sesión.
Secuencia normal	1- El usuario accede a una vacante.
	2- El usuario hace clic en el botón de enviar solicitud.
	3- El usuario confirma la solicitud y el sistema registra su solicitud.
Postcondiciones	El usuario queda registrado como postulante en la vacante.
Excepciones	1- El usuario ya se postuló anteriormente.
	2- La vacante ya no está disponible.
Frecuencia	Según disponibilidad de vacantes y decisión del usuario.
Importancia	Alta

Tabla 14: Requisito de usuario 6

RU-07	Visualización de postulantes
Versión	1.0.0
Actores	Club
Descripción	Los clubes pueden acceder a una lista con los postulantes a sus vacantes publicadas y visualizar la información de cada candidato.
Precondiciones	El club debe haber iniciado sesión en la plataforma. Debe existir al menos una vacante publicada con postulaciones registradas.
Secuencia normal	1- El club accede a la sección de gestión de vacantes.
	2- El club selecciona una vacante publicada.
	3- El sistema muestra la lista de postulantes con información relevante como nombre, posición y experiencia.
	4- El club puede hacer clic en un postulante para ver su perfil completo.
Postcondiciones	El club obtiene información detallada sobre los postulantes. Puede contactar o tomar decisiones respecto a las postulaciones recibidas.
Excepciones	Error en la carga de los datos. Se muestra un mensaje de error y se sugiere intentarlo más tarde.
Frecuencia	Según la necesidad del club de revisar postulaciones.
Importancia	Alta

Tabla 15: Requisito de usuario 7

RU-08	Agregar favoritos
Versión	1.0.0
Actores	Futbolista, Entrenador, Club
Descripción	Los usuarios pueden agregar a otros usuarios a su lista de favoritos para hacer seguimiento de sus perfiles y recibir notificaciones cuando actualicen su información.
Precondiciones	El usuario debe haber iniciado sesión. El usuario objetivo debe tener un perfil público visible en la plataforma (esto es para los clubes).
Secuencia normal	1- El usuario accede al perfil de otro usuario.
	2- El usuario hace clic en el botón "Agregar a favoritos".
	3- El sistema guarda la relación en la lista de favoritos del usuario.
	4- El usuario puede ver a sus favoritos en una sección específica.
Postcondiciones	El usuario puede acceder rápidamente a sus favoritos desde su lista personal. Se activan las notificaciones en caso de que el usuario agregado actualice su perfil.
Excepciones	El usuario que queremos incluir como favorito ya está en la lista de favoritos
Frecuencia	Según el interés del usuario en seguir otros perfiles.
Importancia	Alta

Tabla 16: Requisito de usuario 8

RU-09	Notificaciones de actualización de perfil
Versión	1.0.0
Actores	Futbolista, Entrenador, Club
Descripción	Los usuarios recibirán notificaciones cuando alguien que tienen en su lista de favoritos realice cambios en su perfil.
Precondiciones	El usuario debe haber agregado a otro usuario a su lista de favoritos. El usuario favorito debe modificar su perfil.
Secuencia normal	1- Un usuario que está en la lista de favoritos realiza cambios en su perfil (foto, experiencia, posición, etc.).
	2- El sistema detecta la actualización.
	3- El sistema genera una notificación para todos los usuarios que tienen a esa persona en favoritos.
	4- El sistema muestra la notificación en el área de avisos del usuario.
Postcondiciones	Los usuarios reciben el aviso. Pueden hacer clic en la notificación para acceder al perfil actualizado.
Excepciones	Se genera un error al generar el aviso.
Frecuencia	Cada vez que un usuario de la lista de favoritos actualiza su perfil.
Importancia	Media

Tabla 17: Requisito de usuario 9

RU-10	Chat entre usuarios
Versión	1.0.0
Actores	Futbolista, Entrenador, Club
Descripción	Se debe permitir la comunicación entre usuarios a través de un sistema de chat en la plataforma.
Precondiciones	Ambos usuarios deben estar registrados en el sistema. El chat debe estar habilitado para los perfiles involucrados.
Secuencia normal	1- Un usuario accede al chat desde su cuenta.
	2- El usuario selecciona un contacto con quien desea conversar.
	3- El usuario escribe un mensaje y lo envía.
	4- El otro usuario recibe el mensaje en tiempo real y puede responder.
Postcondiciones	La conversación queda guardada en la plataforma. Los usuarios pueden continuar la conversación en cualquier momento.
Excepciones	Fallo en la conexión del socket.
Frecuencia	Según la necesidad de los usuarios.
Importancia	Alta

Tabla 18: Requisito de usuario 10

RU-11	Búsqueda de perfiles
Versión	1.0.0
Actores	Futbolista, Entrenador, Club
Descripción	El usuario debe poder buscar perfiles de otros usuarios utilizando filtros como rol, ubicación, experiencia y más.
Precondiciones	El usuario debe haber iniciado sesión. Debe haber perfiles disponibles que coincidan con los criterios de búsqueda.
Secuencia normal	1- El usuario accede a la sección de búsqueda de la plataforma.
	2- El usuario introduce los criterios de búsqueda (rol, ubicación, experiencia, etc.).
	3- El sistema filtra y muestra los resultados correspondientes.
	4- El usuario puede seleccionar un perfil para ver información detallada.
Postcondiciones	El usuario obtiene una lista de perfiles que cumplen con los filtros establecidos. Puede interactuar con los perfiles encontrados.
Excepciones	No hay resultados para los filtros aplicados
Frecuencia	Según la necesidad del usuario de encontrar nuevos contactos.
Importancia	Medio

Tabla 19: Requisito de usuario 11

RU-12	Eliminación de cuenta
Versión	1.0.0
Actores	Futbolista, Entrenador, Club
Descripción	El usuario debe poder eliminar su cuenta en cualquier momento si así lo desea.
Precondiciones	El usuario debe haber iniciado sesión. No debe tener procesos pendientes (ej. postulaciones activas).
Secuencia normal	1- El usuario accede a los detalles del perfil.
	2- El usuario selecciona la opción "Eliminar cuenta".
	3- El usuario confirma la acción tras un aviso de advertencia.
	4- El sistema elimina la cuenta y borra sus datos asociados.
Postcondiciones	La cuenta del usuario es eliminada permanentemente del sistema.
Excepciones	No se encuentra el usuario.
Frecuencia	Bajo, solo cuando un usuario decide abandonar la plataforma.
Importancia	Media

Tabla 20: Requisito de usuario 12

RU-13	Verificación de clubes
Versión	1.0.0
Actores	Administrador
Descripción	Los administradores podrán verificar y desautorizar a los clubes que acaban de registrarse en el sistema.
Precondiciones	Los clubes deben estar dados de alta.
Secuencia normal	1- El administrador visualiza la lista de clubes no verificados o verificados.
	2- El administrador verifica al club seleccionado
Postcondiciones	El club aparecerá público en las búsquedas.
Excepciones	La acción de verificar club no se ha completado correctamente.
Frecuencia	Cada vez que un club se da de alta.
Importancia	Alta

Tabla 21: Requisito de usuario 13

RU-14	Visualizar dashboard personalizado
Versión	1.0.0
Actores	Futbolista, Entrenador, Club
Descripción	La plataforma debe mostrar un *dashboard* personalizado según el tipo de usuario (club, entrenador o futbolista).
Precondiciones	El usuario debe haber iniciado sesión.
Secuencia normal	1- El usuario accede a la plataforma tras iniciar sesión.
	2- El sistema detecta su tipo de perfil.
	3- El sistema muestra un *dashboard* con información y opciones relevantes para su rol.
Postcondiciones	El usuario visualiza un *dashboard* adaptado a sus necesidades.
Excepciones	No se encuentran los datos del usuario
Frecuencia	Cada vez que el usuario accede a la plataforma.
Importancia	Alta

Tabla 22: Requisito de usuario 14

RU-15	Gestión de solicitudes por parte de clubes
Versión	1.0.0
Actores	Club
Descripción	Los clubes deben poder gestionar las solicitudes recibidas, aceptando o rechazando postulantes.
Precondiciones	El club debe haber iniciado sesión. Debe haber solicitudes recibidas en una vacante.
Secuencia normal	1- El club accede a la lista de solicitudes.
	2- El club revisa los postulantes disponibles.
	3- El club acepta o rechaza postulaciones según su criterio.
Postcondiciones	Los postulantes reciben una notificación con la decisión del club.
Excepciones	No hay postulaciones → Se muestra un mensaje indicando que no hay solicitudes.
Frecuencia	Según la gestión de vacantes del club.
Importancia	Alta

Tabla 23: Requisito de usuario 15

RU-16	Cierre de sesión
Versión	1.0.0
Actores	Futbolista, Entrenador, Club
Descripción	Los usuarios deben poder cerrar sesión de forma segura en cualquier momento.
Precondiciones	El usuario debe haber iniciado sesión.
Secuencia normal	1- El usuario accede a la opción de cerrar sesión.
	2- El sistema finaliza la sesión y redirige a la pantalla de inicio.
Postcondiciones	El usuario queda desconectado de la plataforma.
Excepciones	Error en el proceso → Se muestra un mensaje indicando que la sesión no pudo cerrarse correctamente.
Frecuencia	Cada vez que finaliza la sesión.
Importancia	Alta

Tabla 24: Requisito de usuario 16

4.3 Requisitos funcionales

Los requisitos funcionales describen qué debe hacer el sistema para cumplir con sus objetivos. Especifican las características, funciones y comportamientos que debe tener la aplicación para satisfacer las necesidades de los usuarios. A continuación, se detallan los requisitos funcionales del sistema:

RF-01: El sistema debe permitir que los usuarios se registren con un correo electrónico, nombre de usuario, contraseña, tipo de perfil (futbolista, entrenador, club).

RF-02: El sistema debe permitir que los usuarios inicien sesión utilizando su correo electrónico y contraseña.

RF-03: El sistema debe manejar la autenticación mediante *tokens* JWT para sesiones seguras.

RF-04: Los usuarios deben poder modificar su información personal, como nombre, foto de perfil, descripción y detalles de contacto.

RF-05: Los usuarios deben poder eliminar su cuenta de forma definitiva.

RF-06: Los perfiles deben tener campos específicos dependiendo del tipo de usuario (futbolista, entrenador o club).

RF-07: Los usuarios deben poder buscar otros perfiles (futbolistas, entrenadores y clubes) utilizando filtros como posición, edad, ubicación y otros parámetros relevantes.

RF-08: Los usuarios deben poder marcar otros perfiles como favoritos.

RF-09: Los usuarios deben tener una sección donde se muestren sus perfiles favoritos para acceder a ellos fácilmente.

RF-10: Los futbolistas y entrenadores deben poder enviar solicitudes a vacantes disponibles.

RF-11: El sistema debe permitir a los clubes ver las solicitudes recibidas y cambiar su estado (pendiente, en proceso, aceptado, rechazado).

RF-12: Los usuarios deben poder ver el estado de sus solicitudes (pendiente, aceptada, rechazada).

RF-13: Los clubes deben ver un listado de las vacantes activas y los futbolistas y entrenadores asociados a ellas.

RF-14: Los usuarios deben poder enviarse mensajes en tiempo real a través de un sistema de chat.

RF-15: Los usuarios deben poder ver el historial de conversaciones previas.

RF-16: El sistema debe permitir notificaciones cuando un mensaje es recibido.

RF-17: Los clubes deben recibir notificaciones cuando se modifique el estado de una solicitud o vacante.

RF-18: Los usuarios deben poder marcar las notificaciones como leídas.

RF-19: Al crear una vacante, los clubes deben poder definir el tipo de perfil requerido (futbolista o entrenador), puesto, salario (opcional) y descripción.

RF-20: Los clubes deben poder ver las solicitudes recibidas para cada vacante y gestionar su estado.

RF-21: Los usuarios deben poder ver la información más relevante de los perfiles (experiencia, habilidades, vacantes recomendadas, etc.).

RF-22: El sistema debe permitir verificar perfiles de clubes mediante un sistema de autenticación o verificación manual.

RF-23: Los futbolistas y entrenadores deben poder filtrar vacantes por ubicación, tipo de puesto, salario, club, entre otros.

RF-24: El sistema debe permitir personalizar la búsqueda para ofrecer resultados más específicos.

RF-25: Los futbolistas y entrenadores deben poder ver las solicitudes enviadas y su estado (pendiente, en proceso, aceptada o rechazada).

RF-26: Los clubes deben poder cambiar el estado de las solicitudes recibidas.

RF-27: El sistema debe garantizar que las funcionalidades sean intuitivas y fáciles de usar para todo tipo de usuarios.

RF-28: El sistema debe garantizar la seguridad de los datos de los usuarios a través de encriptación de contraseñas y otras medidas de seguridad.

RF-29: Las peticiones a la API deben ser validadas y gestionadas de forma segura para evitar vulnerabilidades.

4.3.1 Matriz de trazabilidad

RU/RF	RF-01	RF-02	RF-03	RF-04	RF-05	RF-06	RF-07	RF-08	RF-09	RF-10	RF-11
RU-01	☑		☑			☑					
RU-02		☑	☑								
RU-03				☑		☑					
RU-04							☑				
RU-05											
RU-06										☑	
RU-07											☑
RU-08								☑	☑		
RU-09								☑			
RU-10											
RU-11							☑				
RU-12					☑						
RU-13											
RU-14						☑			☑		
RU-15											☑
RU-16		☑	☑								

Tabla 25: Matriz de trazabilidad 1

RU/RF	RF-12	RF-13	RF-14	RF-15	RF-16	RF-17	RF-18	RF-19	RF-20	RF-21	RF-22
RU-01											
RU-02											
RU-03											
RU-04											
RU-05		☑						☑			
RU-06	☑										
RU-07		☑							☑		
RU-08											
RU-09					☑						
RU-10			☑	☑	☑						
RU-11										☑	
RU-12											
RU-13											☑
RU-14	☑	☑				☑	☑			☑	
RU-15						☑			☑		
RU-16											

Tabla 26: Matriz de trazabilidad 2

RU/RF	RF-23	RF-24	RF-25	RF-26	RF-27	RF-28	RF-29
RU-01					☑	☑	☑
RU-02					☑	☑	☑
RU-03					☑		☑
RU-04	☑	☑			☑		☑
RU-05					☑		☑
RU-06			☑		☑		☑
RU-07					☑		☑
RU-08					☑		☑
RU-09					☑		☑
RU-10					☑		☑
RU-11		☑			☑		☑
RU-12					☑	☑	☑
RU-13					☑	☑	☑
RU-14			☑		☑		☑
RU-15				☑	☑		☑
RU-16					☑	☑	☑

Tabla 27: Matriz de trazabilidad 3

4.4 Requisitos No Funcionales

Los requisitos no funcionales no describen funcionalidades específicas, sino cómo debe comportarse el sistema. A continuación, se detallan los diferentes tipos:

4.4.1 Disponibilidad (DP)

Indica el tiempo que el sistema debe estar en funcionamiento sin interrupciones.

ID	Requisitos de disponibilidad
DP-01	El sistema debe estar disponible la mayoría del tiempo, garantizando un servicio continuo.
DP-02	El sistema deberá alertar a los usuarios de las labores de mantenimiento y la detención de este.
DP-03	En caso de caída del sistema, este debe recuperarse en un tiempo máximo de 5 minutos.
DP-04	La API debe ser accesible en todo momento sin interrupciones prolongadas.

Tabla 28: Requisitos de disponibilidad

4.4.2 Seguridad (SG)

Asegura la protección de los datos y accesos al sistema.

ID	Requisitos de seguridad
SG-01	La autenticación de usuarios debe realizarse mediante tokens JWT para garantizar sesiones seguras.
SG-02	Los *endpoints* de la API deben requerir autenticación para evitar accesos no autorizados.
SG-03	El sistema debe proteger la información de los usuarios mediante encriptación de contraseñas con un algoritmo seguro (bcrypt o similar).
SG-04	Los datos sensibles, como contraseñas, no deben almacenarse en formato plano en la base de datos.
SG-05	Se deben aplicar permisos y roles para restringir el acceso a ciertas funcionalidades según el tipo de usuario.

Tabla 29: Requisitos de seguridad

4.4.3 Integridad (IT)

Garantiza la consistencia y confiabilidad de los datos:

ID	Requisitos de integridad
IT-01	El sistema debe garantizar la consistencia de los datos almacenados en la base de datos, evitando duplicidades o datos inconsistentes.
IT-02	Todas las transacciones deben seguir el principio ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad) para asegurar que los cambios en la base de datos sean fiables
IT-03	La plataforma debe manejar correctamente los errores en el procesamiento de datos, evitando corrupción de información.

IT-04	La API debe implementar validaciones de datos para garantizar la integridad de la información recibida.

Tabla 30: Requisitos de integridad

4.4.4 Usabilidad (US)

Se enfoca en la experiencia del usuario y facilidad de uso.

ID	Requisitos de usabilidad
US-01	La interfaz debe ser intuitiva y fácil de usar para los diferentes tipos de usuarios (futbolistas, entrenadores, clubes).
US-02	La plataforma debe permitir la navegación fluida entre las diferentes secciones sin confusión.
US-03	Se deben proporcionar mensajes de error y confirmación claros y comprensibles para el usuario.
US-04	La aplicación debe contar con un diseño responsive para adaptarse a distintos dispositivos (móviles, tablets y ordenadores).
US-05	La aplicación debe minimizar la cantidad de pasos necesarios para realizar acciones clave como el registro, la búsqueda de vacantes o el envío de solicitudes.

Tabla 31: Requisitos de usabilidad

4.4.5 Robustez (RB)

Define la capacidad del sistema para manejar errores y fallos sin colapsar.

ID	Requisitos de robustez
RB-01	El sistema debe seguir funcionando correctamente ante caídas parciales de la infraestructura o componentes internos.
RB-02	En caso de fallo en una operación crítica, el sistema debe proporcionar un mensaje de error claro y una alternativa para continuar con la sesión del usuario.

RB-03	La plataforma debe poder recuperar datos en caso de fallo inesperado o reinicio del servidor.
RB-04	La API debe manejar excepciones de manera controlada para evitar interrupciones inesperadas del servicio.

Tabla 32: Requistos de robustez

4.4.6 Rendimiento (RD)

Define tiempos de respuesta y eficiencia del sistema.

ID	Requisitos de rendimiento
RD-01	Las peticiones a la API no deben tardar más de 2 segundos en responder bajo carga normal.
RD-02	La interfaz debe cargar en un tiempo menor a 3 segundos en conexiones estándar.
RD-03	La base de datos debe estar optimizada para manejar grandes volúmenes de información sin afectar el rendimiento.
RD-04	Las búsquedas y filtrados deben ejecutarse en menos de 2 segundos en condiciones normales de uso.

Tabla 33: Requisitos de rendimiento

4.4.7 Escalabilidad (EC)

Asegura que el sistema pueda crecer sin perder eficiencia.

ID	Requisitos de escalabilidad
EC-01	El sistema debe estar diseñado para soportar un aumento progresivo en la cantidad de usuarios sin degradar su rendimiento.
EC-02	La arquitectura debe permitir la integración de nuevas funcionalidades sin necesidad de grandes modificaciones.
EC-03	La API debe estar preparada para manejar múltiples peticiones simultáneas sin afectar su rendimiento.

Tabla 34: Requisitos de escalabilidad

4.4.8 Portabilidad (PR)

Es la capacidad del sistema para ejecutarse en distintos entornos o dispositivos.

ID	Requisitos de portabilidad
PR-01	La plataforma debe ser compatible con distintos navegadores (Chrome, Firefox, Edge, Safari).
PR-02	La aplicación debe poder ejecutarse tanto en dispositivos móviles como en ordenadores sin problemas de compatibilidad.
PR-03	La arquitectura debe permitir su despliegue en diferentes entornos (local, desarrollo, producción) sin cambios significativos.

Tabla 35: Requisitos de portabilidad

4.5 Requisitos de información

Los requisitos de información definen todos los datos que manejará el sistema, cómo estarán organizados y cómo deben almacenarse para que todo funcione de manera eficiente y segura. Es importante que la información fluya correctamente entre los diferentes módulos y que las relaciones entre los datos sean claras y bien estructuradas. Para esto, se detallan las entidades principales, sus atributos y la manera en que se conectan entre sí.

Además, se documentan las estructuras de datos utilizadas, explicando cada tabla, campo y sus restricciones para garantizar coherencia e integridad en la base de datos. Como apoyo visual, se incluirá un modelo entidad-relación que representará gráficamente la estructura del sistema y cómo se relacionan sus distintos elementos. También se añadirá un diccionario de datos que describirá cada campo de manera detallada, facilitando la comprensión y mantenimiento del sistema en el futuro.

4.5.1 Modelo entidad-relación

El siguiente diagrama representa el modelo entidad-relación del sistema. Un pequeño apunte: los atributos de cada entidad están en una única figura redonda por la magnitud del modelo:

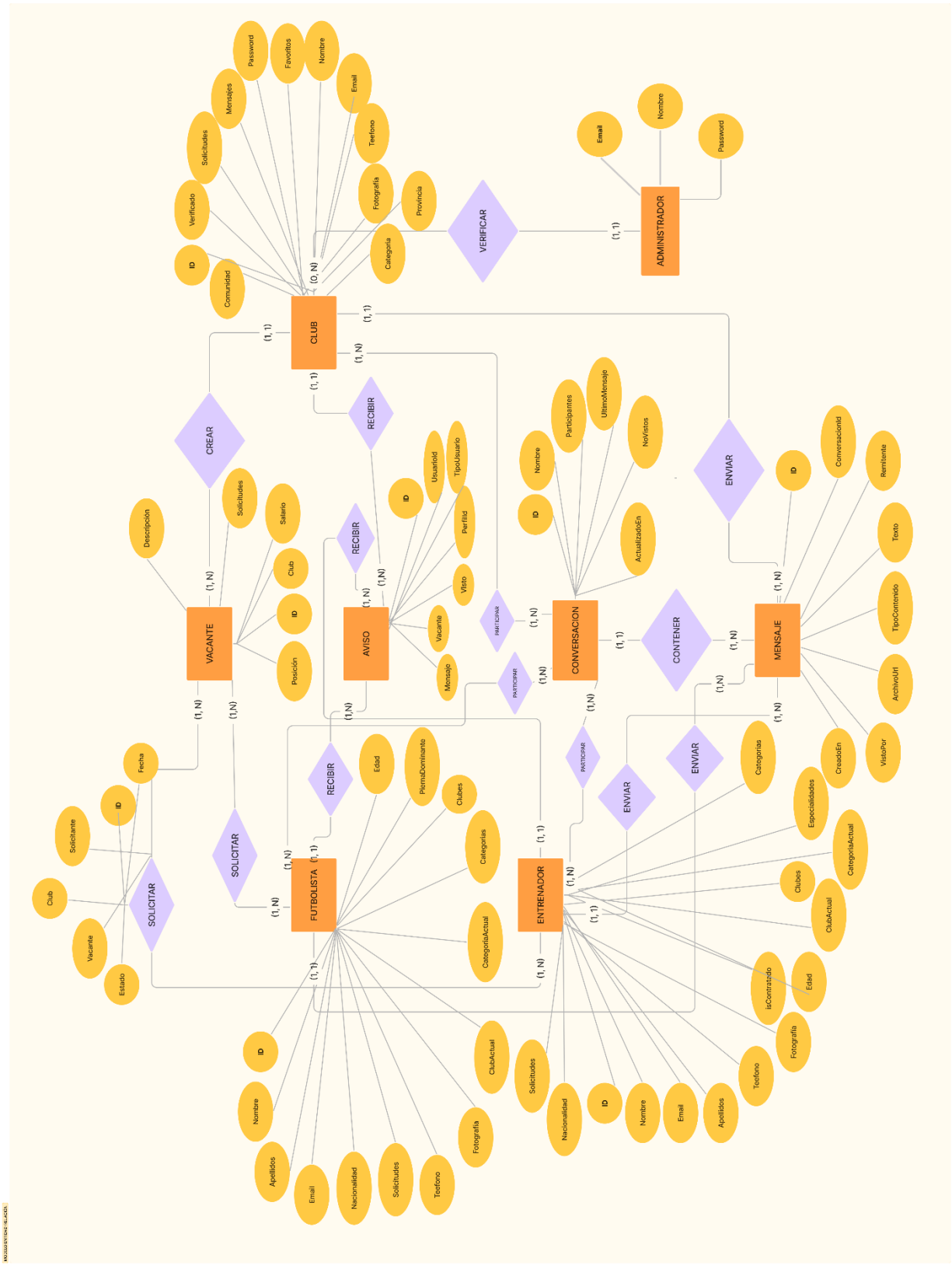


Ilustración 18: Modelo Entidad-Relación

4.5.2 Diccionario de datos

El diccionario de datos es un documento clave que define en detalle todos los datos que manejará el sistema. Sirve como referencia para los desarrolladores, administradores de bases de datos y cualquier persona involucrada en el diseño e implementación del proyecto. En esta sección se describen todas las entidades del sistema, sus atributos, tipos de datos, restricciones (si son únicos, obligatorio.) y relaciones con otras entidades. Esto permite estandarizar la estructura de la información y evitar inconsistencias en el manejo de los datos. A continuación, se detallan las tablas con la información de cada entidad:

Entidad Futbolista			
Atributo	**Tipo**	**Único**	**Requerido**
Id	ObjectId	si	si
nombre	string	no	si
apellidos	string	no	si
edad	number	no	si
email	string	si	si
telefono	string	no	no
fotografía	Object: { url: string, public_id: string }	no	no
clubActual	Club	no	no
categoriaActual	string	no	no
posiciones	[string]	no	si
piernaDominante	string	no	si
clubes	[Club]	no	no
categorías	[string]	no	no
nacionalidad	String	no	si
solicitudes	[Solicitud]	no	no
mensajes	[Mensaje]	no	no
password	String	no	si
favoritos	[ObjectId]	no	no

Tabla 36: Diccionario de datos - futbolista

Entidad Entrenador			
Atributo	**Tipo**	**Único**	**Requerido**
Id	ObjectId	si	si
nombre	string	no	si
apellidos	string	no	si
edad	number	no	si
email	string	si	si
telefono	string	no	no
fotografía	Object: { url: string, public_id: string }	no	no
clubActual	Club	no	no
categoriaActual	string	no	no
especialidades	[string]	no	si
isContratado	boolean	no	si
clubesEspecialidad	[Club]	no	no
categoríasEspacialidad	[string]	no	no
nacionalidad	string	no	si
solicitudes	[Solicitud]	no	no
mensajes	[Mensaje]	no	no
password	string	no	si
favoritos	[ObjectId]	no	no

Tabla 37: Diccionario de datos - entrenadorTabla 38: Diccionario de datos - club

Entidad Club			
Atributo	**Tipo**	**Único**	**Requerido**
Id	ObjectId	si	si
nombre	string	no	si
email	string	si	si
telefono	string	no	no
fotografía	Object: { url: string, public_id: string }	no	no
categoria	string	no	si
comunidad	string	no	si
provincia	string	no	si
verificado	boolean	no	si
solicitudes	[Solicitud]	no	no
mensajes	[Mensaje]	no	no
password	string	no	si
favoritos	[ObjectId]	no	no

Entidad Administrador			
Atributo	**Tipo**	**Único**	**Requerido**
nombre	string	no	si
email	string	si	si
password	string	no	si

Tabla 39: Diccionario de datos - administrador

Entidad Aviso			
Atributo	**Tipo**	**Único**	**Requerido**
Id	ObjectId	si	si
usuarioId *	ObjectId	si	si
tipoUsuario *	enum [futbolista, entrenador, club]	no	si
perfilId *	ObjectId	si	si
tipoPerfil *	enum [futbolista, entrenador, club]	no	si
mensaje	string	no	si
visto	boolean	no	si
vacante *	Vacante	si	no

Tabla 40: Diccionario de datos - aviso

*usuarioId: Se refiere al usuario destinatario del aviso.

*tipoUsuario: Se refiere al tipo de usuario del destinatario del aviso.

*perfilId: Se refiere al usuario del que se habla en el aviso.

*tipoPerfil: Se refiere al tipo de usuario del que se habla en el aviso.

*vacante: Se refiere a la vacante a la cual un usuario ha aplicado.

Relación Solicitar			
Atributo	**Tipo**	**Único**	**Requerido**
Id	ObjectId	si	si
solcitante	Object: { id: ObjectId, tipo: enum [futbolista, entrenador, club] }	no	si
club	Club	si	si
vacante	Vacante	si	si
fecha	Date	no	si
estado	enum [Pendiente, En Proceso, Aceptada, Rechazada]	no	si

Tabla 41: Diccionario de datos – solicitud

Entidad Vacante			
Atributo	**Tipo**	**Único**	**Requerido**
Id	ObjectId	si	si
club	Club	si	si
destinatario	enum [futbolista, entrenador]	no	si
posicion	string	no	si
descripcion	string	no	no
salario	number	no	no
solicitudes	[Solicitud]	no	si

Tabla 42: Diccionario de datos - vacantes

Entidad Conversación *			
Atributo	**Tipo**	**Único**	**Requerido**
Id	ObjectId	si	si
nombre	string	no	si
participantes	Object: { usuarioId: ObjectId, tipoUsuario: enum [futbolista, entrenador, club] }	no	si
ultimoMensaje	Mensaje	si	si
actualizadoEn	Date	no	si
noVistos	Object: { usuarioId: ObjectId, cantidad: Number }	no	si

Tabla 43: Diccionario de datos - conversación

* La entidad Conversación está preparada para ser escalable y poder admitir chats entre grupos y no solo entre individuales.

Entidad Mensaje			
Atributo	**Tipo**	**Único**	**Requerido**
Id	ObjectId	si	si
conversacionId	Conversacion	si	si
remitente	Object: { usuarioId: ObjectId, tipoUsuario: enum [futbolista, entrenador, club] }	no	si
texto	string	no	si
tipoContenido	enum [texto, imagen, archivo]	no	si
archivoUrl	string	no	no

vistoPor	Object: { usuarioId: ObjectId, tipoUsuario: enum [futbolista, entrenador, club] }	no	no
creadoEn	Date	no	si

Tabla 44: Diccionario de datos - mensaje

Relación Participar _conversación			
Atributo	**Tipo**	**Único**	**Requerido**
Id	ObjectId	si	si
conversacion	string	no	si
participantes	Object: { usuarioId: ObjectId, tipoUsuario: enum [futbolista, entrenador, club] }	no	si

5. Diseño

En este capítulo se presenta el diseño del sistema, abordando la estructura y organización de sus componentes clave. Se detalla la arquitectura del software, incluyendo la distribución de responsabilidades entre el *frontend* y el *backend*, así como la integración con servicios externos.

5.1 Arquitecturas

A continuación, se muestran los dos principales diagramas sobre las arquitecturas del sistema (física y lógica):

5.1.1 Arquitectura física

La arquitectura física representa cómo y dónde se implementa el sistema en términos de *hardware* y redes. Su función es garantizar que la aplicación tenga el rendimiento, la seguridad y la disponibilidad adecuados para los usuarios.

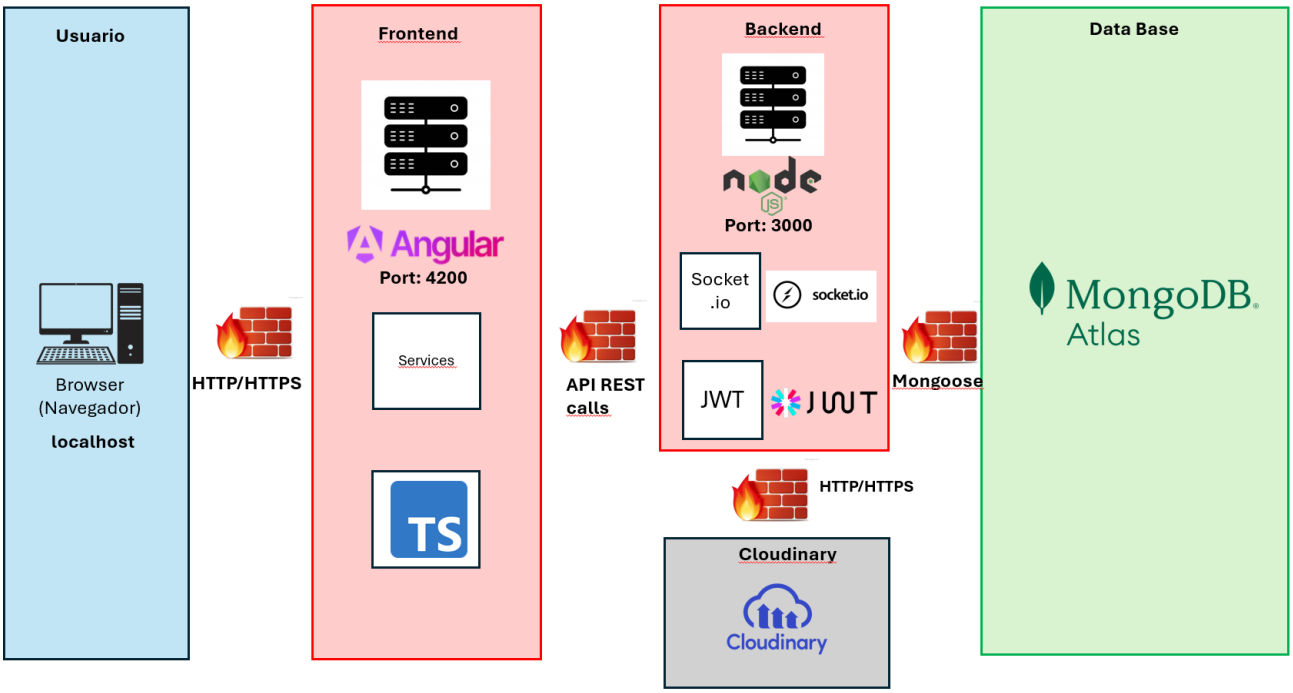


Ilustración 19: Arquitectura física

5.1.2 Arquitectura lógica

La arquitectura lógica es la organización del sistema a nivel de *software*, donde se define cómo interactúan los distintos módulos y componentes sin considerar la infraestructura física en la que se ejecutan. Su propósito es estructurar el software de manera que sea mantenible, escalable y eficiente, permitiendo el desarrollo independiente de los distintos módulos. Además, facilita la comprensión del flujo de información dentro del sistema.

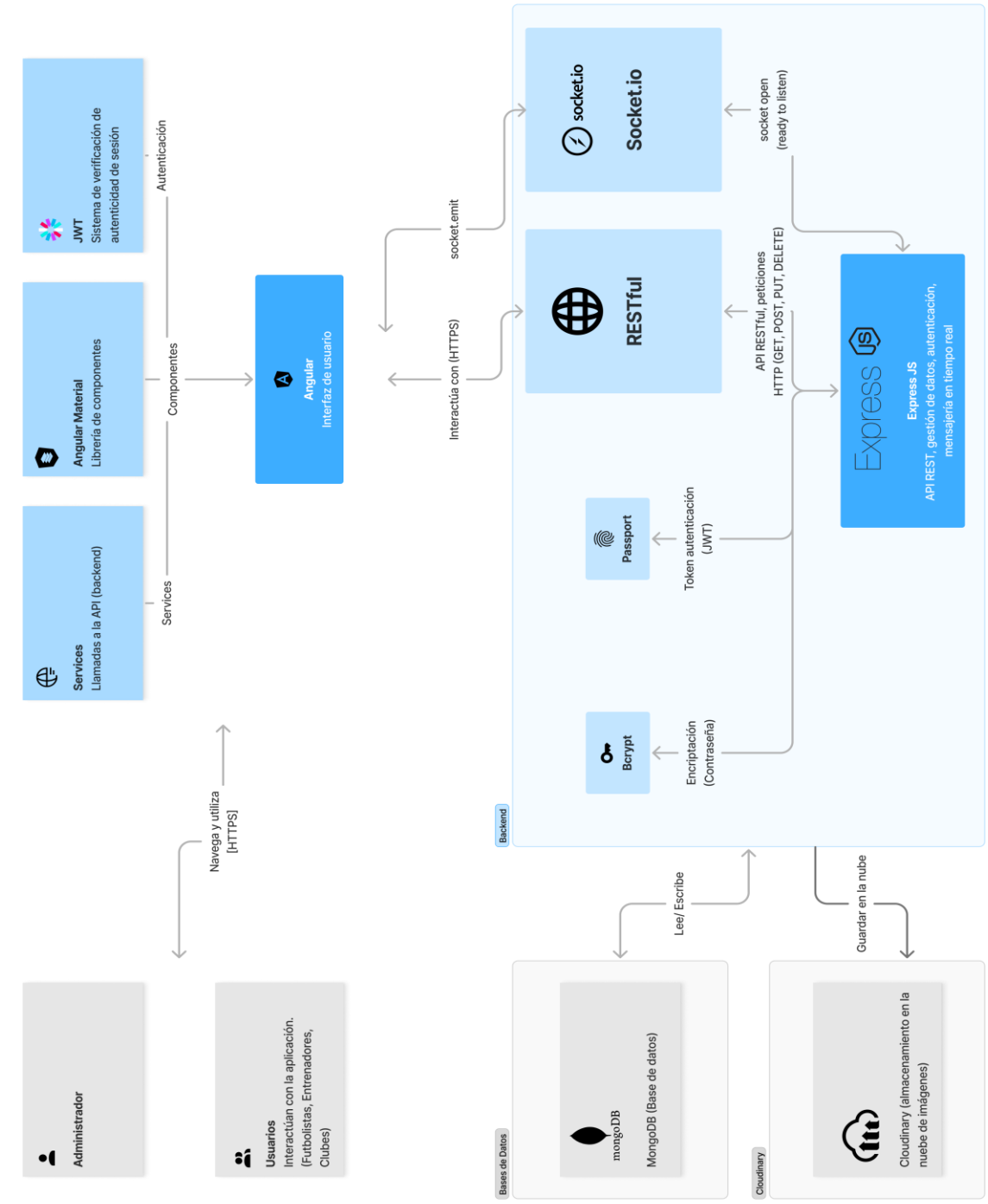


Ilustración 20: Arquitectura lógica

5.2 Diagramas de secuencia

5.2.1 Secuencia Registro

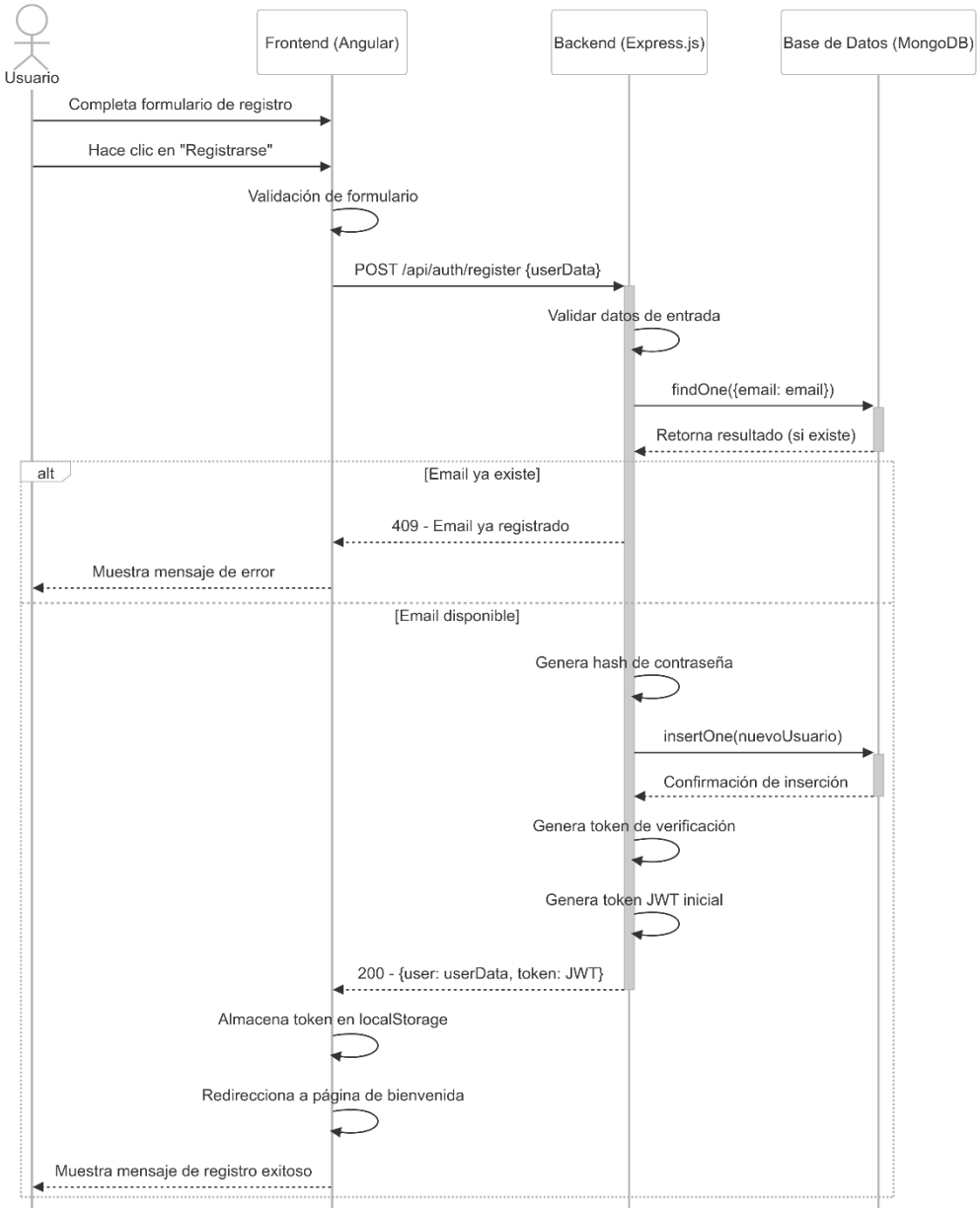


Ilustración 21: Diagrama de secuencia (registro)

5.2.2 Secuencia Inicio de sesión

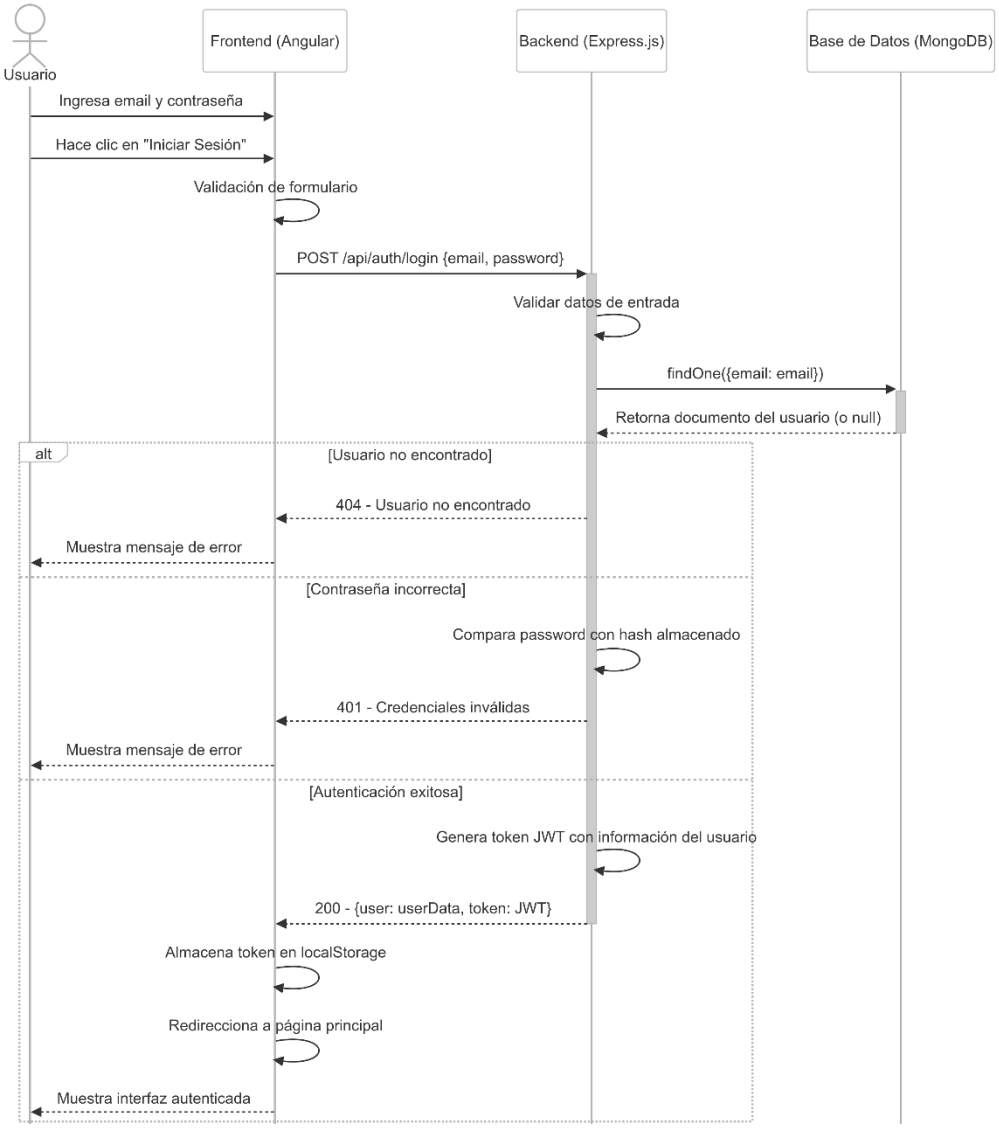


Ilustración 22: Diagrama de secuencia (inicio de sesión)

5.2.3 Secuencia Cierre de sesión

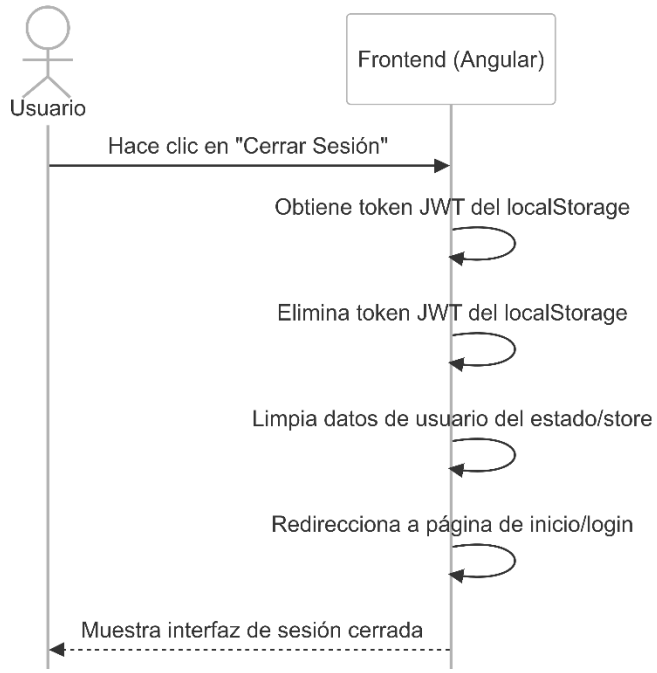


Ilustración 23: Diagrama de secuencia (cierre de sesión)

5.2.4 Secuencia Verificar club

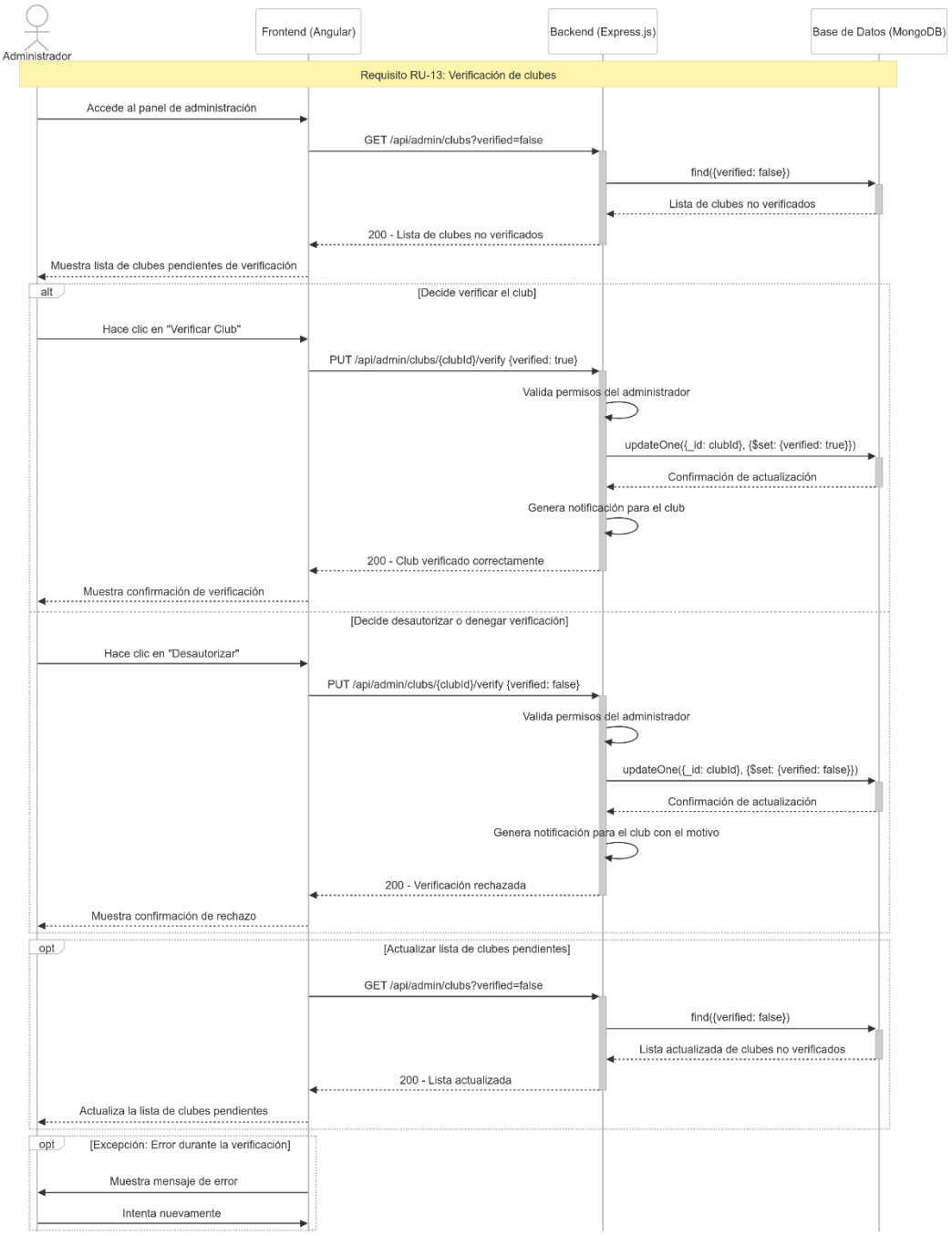


Ilustración 24: Diagrama de secuencia (verificar club)

5.2.5 Secuencia Enviar solicitud vacante

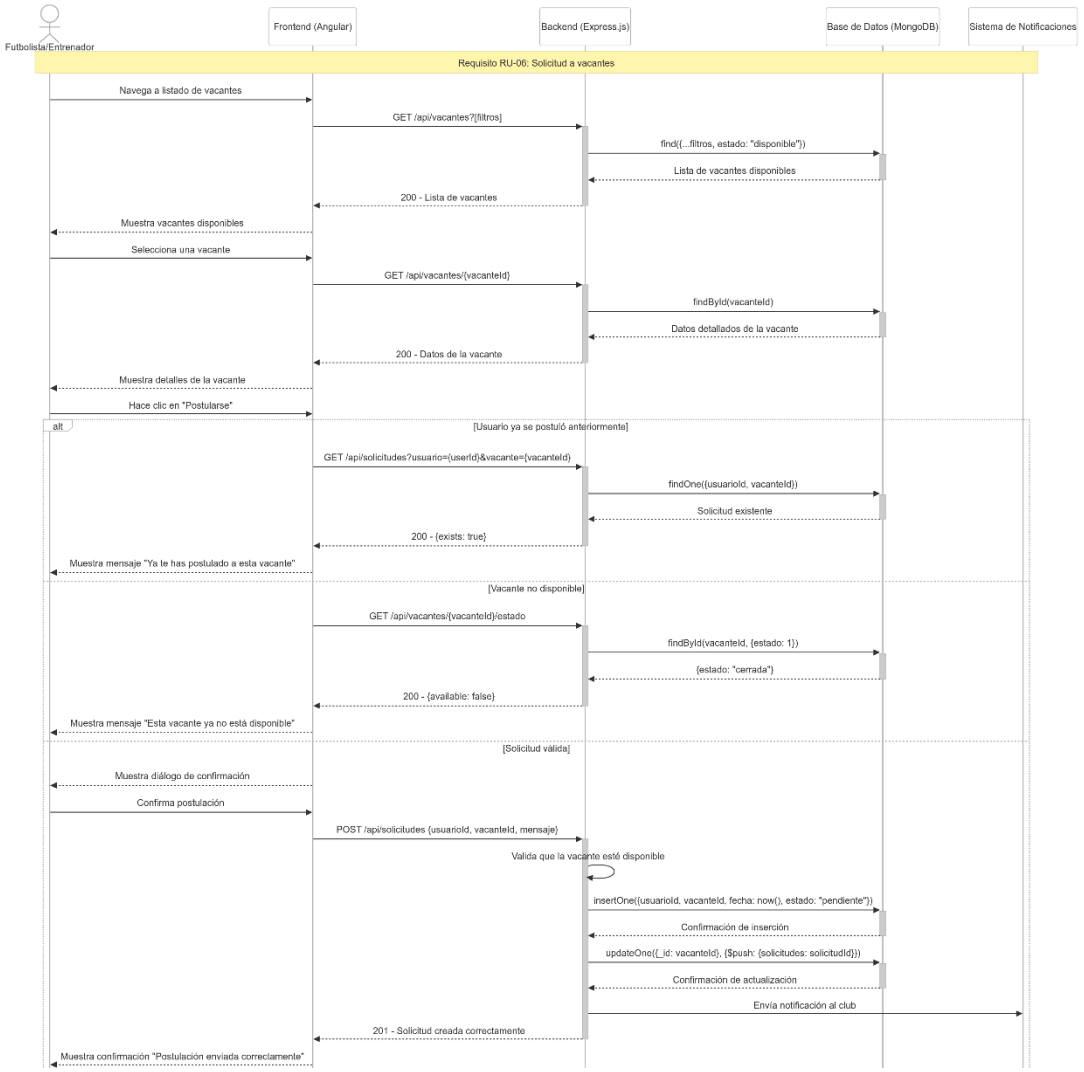


Ilustración 25: Diagrama de secuencia (enviar solicitud)

5.3 Diseño de la interfaz

El diseño de la interfaz de usuario es el proceso de estructurar y definir la apariencia visual y la experiencia interactiva de una aplicación, asegurando que los usuarios puedan navegar y utilizar sus funcionalidades de manera intuitiva y eficiente. El objetivo principal es optimizar la usabilidad y accesibilidad, proporcionando una experiencia agradable y fluida.

5.3.1 Paleta de colores

Una paleta de colores es un conjunto de tonalidades seleccionadas estratégicamente para definir la identidad visual de un diseño, interfaz o marca. Su función principal es garantizar coherencia y armonía en los elementos gráficos, facilitando la comunicación visual y mejorando la experiencia del usuario. Los colores elegidos para esta aplicación son los siguientes:

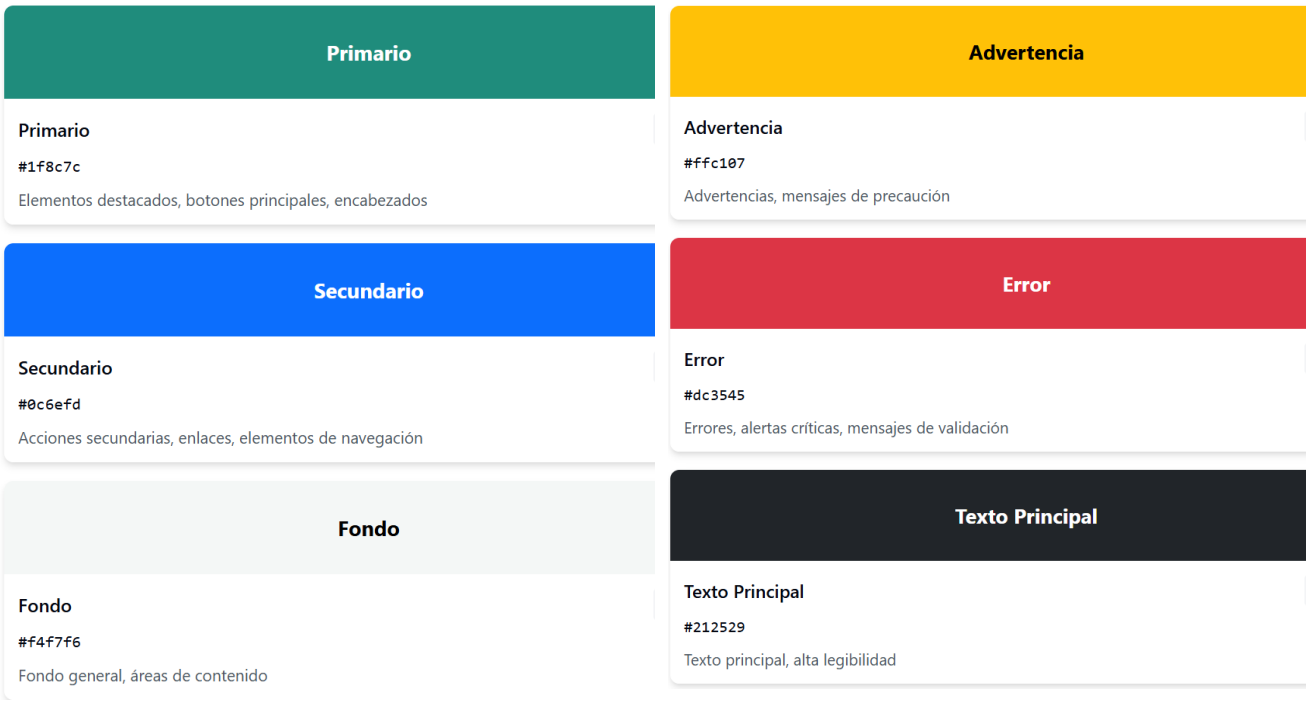


Ilustración 26: Paleta de colores

5.3.2 Diseño de interfaces

Inicio de sesión	
Descripción	A través de esta pantalla los usuarios registrados en el sistema pueden iniciar sesión introduciendo su email y contraseña.
Activación	- Al entrar a la aplicación - Al darle a “Iniciar sesión” en la pantalla de selección de usuario a registrar. - Al cerrar sesión.
Boceto	
Eventos	1. Pulsando en el botón “Iniciar sesión” se envía el formulario con las credenciales para verificarlas. 2. Pulsando en “Regístrate” te redirige a la vista de selección de usuario

Tabla 45: Interfaz inicio de sesión

Selección de usuario para registro	
Descripción	A través de esta pantalla los usuarios que quieran registrarse en la aplicación podrán elegir qué tipo de usuario son.
Activación	- Al pulsar “Regístrate” en la página de inicio de sesión.
Boceto	
Eventos	1. Selección de usuario 2. Una vez seleccionada una opción te deja pulsar en “Crear cuenta” y pasas al formulario específico del usuario

Tabla 46: Interfaz selección de usuario

Formulario de registro	
Descripción	A través de esta pantalla los usuarios que quieran registrarse en la aplicación deberán cumplimentar los datos
Activación	- Al pulsar “Crear cuenta” en la página de selección de usuarios
Boceto	
Eventos	1. Introducir dato a dato los campos obligatorios. 2. Una vez completados todos los campos pasando el filtro de validación el usuario podrá pulsar sobre el botón “Crear cuenta” para registrar el usuario en el sistema. 3. Una vez registrado se iniciará sesión automáticamente.

Tabla 47: Interfaz formulario de registroTabla 48: Interfaz pantalla principal

Pantalla principal (*Dashboard*)	
Descripción	A través de esta pantalla los usuarios que hayan iniciado sesión podrán visualizar en este *dashboard* los datos principales.
Activación	- Después de finalizar el inicio de sesión. - Al pulsar en “Home” en la *navbar*.
Boceto	
Eventos	1. Visualizar las últimas vacantes activas en detalle pulsando sobre ellas. 2. Visualizar los últimos avisos en detalle pulsando sobre ellos. 3. En caso de ser un usuario club, eliminar o editar las últimas vacantes.

Perfil de un futbolista	
Descripción	A través de esta pantalla los usuarios que hayan iniciado sesión podrán visualizar los detalles de su perfil y sus datos
Activación	- Al pulsar en la imagen de perfil en la barra de navegación.
Boceto	
Eventos	1- Pulsando sobre editar perfil se pueden modificar los datos sobre el mismo formulario de registro. 2- Pulsando en eliminar cuenta se eliminaría la cuenta en el sistema

Tabla 49: Intergaz perfil de usuario

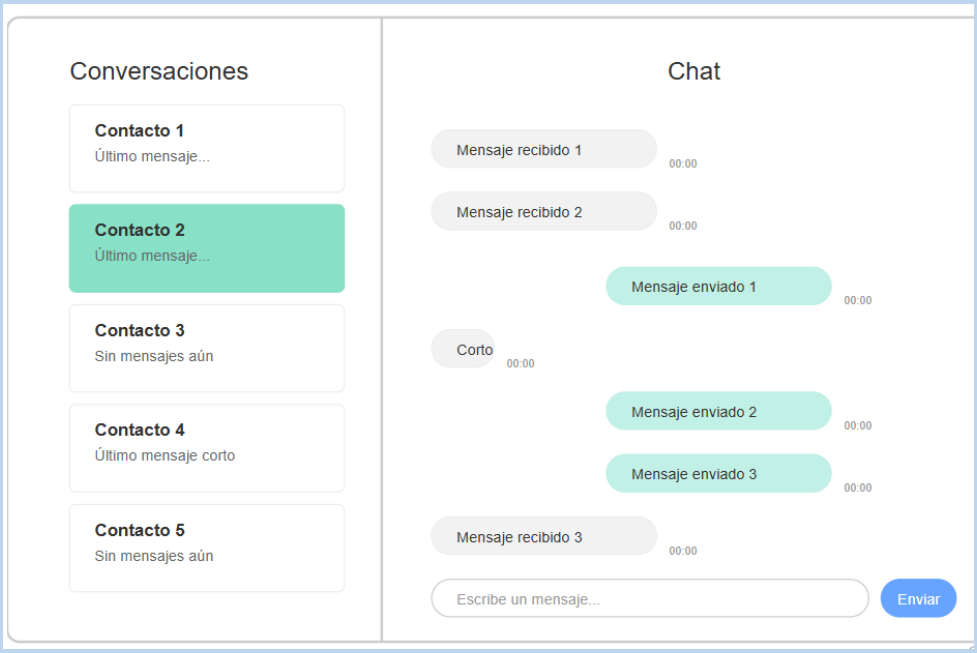
Chat	
Descripción	A través de esta pantalla los usuarios que hayan iniciado sesión podrán compartir
Activación	- Al pulsar en el icono del chat la barra de navegación. - A través de el detalle de cada perfil.
Boceto	
Eventos	1- Pulsando sobre el nombre de un contacto se accede a la conversación con ese usuario. 2- Pulsando en la barra de los mensajes se puede escribir el mensaje que se quiere enviar. 3- Pulsando en enviar se enviará el mensaje y el usuario receptor lo recibirá en tiempo real.

Tabla 50: Interfaz chat

5.4 Diseño *schemas* a partir de modelos

En Express.js, los *schemas* se utilizan para definir la estructura y las reglas de validación de los documentos que se almacenarán en las colecciones de MongoDB. Estos modelos permiten organizar y gestionar los datos de manera eficiente dentro de la aplicación. A continuación, se presentan los *schemas* de los modelos implementados:

```
const mongoose = require('mongoose');
const { Schema } = mongoose;

const AdministradorSchema = new Schema({
    nombre: { type: String, required: true },
    email: { type: String, required: true, unique: true},
    password: { type: String, required: true }, // Campo para la contraseña
}, { timestamps: true });

const Administrador = mongoose.model('Administrador', AdministradorSchema);
module.exports = Administrador;
```

Ilustración 27: Modelo Administrador

```
const mongoose = require('mongoose');
const { Schema } = mongoose;

const AvisoSchema = new Schema({
    usuarioId: { type: Schema.Types.ObjectId, required: true },
    tipoUsuario: { type: String, enum: ['club', 'futbolista', 'entrenador'], required: true },
    perfilId: { type: Schema.Types.ObjectId, required: true },
    tipoPerfil: { type: String, enum: ['futbolista', 'entrenador', 'club'], required: true },
    mensaje: { type: String, required: true },
    visto: { type: Boolean, default: false },
    createdAt: { type: Date, default: Date.now },
    vacante: { type: Schema.Types.ObjectId, required: false },
}, { timestamps: true });

const Aviso = mongoose.model('Aviso', AvisoSchema);
module.exports = Aviso;
```

Ilustración 28: Modelo Aviso

```
const ConversacionSchema = new Schema({
    nombre: { type: String },
    participantes: [{
        tipoUsuario: { type: String, enum: ['Futbolista', 'Entrenador', 'Club'], required: true },
        usuarioId: { type: Schema.Types.ObjectId, required: true, refPath: 'participantes.tipoUsuario' }
    }],
    ultimoMensaje: { type: Schema.Types.ObjectId, ref: 'Mensaje' },
    actualizadoEn: { type: Date, default: Date.now },
    noVistos: [{
        usuarioId: { type: Schema.Types.ObjectId, refPath: 'participantes.tipoUsuario' },
        cantidad: { type: Number, default: 0 }
    }]
}, { timestamps: true });

const Conversacion = mongoose.model('Conversacion', ConversacionSchema);
module.exports = Conversacion;
```

Ilustración 29: Modelo Conversación

```
const mongoose = require('mongoose');
const { Schema } = mongoose;

const ClubSchema = new Schema({
    nombre: { type: String, required: true, unique: true },
    email: { type: String, required: true, unique: true},
    telefono: { type: String },
    fotografia: {
        url: { type: String },
        public_id: { type: String }
    },
    categoria: { type: String, required: true },
    comunidad: { type: String, required: false },
    provincia: { type: String, required: false },
    solicitudes: [{ type: Schema.Types.ObjectId, ref: 'Solicitud' }],
    mensajes: [{ type: Schema.Types.ObjectId, ref: 'Mensaje' }],
    vacantes: [{ type: Schema.Types.ObjectId, ref: 'Vacante' }],
    verificado: { type: Boolean, default: false },
    password: { type: String, required: true },
    favoritos: [{ type: Schema.Types.ObjectId }]
}, { timestamps: true });

const Club = mongoose.model('Club', ClubSchema);
module.exports = Club;
```

Ilustración 30: Modelo Club

```
const { Schema } = mongoose;

const EntrenadorSchema = new Schema({
    nombre: { type: String, required: true },
    apellidos: { type: String, required: true },
    edad: { type: Number, required: true },
    email: { type: String, required: true, unique: true},
    telefono: { type: String },
    fotografia: {
        url: { type: String },
        public_id: { type: String }
    },
    nacionalidad: { type: String, required: true },
    clubActual: { type: Schema.Types.ObjectId, ref: 'Club', required: false },
    categoriaActual: { type: String, required: false },
    especialidades: { type: [String], required: true },
    clubesEspecialidad: { type: [String], required: false },
    categoriasEspecialidad: { type: [String], required: false },
    solicitudes: [{ type: Schema.Types.ObjectId, ref: 'Solicitud' }],
    mensajes: [{ type: Schema.Types.ObjectId, ref: 'Mensaje' }],
    isContratado: { type: Boolean, default: false},
    password: { type: String, required: true },
    favoritos: [{ type: Schema.Types.ObjectId }]
}, { timestamps: true });

const Entrenador = mongoose.model('Entrenador', EntrenadorSchema);
module.exports = Entrenador;
```

Ilustración 31: Modelo Entrenador

```
const mongoose = require('mongoose');
const { Schema } = mongoose;

const FutbolistaSchema = new Schema({
    nombre: { type: String, required: true },
    apellidos: { type: String, required: true },
    edad: { type: Number, required: true },
    email: { type: String, required: true, unique: true},
    telefono: { type: String },
    fotografia: {
        url: { type: String },
        public_id: { type: String }
    },
    clubActual: { type: Schema.Types.ObjectId, ref: 'Club', required: false },
    categoriaActual: { type: String, required: false },
    posiciones: { type: [String], required: true },
    piernaDominante: { type: String, required: true },
    clubes: { type: [String], required: false },
    categorias: { type: [String], required: false },
    nacionalidad: { type: String, required: true },
    solicitudes: [{ type: Schema.Types.ObjectId, ref: 'Solicitud' }],
    mensajes: [{ type: Schema.Types.ObjectId, ref: 'Mensaje' }],
    password: { type: String, required: true },
    favoritos: [{ type: Schema.Types.ObjectId }]
}, { timestamps: true });

const Futbolista = mongoose.model('Futbolista', FutbolistaSchema);
module.exports = Futbolista;
```

Ilustración 32: Modelo Futbolista

```
const mongoose = require('mongoose');
const { Schema } = mongoose;

const MensajeSchema = new Schema({
    conversacionId: { type: Schema.Types.ObjectId, ref: 'Conversacion', required: true },
    remitente: {
        tipoUsuario: { type: String, enum: ['Futbolista', 'Entrenador', 'Club'], required: true },
        usuarioId: { type: Schema.Types.ObjectId, required: true, refPath: 'remitente.tipoUsuario' },
    },
    texto: { type: String },
    tipoContenido: { type: String, enum: ['texto', 'imagen', 'archivo'], default: 'texto' },
    archivoUrl: { type: String }, // URL del archivo si es imagen o documento
    vistoPor: [{
        tipoUsuario: { type: String, enum: ['Futbolista', 'Entrenador', 'Club'], required: true },
        usuarioId: { type: Schema.Types.ObjectId, required: true, refPath: 'vistoPor.tipoUsuario' }
    }],
    creadoEn: { type: Date, default: Date.now }
}, { timestamps: true });

const Mensaje = mongoose.model('Mensaje', MensajeSchema);
module.exports = Mensaje;
```

Ilustración 33: Modelo Mensaje

```
const mongoose = require('mongoose');
const { Schema } = mongoose;

const SolicitudSchema = new Schema({
    solicitante: {
        id: { type: Schema.Types.ObjectId, required: true },
        tipo: { type: String, enum: ['futbolista', 'entrenador'], required: true }
    },
    club: { type: Schema.Types.ObjectId, ref: 'Club', required: true }, // Referencia al club
    vacante: { type: Schema.Types.ObjectId, ref: 'Vacante', required: true }, // Referencia a la vacante a la que se aplica
    fecha: { type: Date, required: true },
    estado: { type: String,  enum: ['Pendiente', 'En Proceso', 'Aceptada', 'Rechazada'], default: 'Pendiente', required: true }
}, { timestamps: true });

const Solicitud = mongoose.model('Solicitud', SolicitudSchema);
module.exports = Solicitud;
```

Ilustración 34: Modelo Solicitud

```
const mongoose = require('mongoose');
const { Schema } = mongoose;

const VacanteSchema = new Schema({
    club: { type: Schema.Types.ObjectId, ref: 'Club', required: true },
    destinatario: { type: String, enum: ['Futbolista', 'Entrenador'], required: true },
    posicion: { type: String, required: true },
    descripcion: { type: String, required: false },
    salario: { type: Number, required: false },
    solicitudes: [{ type: Schema.Types.ObjectId, ref: 'Solicitud' }],
}, { timestamps: true });


const Vacante = mongoose.model('Vacante', VacanteSchema);
module.exports = Vacante;
```

Ilustración 35: Modelo Vacante

5.5 Diagrama de *endpoints*

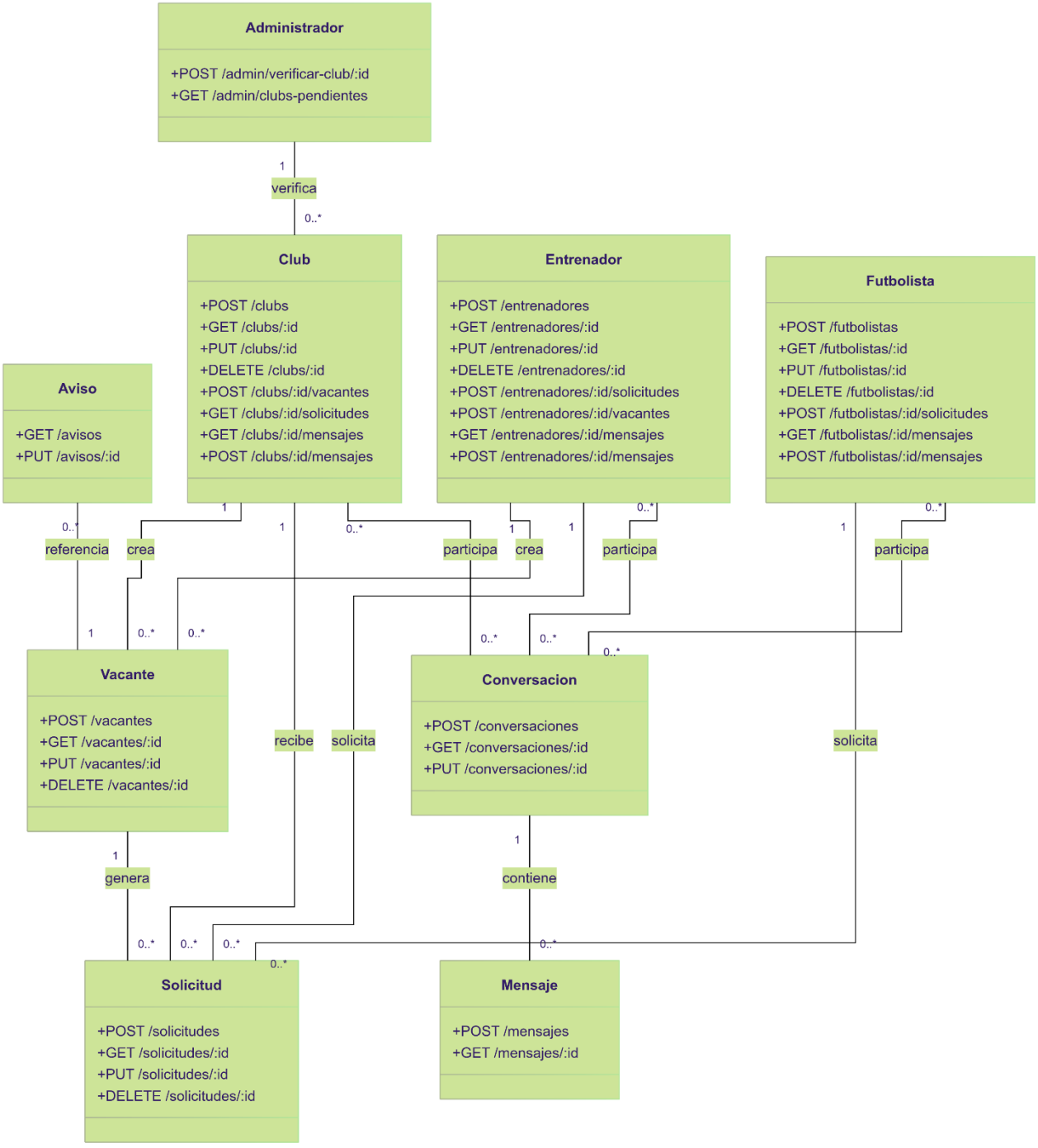


Ilustración 36: Diagrama de endpoints

Los *endpoints* son las *URLs* específicas de la API que permiten la comunicación entre el *frontend* (Angular) y el *backend* (Node.js/Express). Cada *endpoint* representa una ruta que expone una funcionalidad del servidor, como crear, leer, actualizar o eliminar datos (operaciones CRUD).

Estos *endpoints* estructuran la lógica de negocio y permiten que los diferentes tipos de usuarios (clubes, futbolistas, entrenadores, administradores) interactúen con la plataforma de forma segura y organizada.

Las operaciones **CRUD** (por sus siglas en inglés: *Create, Read, Update, Delete*) son las acciones básicas que se pueden realizar sobre los datos en una aplicación:

- ***Create* (Crear):** Añadir nuevos datos. Ejemplo: registrar un nuevo usuario o crear una vacante.
- ***Read* (Leer):** Consultar datos existentes. Ejemplo: obtener el perfil de un futbolista o listar todas las vacantes.
- ***Update* (Actualizar):** Modificar datos existentes. Ejemplo: editar la información de un perfil.
- ***Delete* (Eliminar):** Borrar datos. Ejemplo: eliminar una vacante o una solicitud enviada.

Estas operaciones se implementan a través de los *endpoints* del *backend* y son fundamentales para el funcionamiento de cualquier sistema de gestión de datos.

6. Implementación

En este capítulo se explica cómo se ha convertido el diseño en código y cómo se han unido todas las piezas para hacer que LinkByBall funcione. Aquí se detallan las tecnologías utilizadas, la organización del código y cómo se han desarrollado las principales funcionalidades.

Para el *backend*, se ha utilizado **Node.js con Express**, gestionando toda la lógica de negocio y la conexión con la base de datos. En el *frontend*, se ha trabajado con **Angular**, creando una interfaz clara y fácil de usar. También se han integrado servicios como **Cloudinary** para el almacenamiento de imágenes y un sistema de mensajería en tiempo real, que permite a los usuarios comunicarse sin retrasos.

A lo largo de este capítulo, se verá cómo se ha estructurado el proyecto, los retos que han surgido y cómo se han resuelto para construir una plataforma funcional y eficiente.

6.1 Backend

El *backend* es el encargado de gestionar la lógica de negocio, procesar solicitudes y manejar la base de datos. Está desarrollado en Node.js con Express, lo que permite una estructura modular y escalable.

La API sigue un enfoque **RESTful**, facilitando la comunicación con el *frontend* y asegurando una gestión eficiente de los datos. Además, se ha implementado autenticación con **JWT (*JSON Web Tokens*)** para garantizar la seguridad de los usuarios. También se han integrado servicios externos como Cloudinary para el almacenamiento de imágenes y **WebSockets** para la mensajería en tiempo real.

A lo largo del desarrollo, se ha priorizado la optimización del rendimiento y la organización del código, siguiendo buenas prácticas para mantener el sistema limpio y fácil de mantener. La estructura llevada a cabo es la siguiente:

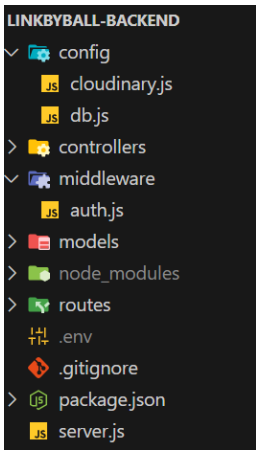


Ilustración 37: Estructura backend

El archivo principal es el `server.js`, es donde se inicializa la API a través del puerto indicado, donde está configurado el CORS (para únicamente permitir llamadas desde orígenes controlados), donde se hace efectiva la conexión a la base de datos y donde se inicia la escucha a través de los `socket.io` para manejar los eventos del chat integrado en la aplicación. El server.js se alimenta de las variables de entorno que se encuentran en el archivo `.env` y que se maneja a través de la carpeta `config`**,** este archivo tiene todas las URLs y claves que no queremos que sean visibles como son la URL de conexión a la base de datos de mongo, las claves de conexión a cloudinary y el secreto de JWT. En el `package.json` están todas las dependencias necesarias a instalar para el correcto funcionamiento del sistema, todas estas dependencias se instalan mediante el comando `npm -i`. El archivo **`.gitignore`** contiene todos aquellos directorios o archivos que no queremos que se incluyan cuando subimos el proyecto al repositorio de git o github.

En la arquitectura implementada, se ha utilizado el modelo **MRC** (*Model-Route-Controller*), una variante del conocido **MVC** (*Model-View-Controller*), especialmente utilizada en el desarrollo de **APIs RESTful**.

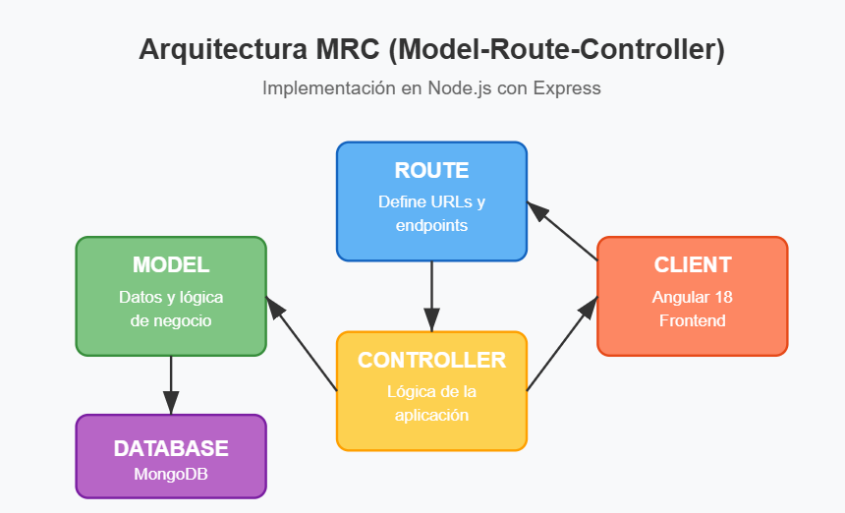


Ilustración 38: Arquitectura modelo MRC

La organización del proyecto se divide en tres capas principales:

- **Modelos (**`Models`**):** Contienen la estructura y atributos de los objetos que se almacenan en la base de datos, definiendo su esquema y relaciones.
- **Controladores (**`Controllers`**):** Gestionan la lógica de negocio y procesan las peticiones HTTP (`GET`, `POST`, `PUT`, `DELETE`), interactuando con los modelos y enviando respuestas al cliente.
- **Rutas (**`Routes`**):** Definen los endpoints de la API y asocian cada ruta con su correspondiente controlador, asegurando que cada solicitud se dirija correctamente.

Además, la carpeta `routes` incluye un archivo `index.js`, encargado de centralizar e indexar todas las rutas, facilitando su gestión y mantenimiento dentro de la aplicación. Esta estructura modular permite un desarrollo escalable, organizado y fácil de mantener.

Otra de las carpetas más importantes, sobre todo cuando hablamos de la seguridad, es la de `middleware`. Este archivo inyecta una capa de seguridad a cada una de las rutas. Al iniciar sesión en el sistema el *backend* envía un `token` de sesión con una duración máxima de validez de 30 minutos para que el usuario pueda interactuar de forma segura con la base de datos. Antes de poder acceder al *endpoint* de la ruta a la que se le ha hecho la llamada, se aplica una función (capa `middleware`) que desencripta el `token` de sesión requerido para validar si es un usuario válido, si en caso de querer acceder a funciones de administrador puede hacerlo, etc.

6.2 Frontend

Para el desarrollo del *frontend*, se ha usado **Angular**, un *framework* de **TypeScript** (ts) que ayuda a construir aplicaciones web dinámicas y organizadas. Se ha seguido una estructura modular para que el código sea fácil de mantener y ampliar en el futuro sin que todo se vuelva un caos.

El archivo principal de la aplicación en Angular es el `main.ts`, encargado de iniciar el proyecto. A través de este archivo, se carga el módulo raíz y se arranca la aplicación, invocando el componente inicial **AppComponent (**`app.component.ts`**)**.

6.2.1 Archivos principales

Estos son los principales archivos del proyecto:

- `app.component.ts`: Es el **componente principal** de la aplicación. Define la estructura básica de la interfaz de usuario y actúa como el punto de entrada visual de la app. Desde aquí se gestionan otros componentes y se define la plantilla principal.

- `app.module.ts`: Es el **módulo raíz** de Angular. En este archivo se declaran los componentes, directivas y pipes que forman parte de la aplicación. También se importan módulos externos y se configuran proveedores de servicios.
- `app-routing.module.ts`: Es el módulo de enrutamiento, encargado de gestionar la navegación entre las distintas vistas de la aplicación. Aquí se definen las rutas y se asocian con los componentes correspondientes.
- `index.html`: Es el documento base en el que Angular inyecta la aplicación. Contiene el `<app-root>`, que es el selector donde se monta el componente principal. También es el lugar donde se cargan los estilos globales y scripts esenciales.

6.2.2 Environments

El proyecto está diseñado para ser escalable y seguro, garantizando una gestión robusta de los cambios y mejoras que puedan implementarse en futuras versiones. Para ello, la arquitectura en Angular incluye tres entornos diferentes (*environments*) que permiten configurar de manera flexible las conexiones con la API y gestionar variables sensibles sin exponerlas directamente en el código. Estos entornos son:

- `Default` **(**local**):** Utilizado para el desarrollo en `localhost`, permitiendo pruebas sin afectar otros entornos.
- `Environment.dev`**:** Configuración destinada a un entorno de desarrollo más avanzado, ideal para pruebas internas antes de desplegar cambios en producción.
- `Environment.prod`**:** Entorno optimizado para producción, donde la aplicación funciona de manera estable con configuraciones seguras y optimizadas para los usuarios finales.

Ilustración 39: Entornos frontend

Esta separación facilita el mantenimiento, mejora la seguridad y permite realizar pruebas en entornos controlados antes de llevar cualquier cambio a producción. La utilización de uno u otro entorno se gestiona en el `angular.json` y mediante el comando introducido a la hora de construir el proyecto `ng build --configuration=production`.

6.2.3 Conexión con el *backend*

Para comunicarse con el *backend*, se han creado servicios específicos en Angular que se encargan de gestionar todas las peticiones a la API RESTful. Estos servicios funcionan como una capa intermedia entre los componentes de la aplicación y el servidor, permitiendo que la información se maneje de manera más eficiente y estructurada. En lugar de realizar llamadas directas al *backend* desde los componentes, los servicios encapsulan esta lógica, haciendo que el código sea más limpio, reutilizable y fácil de mantener.

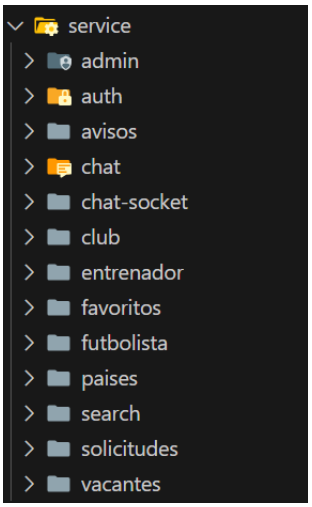


Ilustración 40: Servicios frontend

Cada servicio está diseñado para gestionar una parte específica del sistema, lo que ayuda a mantener la separación de responsabilidades dentro del proyecto. Por ejemplo, hay servicios dedicados a la autenticación de usuarios, a la gestión de vacantes, al manejo de perfiles o a la comunicación entre usuarios a través del *chat*. Gracias a esta organización, cualquier cambio en la API o en la lógica de negocio puede realizarse en un solo lugar sin afectar otras partes de la aplicación.

Estos servicios utilizan `HttpClient` de Angular para realizar peticiones HTTP como `GET`, `POST`, `PUT` y `DELETE`. De esta manera, los componentes solo se preocupan por mostrar la información y delegan la lógica de obtención o envío de datos a los servicios. Un ejemplo claro es el servicio de autenticación, que gestiona el inicio de sesión y el registro de usuarios. Los componentes simplemente llaman a los métodos de este servicio cuando necesitan autenticar a alguien, sin preocuparse por la implementación interna de las peticiones.

El uso de servicios aporta múltiples ventajas. En primer lugar, hace que el código sea más modular y reutilizable, ya que la misma funcionalidad puede ser utilizada en diferentes partes de la aplicación sin necesidad de repetir código. Además, facilita el mantenimiento y la escalabilidad del proyecto, permitiendo realizar cambios de manera más sencilla. Por último, ayuda a mejorar el rendimiento, ya que se pueden implementar estrategias como la caché de datos o la gestión centralizada de errores, reduciendo el número de llamadas innecesarias al servidor.

```typescript
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';
import { environment } from '../../../environments/environment';

@Injectable({
  providedIn: 'root'
})
export class FutbolistaService {

  private baseUrl = `${environment.apiUrl}`;  // Cambia esto por tu URL de backend

  constructor(private http: HttpClient) { }

  // Obtener el perfil del futbolista
  getFutbolistaProfile(): Observable<any> {
    return this.http.get(`${this.baseUrl}/futbolista/perfil`);
  }

  // Obtener un futbolista por ID
  getFutbolistaById(id: string): Observable<any> {
    return this.http.get(`${this.baseUrl}/futbolista/perfil/${id}`);
  }

  // Actualizar el perfil del futbolista
  updateFutbolistaProfile(futbolistaData: any): Observable<any> {
    return this.http.put(`${this.baseUrl}/futbolista/actualizar`, futbolistaData);
  }

  // Eliminar el perfil del futbolista
  deleteFutbolistaProfile(): Observable<any> {
    return this.http.delete(`${this.baseUrl}/futbolista/perfil`);
  }
}
```

Ilustración 41: Ejemplo de servicio frontend

6.2.4 Vistas y componentes

En Angular, los componentes son la base de la interfaz de usuario y están formados por tres archivos principales: el archivo TypeScript (`.ts`), el archivo de plantilla HTML y la hoja de estilos (`.scss`). Cada uno cumple una función específica dentro del desarrollo de la aplicación y juntos permiten construir vistas dinámicas y reutilizables.

El archivo TypeScript **(`.ts`)** es donde se define la lógica del componente. Aquí se importan los módulos necesarios, se declaran variables, se implementan funciones y se maneja la interacción con los servicios. También es el lugar donde se definen los `@Input` y `@Output`, que permiten la comunicación entre componentes. Además, este archivo se encarga de gestionar los eventos que se activan en la interfaz, como clics en botones o cambios en formularios, procesando la información y actualizando la vista cuando sea necesario.

El archivo HTML (`.html`) contiene la estructura visual del componente, es decir, el código que define lo que el usuario verá en pantalla. En este archivo se utilizan directivas de Angular como `*ngFor` para recorrer listas, `*ngIf` para mostrar u ocultar elementos según condiciones y [(`ngModel`)] para la vinculación de datos en formularios. También se integran los componentes hijos, permitiendo la creación de interfaces modulares y organizadas.

Por último, el archivo **SCSS (.`scss`)** se encarga del diseño y la apariencia del componente. Aquí se definen los estilos personalizados usando *Sass*, una extensión de CSS que permite utilizar variables, anidaciones y funciones avanzadas para optimizar la escritura del código. Gracias a los estilos encapsulados de Angular, cada componente puede tener su propio diseño sin afectar el resto de la aplicación, lo que facilita la mantenibilidad y evita conflictos de estilos.

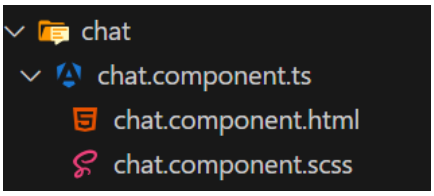


Ilustración 42: Ejemplo estructura componente

6.2.5 Otros aspectos

Se ha trabajado en que la app responda rápido y de forma reactiva, actualizando los datos en tiempo real sin necesidad de recargar la página. También se han optimizado las solicitudes para que la carga sea más rápida y la experiencia de usuario sea lo más fluida posible.

Uno de los puntos clave es el chat en tiempo real, que permite a futbolistas, entrenadores y clubes comunicarse al instante. Se ha implementado un sistema basado en eventos para enviar y recibir mensajes sin demoras. Para desarrollar el chat, se ha implementado un sistema basado en **WebSockets** (`Sockets.IO`), lo que permite el envío y recepción de mensajes de manera instantánea, sin necesidad de realizar peticiones constantes al servidor, como ocurriría con un sistema basado únicamente en HTTP. Gracias a esto, los usuarios pueden mantener conversaciones fluidas sin experimentar retrasos innecesarios.

Cada mensaje enviado es procesado a través del *backend*, que se encarga de almacenarlo en la base de datos para que las conversaciones queden registradas y puedan ser recuperadas en cualquier momento. Además, se han implementado eventos específicos para notificar a los usuarios cuando reciben un nuevo mensaje, incluso si no tienen el chat abierto en ese momento.

Las notificaciones avisan a los usuarios sobre cualquier cambio importante, como nuevas vacantes, actualizaciones de sus solicitudes o cambios en los perfiles que siguen. Esto les permite estar al tanto de todo sin tener que estar revisando constantemente la app.

7. Pruebas

En el desarrollo de una aplicación, garantizar su correcto funcionamiento es un paso fundamental antes de su despliegue. Para ello, se realizan diferentes tipos de pruebas que permiten detectar posibles fallos y mejorar la calidad del software. En este capítulo, se presentan las pruebas realizadas en el proyecto, enfocándose en dos enfoques principales: **pruebas de caja negra y pruebas de caja blanca**.

7.1 Pruebas de caja negra

Las pruebas de caja negra son una técnica de prueba de software en la que se evalúa el comportamiento del sistema sin necesidad de conocer su estructura interna o código fuente. Se centran en analizar las entradas y salidas del software para verificar que responde correctamente según los requisitos establecidos. Este enfoque permite identificar fallos en la funcionalidad, usabilidad y cumplimiento de especificaciones, asegurando que el sistema cumpla con las expectativas del usuario final.

Registro de usuario futbolista	
Objetivo	Registrar al usuario en el sistema para poder hacer uso de las funcionalidades del mismo.
Precondiciones	El usuario no debe estar registrado con el mismo correo electrónico que fuese a introducir en el formulario.
Datos de entrada	Nombre: Adrián Apellidos: Contreras Sanz Correo electrónico: adri@mail.com Contraseña: EstudianteUva Confirmar contraseña: EstudianteUva Edad: 22 Fotografía: *FILE* Posiciones: [lateral_diestro, carriilero_diestro] Pierna dominante: ambas País de nacimiento: España
Acción esperada	Registrar al usuario en la base de datos correctamente e iniciar sesión automáticamente después.
Resultado	Correcto

Tabla 51: Prueba caja negra (registro futbolista)

Registro de usuario entrenador

Objetivo	Registrar al usuario en el sistema para poder hacer uso de las funcionalidades del mismo.
Precondiciones	El usuario no debe estar registrado con el mismo correo electrónico que fuese a introducir en el formulario.
Datos de entrada	Nombre: Carlo Apellidos: Ancelotti Correo electrónico: carlo@mail.com Contraseña: EntrenadorMadrid Confirmar contraseña: EntrenadorMadrid Edad: 65 Fotografía: *FILE* Especialidades: [primer_entrenador] País de nacimiento: Italia
Acción esperada	Registrar al usuario en la base de datos correctamente e iniciar sesión automáticamente después.
Resultado	Correcto

Tabla 52: Prueba caja negra (registro entrenador)

Registro de usuario club	
Objetivo	Registrar al usuario en el sistema para poder hacer uso de las funcionalidades del mismo.
Precondiciones	El usuario no debe estar registrado con el mismo correo electrónico que fuese a introducir en el formulario.
Datos de entrada	Nombre: Gimnástica Segoviana Categoría: Primera RFEF Correo electrónico: lasego@mail.com Contraseña: VamosSego Confirmar contraseña: VamosSego Fotografía: *FILE* Comunidad autónoma: Castilla y león Provincia: Segovia
Acción esperada	Registrar al usuario en la base de datos correctamente e iniciar sesión automáticamente después.
Resultado	Correcto

Tabla 53: Prueba caja negra (registro club)

Inicio de sesión

Objetivo	Iniciar sesión en el sistema y contar con las funcionalidades disponibles para mi tipo de usuario.
Precondiciones	El usuario debe estar registrado en el sistema.
Datos de entrada	Email: lasego@mail.com Contraseña : VamosSego
Acción esperada	Redirección al *dashboard* personalizado del usuario.
Resultado	Correcto

Tabla 54: Prueba caja negra (inicio de sesión)

Creación de vacante	
Objetivo	Crear una nueva vacante con las especificaciones indicadas a través del formulario.
Precondiciones	El usuario debe estar registrado en el sistema como usuario de tipo club.
Datos de entrada	Destinatario: Futbolista Posición: Carrilero Diestro Descripción: Se busca un carrilero derecho Salario: 1250
Acción esperada	El sistema registra la nueva vacante en la base de datos y después te redirige a la ventana general de todas las vacantes del club.
Resultado	Correcto

Tabla 55: Prueba caja negra (creación vacante)

Envío de mensajes	
Objetivo	Enviar un mensaje a otro usuario del sistema.
Precondiciones	El usuario debe estar registrado en el sistema.
Datos de entrada	Mensaje: Hala Madrid
Acción esperada	El sistema envía el mensaje al usuario destinatario y este lo recibe en tiempo real.
Resultado	Correcto

Tabla 56: Prueba caja negra (envío de mensajes)

7.2 Pruebas de caja blanca

Las pruebas de caja blanca son un enfoque de prueba de software en el que se analiza la estructura interna, el flujo de control y la lógica del código para garantizar su correcto funcionamiento. A diferencia de las pruebas de caja negra, este método requiere conocer el código fuente y permite evaluar aspectos como la cobertura de sentencias, condiciones y caminos de ejecución. Su objetivo es detectar errores en la implementación, optimizar el rendimiento y asegurar que todas las rutas posibles dentro del programa se comporten como se espera.

1. Prueba de conexión con la base de datos: Se verifica que el sistema establece correctamente la conexión con la base de datos, permitiendo operaciones de lectura y escritura sin errores.

2. Pruebas en la autenticación: Se verifica que todas las líneas de código dentro del servicio de autenticación sean ejecutadas al menos una vez. Por ejemplo, al probar el inicio de sesión, se comprueba que la validación del email y la contraseña, la generación del token JWT y la respuesta de éxito o error sean evaluadas correctamente.

3. Prueba de condiciones en la solicitud de vacantes: Se analizan todas las posibles combinaciones de condiciones dentro de la lógica que permite a los futbolistas y entrenadores postularse a una vacante. Por ejemplo, se verifica que un futbolista no pueda postularse dos veces a la misma vacante y que solo los usuarios con el rol adecuado puedan enviar solicitudes.

4. Pruebas en el sistema de favoritos: Se ejecutan todas las rutas posibles dentro del código que permite a los clubes, entrenadores y futbolistas agregar o eliminar favoritos. Se prueba qué sucede cuando un usuario intenta añadir un favorito ya existente o cuando elimina un favorito que no está en su lista.

5. Prueba de flujo en la mensajería interna: Se verifica que el flujo de envío y recepción de mensajes sigue el camino esperado. Por ejemplo, al enviar un mensaje, se comprueba que se guarde correctamente en la base de datos, se actualicen las referencias en las colecciones de los usuarios involucrados y se dispare una notificación si es necesario.

6. Prueba de validación de datos en el registro de usuarios: Se simulan diferentes entradas en el formulario de registro para asegurarse de que se manejan correctamente los casos esperados. Por ejemplo, se prueba qué ocurre cuando se intenta registrar un usuario sin email, con una contraseña débil o con un email ya existente en la base de datos.

8. Documentación de usuario

Este capítulo final se dedica a desarrollar el manual del usuario final, ya sea para un usuario de perfil futbolista, entrenador o club, así como el del usuario administrador.

8.1 Registro de usuario e inicio de sesión

Para poder utilizar la plataforma, es necesario contar con una cuenta de usuario. El proceso de registro es sencillo y permitirá acceder a todas las funcionalidades del sistema.

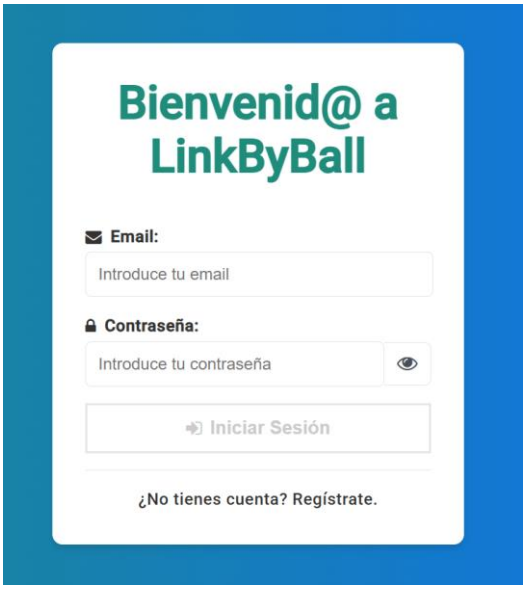


Ilustración 43: Manual incio de sesión

Al abrir la aplicación, en la pantalla de bienvenida se presentan dos opciones: registrarse o iniciar sesión. Si es la primera vez que accedes, deberás crear una cuenta seleccionando la opción de registro. A la hora de registrarte en la aplicación aparecerán tres opciones y para que se habilite la acción del botón “Crear Cuenta” deberá seleccionar una de las opciones

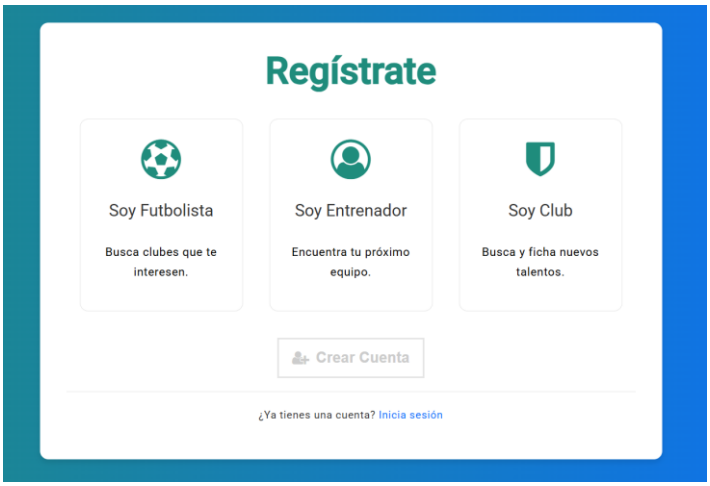


Ilustración 44: Manual selección de usuario

8.1.1 Registro como futbolista

Para poder completar el registro de forma exitosa deberá cumplimentar el siguiente formulario rellenando al menos los campos que aparecen con un asterisco (*), que son aquellos que se consideran obligatorios.

Ilustración 45: Manual registro futbolista

Ilustración 46: Manual registro futbolista 2

Una vez completado el registro se iniciará sesión automáticamente y te redirigirá a la pantalla del *dashboard* general de futbolista.

8.1.2 Registro como entrenador

Para poder completar el registro de forma exitosa deberá cumplimentar el siguiente formulario rellenando al menos los campos que aparecen con un asterisco (*), que son aquellos que se consideran obligatorios.

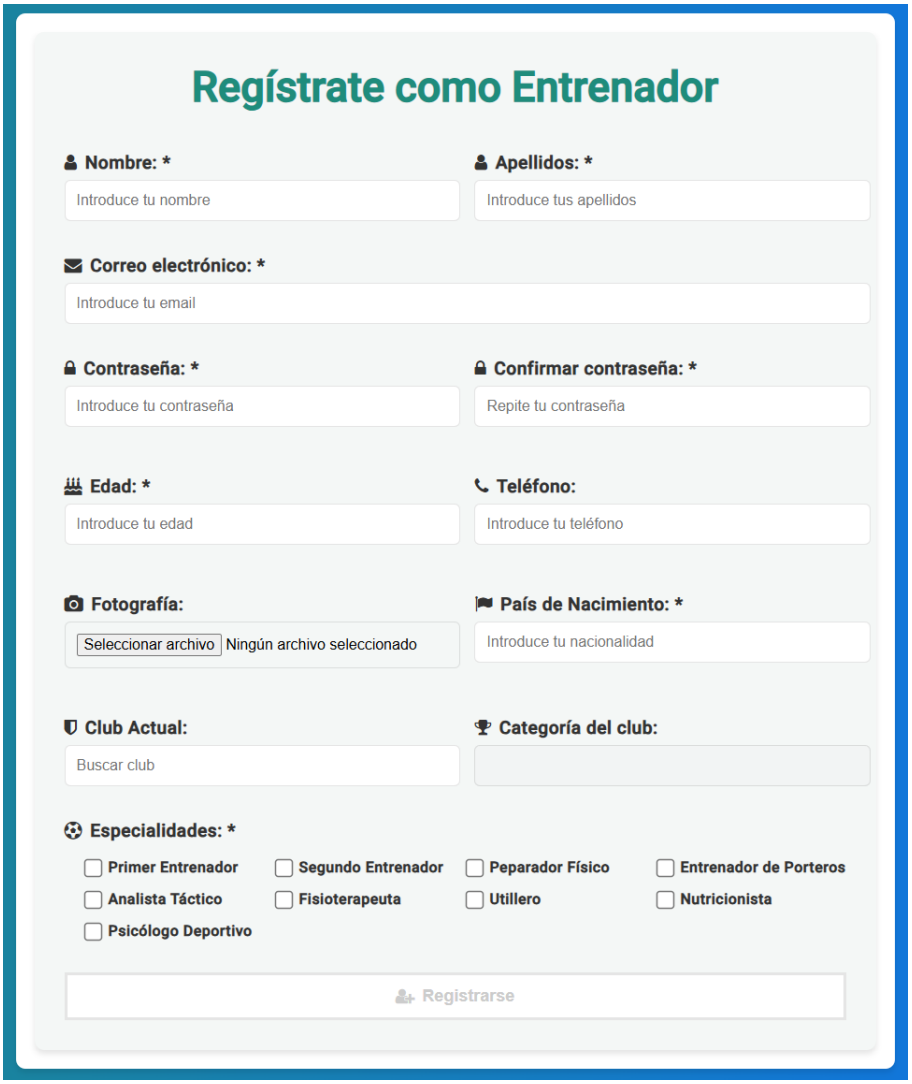


Ilustración 47: Manual registro entrenador

Una vez completado el registro se iniciará sesión automáticamente y te redirigirá a la pantalla del *dashboard* general de entrenador.

8.1.3 Registro como club

Para poder completar el registro de forma exitosa deberá cumplimentar el siguiente formulario rellenando al menos los campos que aparecen con un asterisco (*), que son aquellos que se consideran obligatorios.

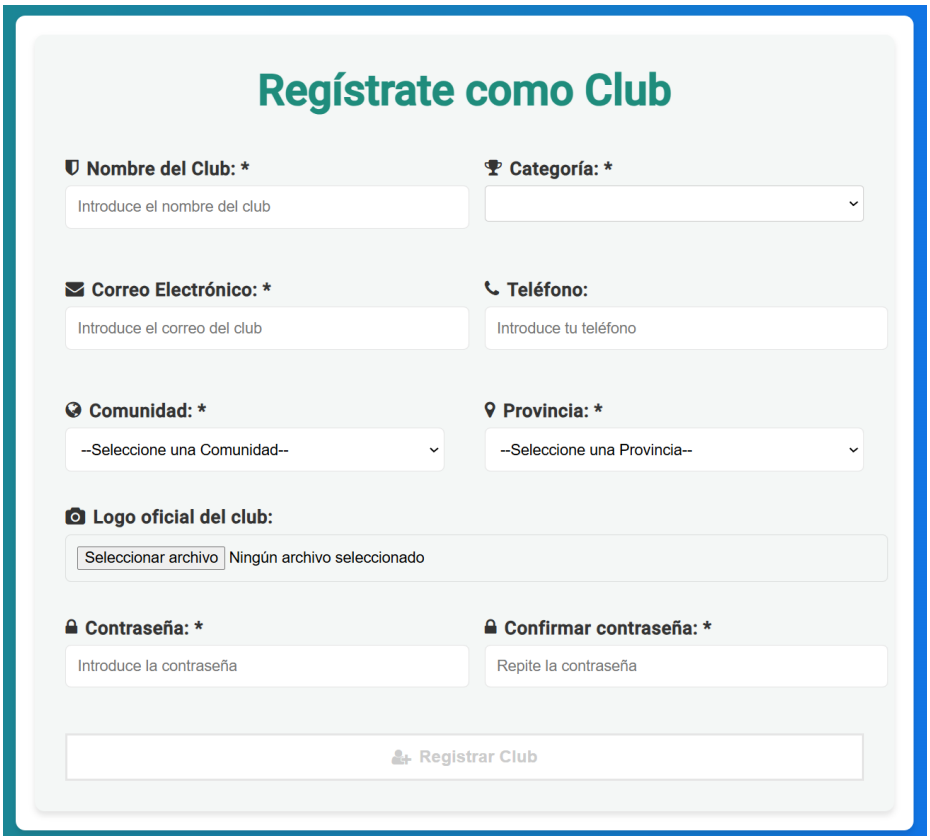


Ilustración 48: Manual registro club

Una vez completado el registro se iniciará sesión automáticamente y te redirigirá a la pantalla del *dashboard* general de club.

IMPORTANTE: Para que un club aparezca en los buscadores, el registro debe ser previamente verificado por el administrador del sistema

8.2 Usuario futbolista ya autenticado

Cuando el usuario ya ha sido autenticado como futbolista lo primero que podrá ver es un *dashboard* con la información general del usuario.

8.2.1 Home

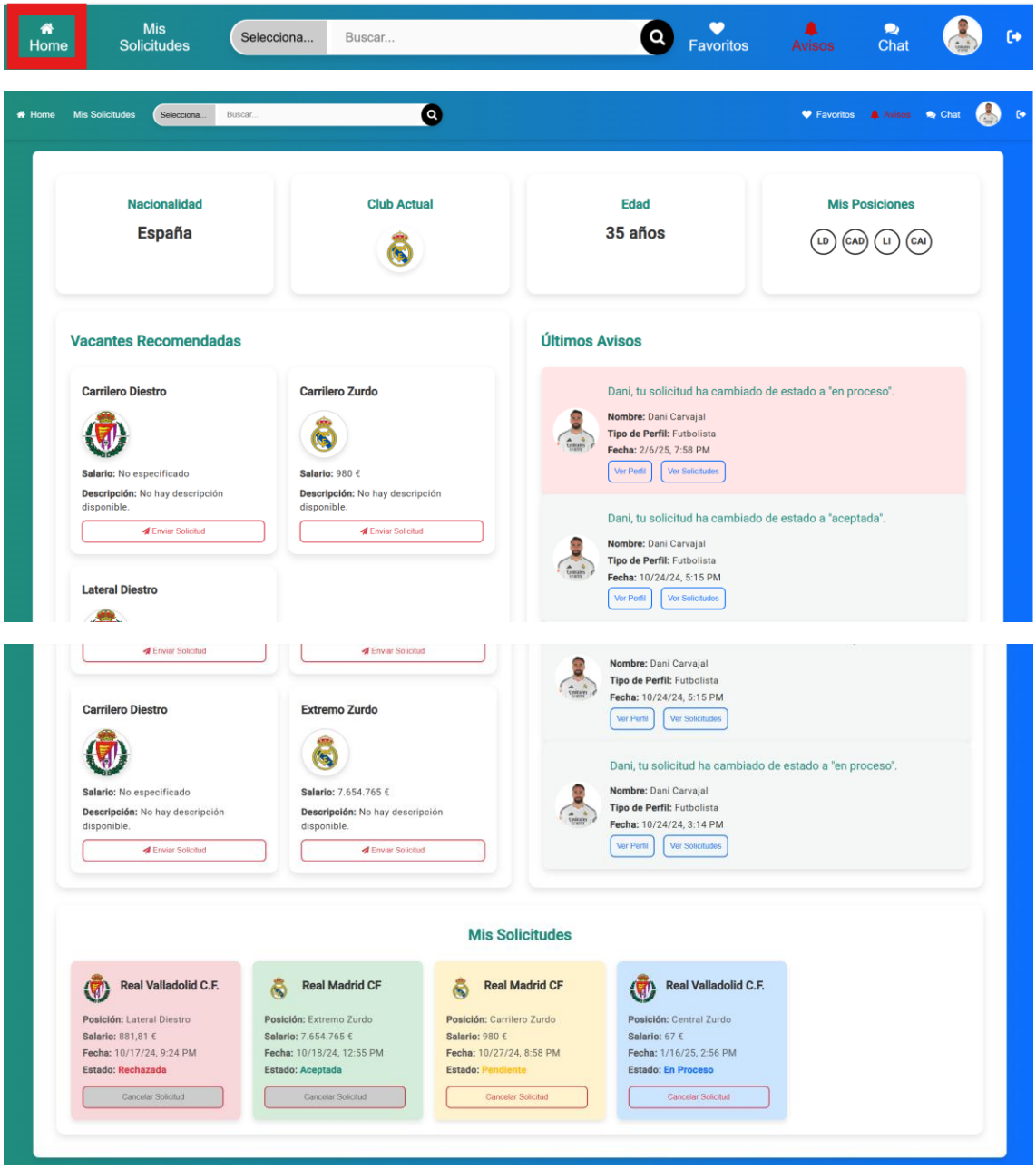


Ilustración 49: Manual dashboard general

La información mostrada es la nacionalidad, club en el que juega actualmente, edad, abreviaturas de las posiciones en las que puede jugar, un listado con las vacantes

recomendadas según las posiciones en las que juega y a través de las cuales se puede solicitar directamente, los tres últimos avisos más recientes (con un fondo rojo para cunado no lo has visto y que se quita en el momento en el que pinchas) y finalmente las solicitudes diferenciadas con distintos colores en función del estado en el que se encuentran.

8.2.2 Mis solicitudes

Se muestra un listado en forma de cards de las solicitudes y al igual que en el dashboard general de la página home, se diferencia el estado por sus colores de fondo.

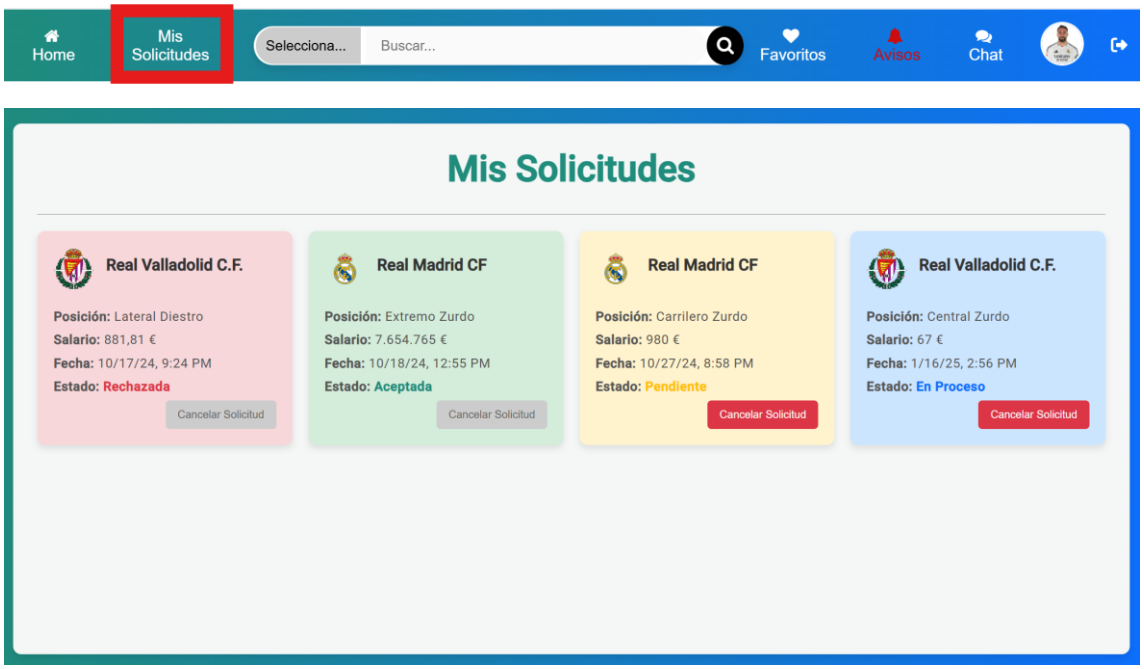


Ilustración 50: Manual mis solicitudes

Puede haber hasta cuatro estados diferentes que son los que se muestran en la imagen superior. Solo se puede cancelar la solicitud si está en “pendiente” o “en proceso”.

8.2.3 Mi perfil

Haciendo click en la imagen del perfil se accede a la vista de “mi perfil” donde se podrá ver la información guardada del usuario, modificar los datos del usuario con el mediante el mismo formulario que el de registro o eliminar la cuenta del sistema.

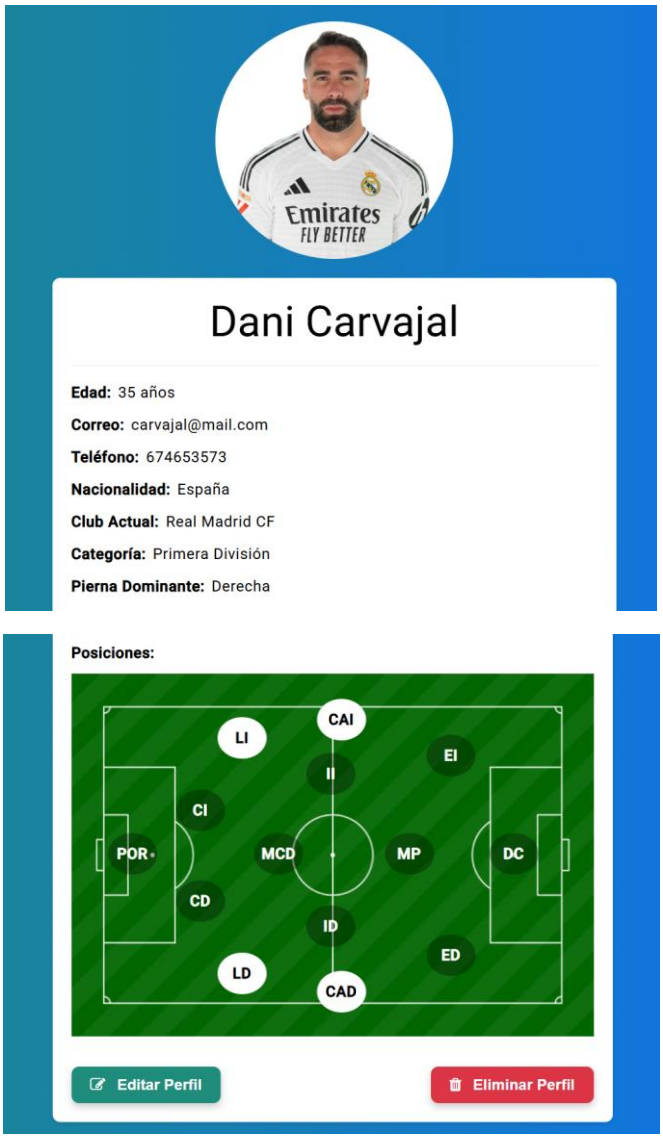


Ilustración 51: Manual mi perfil

Haciendo click en el icono de la *navbar* de al lado del perfil se puede cerrar la sesión.

8.3 Usuario entrenador ya autenticado

Cuando el usuario ya ha sido autenticado como entrenador lo primero que podrá ver es un dashboard con la información general del usuario.

8.3.1 Home

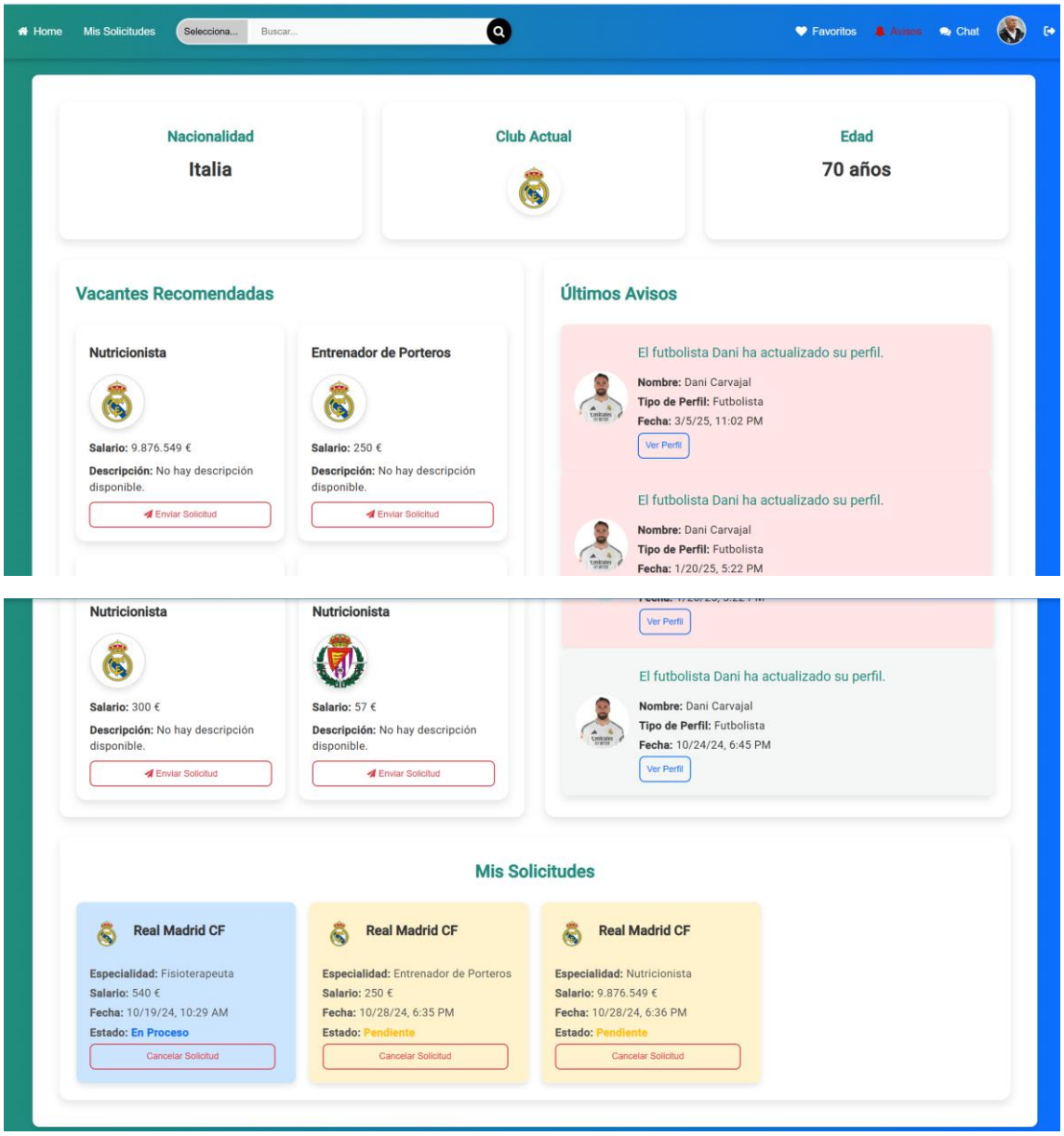


Ilustración 52: Manual dashboard entrenador

La información mostrada es la nacionalidad, club en el que juega actualmente y la edad, un listado con las vacantes recomendadas según las posiciones en las que juega y a través de las cuales se puede solicitar directamente, los tres últimos avisos más recientes (con un fondo rojo para cunado no lo has visto y que se quita en el momento en el que pinchas) y finalmente las solicitudes diferenciadas con distintos colores en función del estado en el que se encuentran.

8.3.2 Mi perfil

Haciendo click en la imagen del perfil se accede a la vista de “mi perfil” donde se podrá ver la información guardada del usuario, modificar los datos del usuario con el mediante el mismo formulario que el de registro o eliminar la cuenta del sistema.

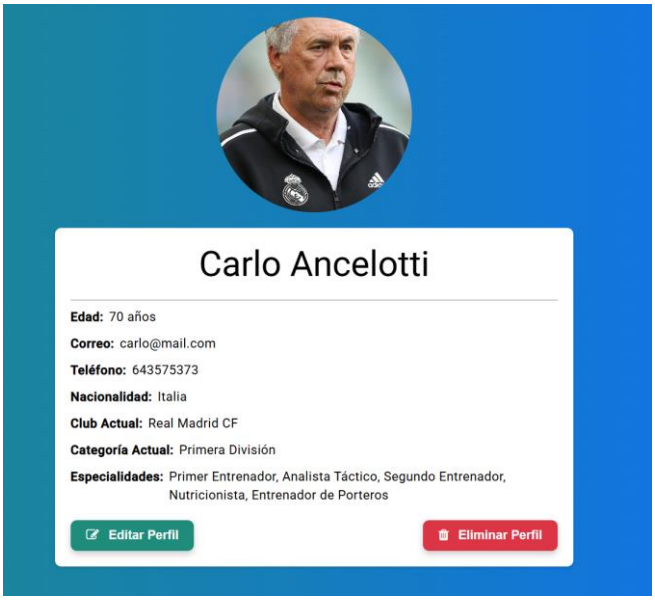


Ilustración 53: Manual perfil entrenador

Haciendo click en el icono de la navbar de al al lado del perfil se puede cerrar la sesión.

8.4 Usuario club ya autenticado

Cuando el usuario ya ha sido autenticado como club lo primero que podrá ver es un dashboard con la información general del usuario.

8.4.1 Home

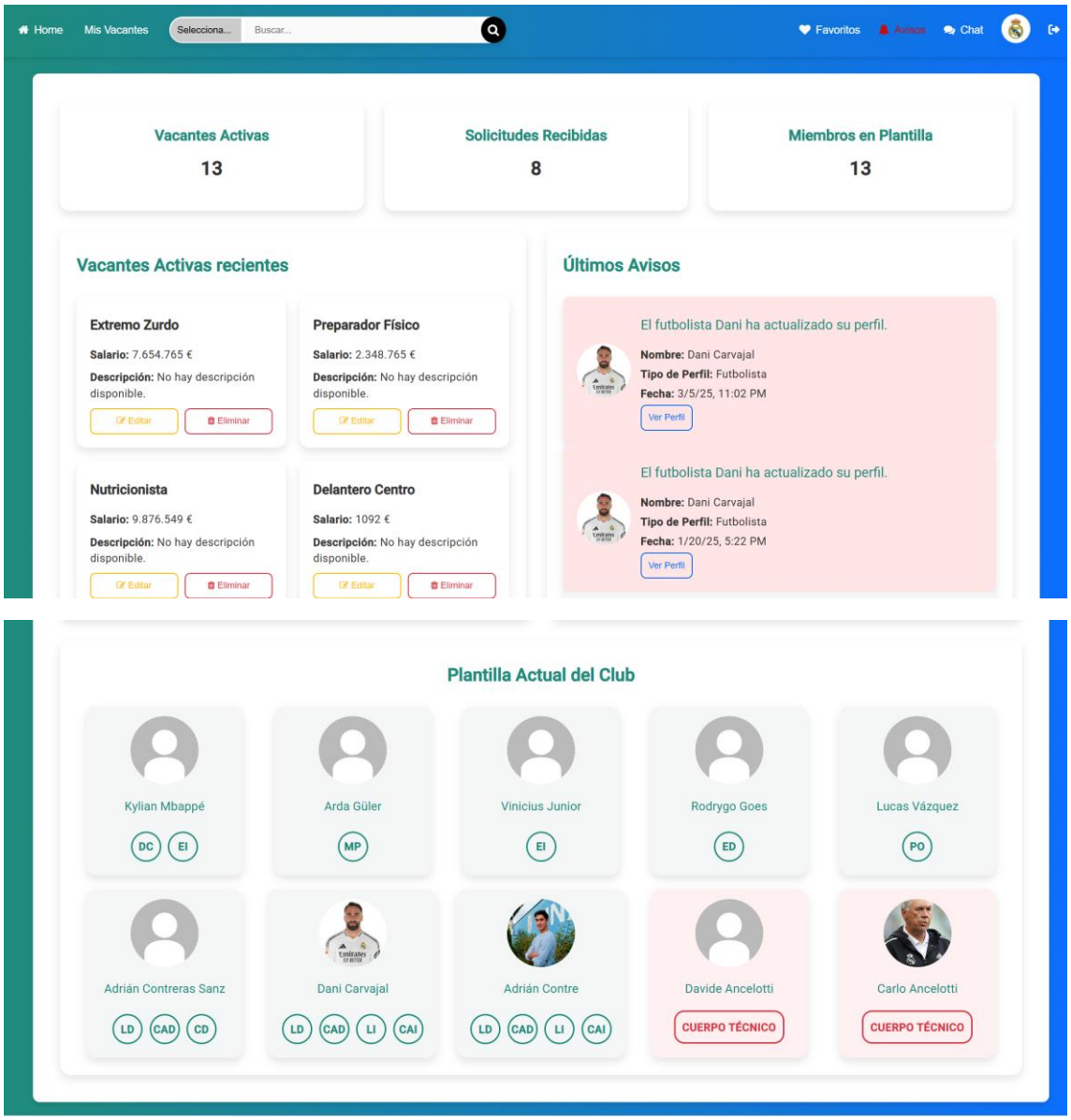


Ilustración 54: Manual dashboard club

En la parte de arriba se puede ver el número de vacantes activas, solicitudes recibidas y el número de miembros de la plantilla que compone el club. Después están las últimas seis vacantes activas desde donde podemos editar o eliminar de forma rápida mediante los botones correspondientes. También se ven las últimas tres notificaciones sobre los avisos que los clubes reciben. Finalmente, en la parte de abajo podemos acceder a la información de los perfiles que están en mi club.

8.4.2 Mi perfil

Haciendo click en la imagen del perfil se accede a la vista de “mi perfil” donde se podrá ver la información guardada del usuario, modificar los datos del usuario con el mediante el mismo formulario que el de registro o eliminar la cuenta del sistema.

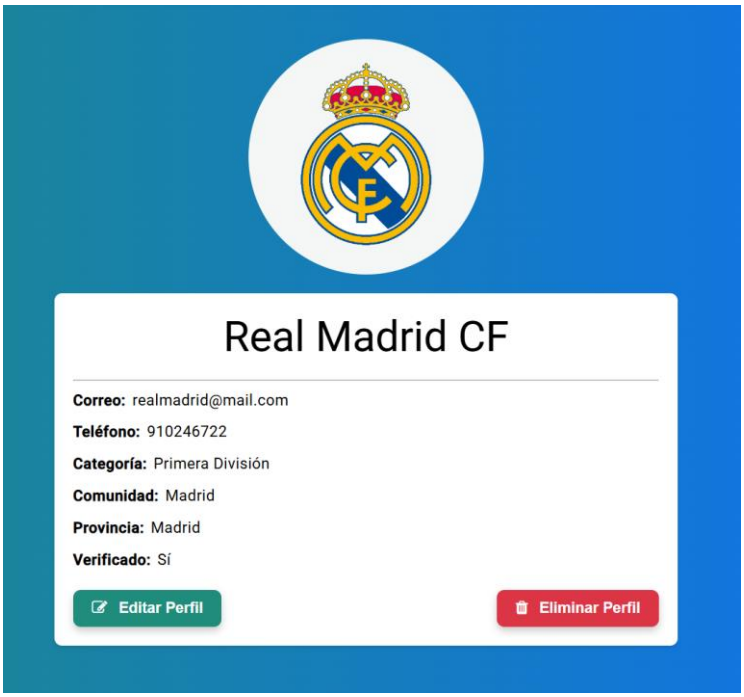


Ilustración 55: Manual perfil club

Haciendo click en el icono de la navbar de al al lado del perfil se puede cerrar la sesión

8.5 Buscadores

En la parte superior de la navbar hay un buscador con diferentes secciones

Se pueden encontrar hasta cuatro buscadores diferentes en la navbar de la aplicación, pero uno de ellos, vacantes solo está disponible para los perfiles que han sido autenticados como futbolistas o entrenadores.

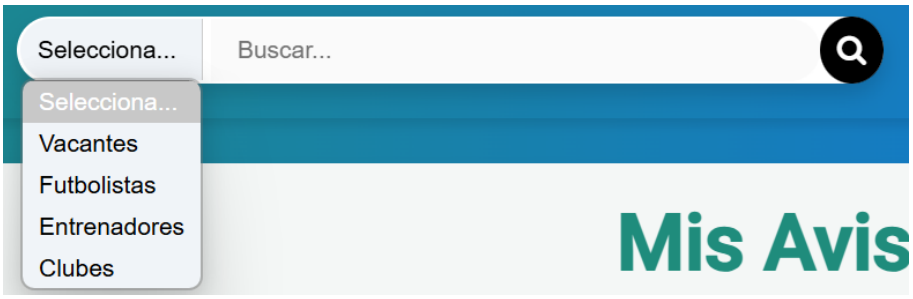


Ilustración 56: Manual buscadores

Hay cuatro buscadores con sus respectivos filtros correspondientes a las diferentes secciones (futbolistas, entrenadores, clubes y vacantes):

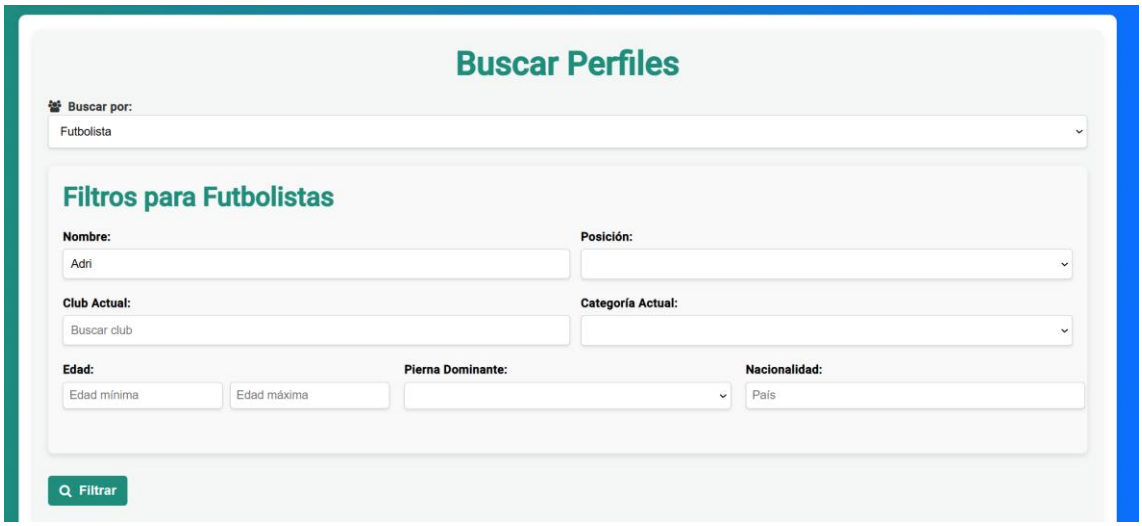


Ilustración 57: Manual buscador perfiles futbolista

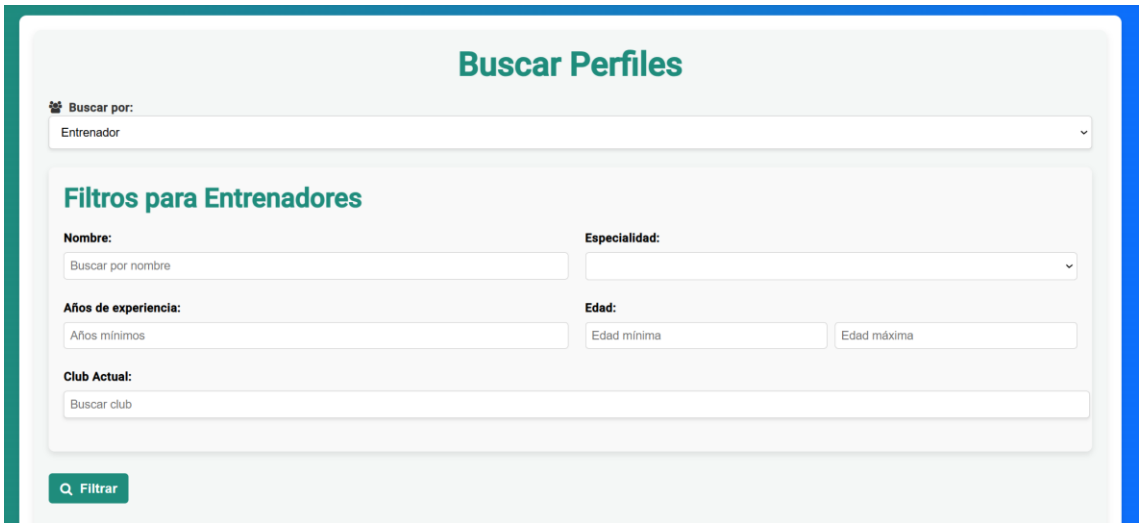


Ilustración 58: Manual buscador perfiles entrenador

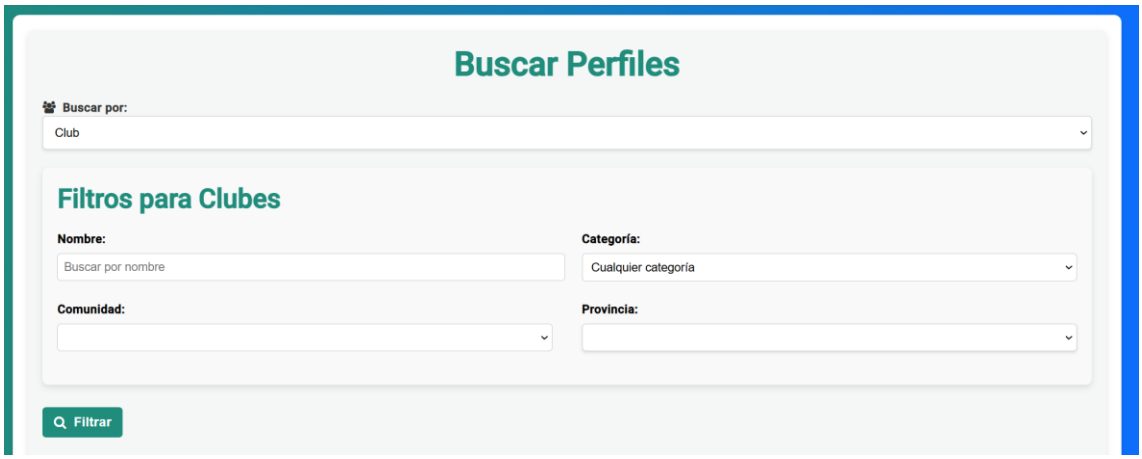


Ilustración 59: Manual buscador perfiles club

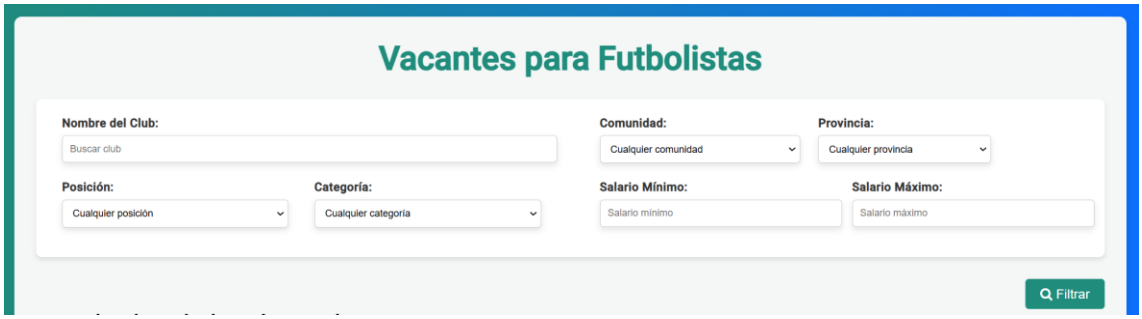


Ilustración 60: Manual buscador perfiles vacantes

8.6 Favoritos

En la sección de favoritos podemos ver en función de la sección elegida los perfiles que hemos guardado y queremos hacer un seguimiento.

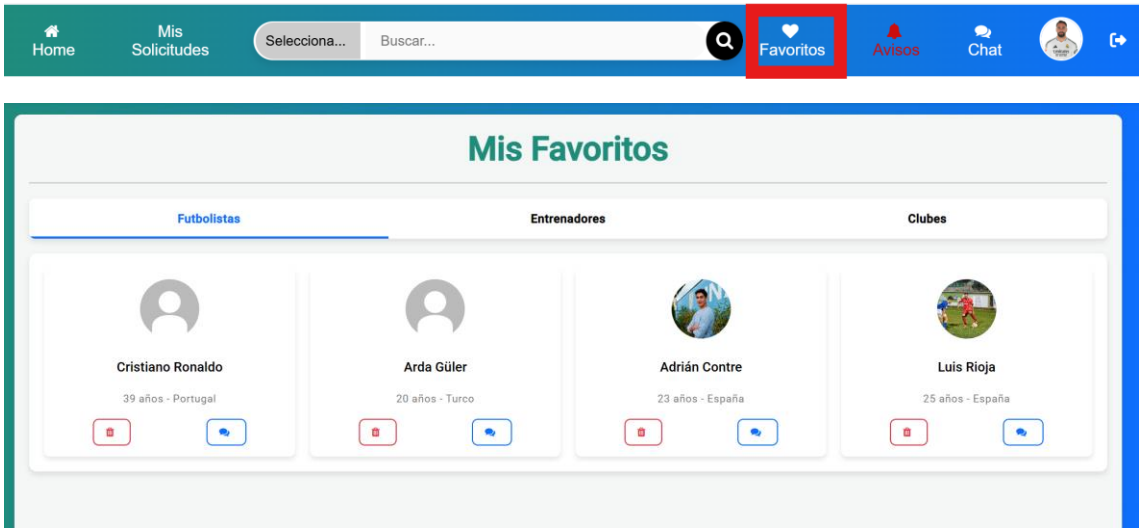


Ilustración 61: Manual favoritos

8.7 Chat

En la página del chat podemos mantener conversaciones en tiempo real con los usuarios de la aplicación.

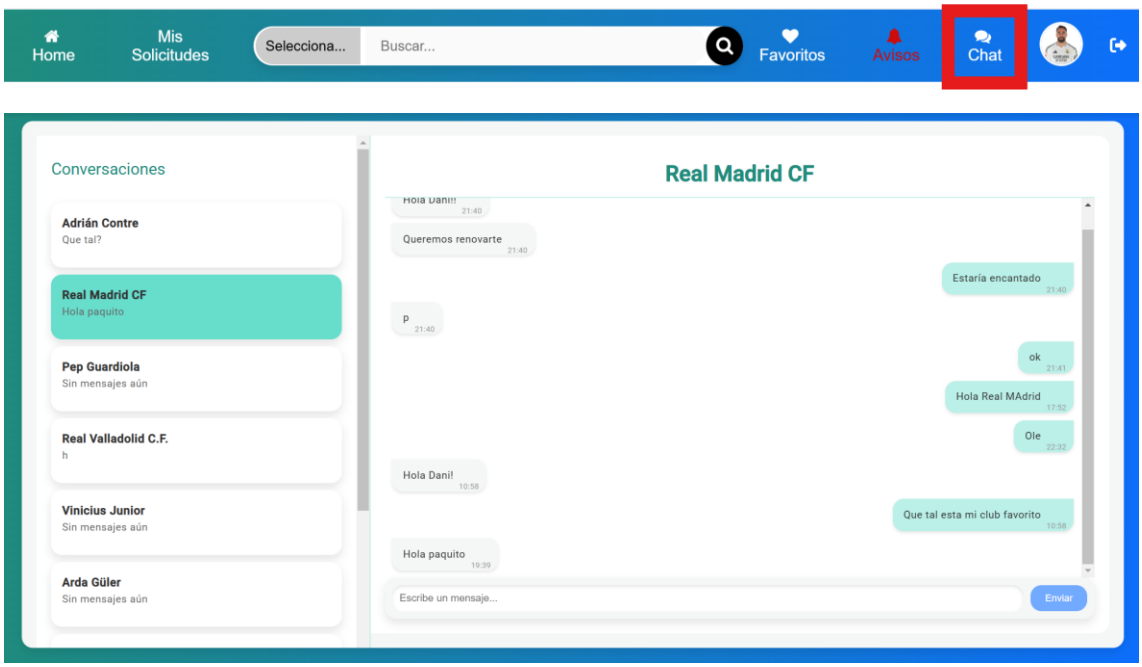


Ilustración 62: Manual chat

8.8 Avisos

En la página de avisos podemos ver las novedades de los perfiles a los que seguimos o tenemos guardados como favoritos y en el caso de los clubes se notifica quienes han solicitado un puesto a una vacante determinada.

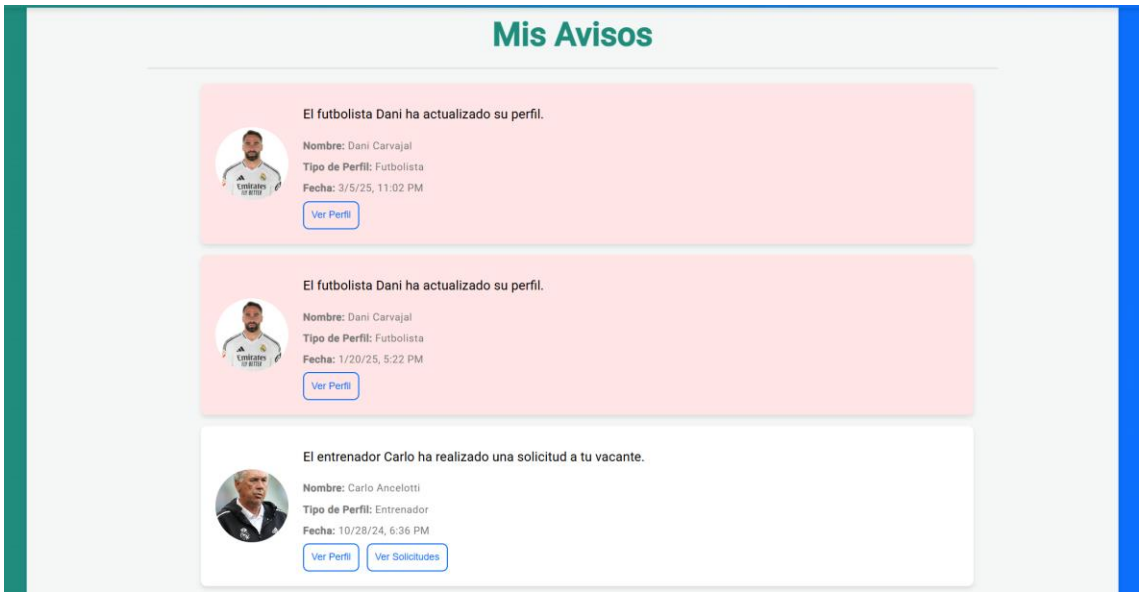


Ilustración 63: Manual Avisos

8.9 Usuario Administrador

Para los usuarios que han sido elegidos administradores su única función es la de verificar a los clubes que se dan de alta en el sistema o si por ejemplo se desea quitarle esos permisos para que no aparezcan al público se le puede invalidar la verificación.

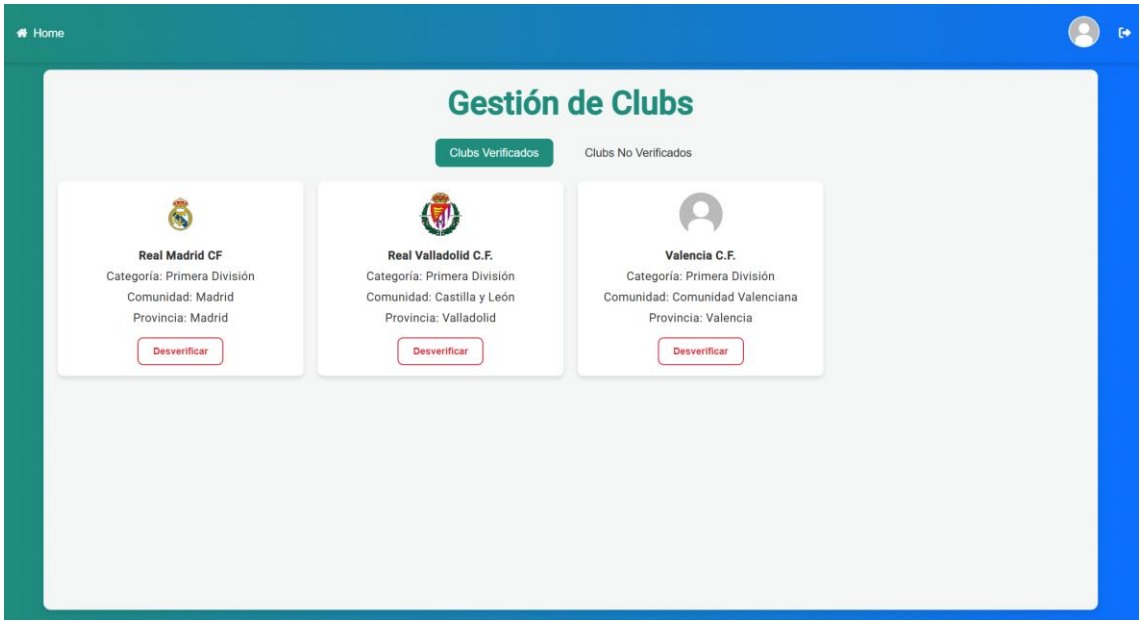


Ilustración 64: Manual administrador

9. Conclusiones y trabajo futuro

9.1 Conclusiones

Desarrollar LinkByBall ha sido tanto un desafío como una experiencia enriquecedora. Logré transformar una idea inicial en una plataforma funcional que conecta futbolistas, entrenadores y clubes, centralizando oportunidades y comunicación en un solo espacio.

El camino estuvo lleno de retos técnicos: desde implementar un *backend* robusto con *API RESTful* hasta diseñar un *frontend* en Angular intuitivo y visualmente atractivo. El objetivo siempre fue claro: crear una herramienta que genuinamente ayude a quienes buscan crecer en el mundo del fútbol y evolucionar como desarrollador de software en cada una de las partes del proyecto.

Mi experiencia laboral actual como desarrollador *fullstack* me permitió aplicar conocimientos previos, pero también me llevó a mejorar significativamente en varios aspectos:

- Estructuración más clara del código
- Profundización en las capacidades de Angular y Node.js
- Desarrollo paralelo de *frontend* y *backend* para optimizar tiempos

La plataforma ya cumple su propósito principal, pero veo claramente el potencial para expandirla con nuevas funcionalidades, mejor usabilidad y rendimiento optimizado. También he comprendido mejor la importancia crítica de la experiencia de usuario en cualquier desarrollo.

Más allá de ser un requisito académico, LinkByBall ha sido un proyecto muy especial para mí porque he podido desarrollar una aplicación sobre mis gustos particulares enfocándolo a una herramienta que podría ser de mucha utilidad para aquellos que han podido encontrarse en situaciones similares.

9.2 Líneas de mejora y de trabajo futuras

A pesar de todo el trabajo realizado e intentando hacerlo lo mejor posible desde el principio, siempre hay margen para seguir mejorando y ampliando la plataforma. Algunas líneas de mejora y trabajo futuro que podrían llevar a LinkByBall a otro nivel incluyen:

- **Optimización del rendimiento**: Mejorar tiempos de carga y eficiencia en las consultas a la base de datos para que la plataforma sea más rápida y fluida.
- **Mejoras en la experiencia de usuario (UX/UI)**: Refinar la interfaz para hacerla más intuitiva y atractiva, asegurando que cualquier usuario pueda navegar sin problemas.

- **Implementación de un sistema de recomendaciones**: Actualmente ya hay un sistema de recomendaciones, pero se podría utilizar algoritmos para sugerir automáticamente vacantes o contactos según el perfil y las interacciones de cada usuario.
- **Ampliación del sistema de notificaciones**: Incluir notificaciones en tiempo real para eventos importantes, como cuando un club visualiza un perfil o cuando se recibe un mensaje en el chat. Implementar un sistema de avisos a través de correo electrónico mediante Power apps, a través del cual se pueden hacer llamadas y automatizar el envío del correo cuando hay cierto evento.
- **Aplicación móvil**: Llevar LinkByBall a iOS y Android para que los usuarios puedan gestionar su perfil y buscar oportunidades desde cualquier lugar de manera más cómoda.
- **Aplicación *responsive***: Hacer la aplicación visible para cualquier tamaño, no solo en tamaño XL de escritorio sino también para tablet o móvil.
- **Integración de inicio de sesión mediante proveedores SSO**: Permitir el inicio de sesión a través de Google, Microsoft o Apple y luego rellenar los formularios en específico dependiendo del tipo de usuario.
- **Sistema de verificación de perfiles**: Implementar un método que garantice la autenticidad de los usuarios y evitar perfiles falsos, no solo verificar clubes sino todo tipo de usuarios.
- **Monetización de la plataforma**: Explorar modelos de negocio como suscripciones premium o anuncios dirigidos a clubes y agentes para hacer sostenible el proyecto a largo plazo.
- **Traducciones a otros idiomas**: Se podría mejorar la experiencia de usuario si hubiese alternativas para cambiar de idioma. Esto se podría hacer por ejemplo utilizando i18n como servicio de traducción rápida y casi inmediata.

Estas mejoras no solo harían que la plataforma fuese más completa, sino que también aumentarían su impacto dentro del mundo del fútbol, facilitando aún más la conexión entre jugadores, entrenadores y clubes.

10. Referencias

[1] LinkedIn Corporation, «About LinkedIn».

[2] D. Brown y A. Green, «How LinkedIn Transformed Professional Networking», *Journal of Digital Employment Strategies* , pp. 45-60, 2022.

[3] FutbolJobs, «Quiénes somos», https://futboljobs.com/quienes-somos/.

[4] R. López, «El mercado laboral en el fútbol: La digitalización del reclutamiento», *Ediciones Deportivas*, Madrid, 2022.

[5] Jobs In Football, «About us», https://jobsinfootball.com/about/.

[6] R. García, «Nuevas plataformas digitales para el empleo en el fútbol», Barcelona, España, 2022.

[7] Microsoft Corporation, «Microsoft Word: Write, edit & share docs on the go», https://www.microsoft.com/es-es/microsoft-365/word.

[8] Elsevier, «Mendeley: Reference management software», https://www.mendeley.com/.

[9] I. GitHub, «GitHub: The platform for collaborative software development», https://github.com/about.

[10] Microsoft Corporation, «Visual Studio Code: Code editing. Redefined», https://code.visualstudio.com/.

[11] Node.js Foundation, «Node.js: JavaScript runtime built on Chrome’s V8 engine», https://nodejs.org/en/about/.

[12] Express.js, «Express: Fast, unopinionated, minimalist web framework for Node.js», https://expressjs.com/.

[13] Google Inc., «Angular: Superheroic JavaScript MVW Framework», https://angular.io/.

[14] Microsoft Corporation, «TypeScript: JavaScript with syntax for types», https://www.typescriptlang.org/.

[15] MongoDB Inc, «MongoDB: The database for modern applications», https://www.mongodb.com/.

[16] «Desarrollo iterativo e incremental», https://proyectosagiles.org/desarrollo-iterativo-incremental/.