



ugr | Universidad
de **Granada**

TRABAJO FIN DE GRADO
INGENIERÍA INFORMÁTICA

eSports Fantasy

Juego de gestión de equipos de deportes electrónicos

Autor

Alberto Plaza Montes

Director

Juan Manuel Fernández Luna



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS INFORMÁTICA Y DE
TELECOMUNICACIÓN

Granada, 10 de 2024



ESPORTSFANTASY

eSports Fantasy

Autor

Alberto Plaza Montes

Director

Juan Manuel Fernández Luna

eSports Fantasy: Juego de gestión de equipos de deportes electrónicos.

Alberto Plaza Montes

Palabras clave: fantasy, juego, jugador, compra, venta, puja, equipo, puntuación, deporte electrónico.

Resumen

Los juegos estilo fantasy surgen con el objetivo de convertir a los consumidores en algo más que meramente espectadores de las competiciones que siguen. Buscan hacerlos partícipes de ella. Estos videojuegos permiten a sus participantes crear y gestionar equipos virtuales compuestos por los jugadores reales en deportes como el fútbol, baloncesto, rugby y otros.

Sin embargo, esta fórmula de entretenimiento no ha sido lo suficientemente explorada en el emergente mundo de los deportes electrónicos. Enormes competiciones de algunos de los videojuegos más seguidos de la historia como League of Legends, Counter Strike o Valorant no cuentan a penas con plataformas que brinden la experiencia fantasy que los usuarios merecen.

Me enfoco en desarrollar una plataforma en la que los usuarios puedan disfrutar de estas competiciones, jugar en ligas públicas o privadas con sus amigos, compitiendo entre sí para demostrar sus habilidades de análisis en rendimiento de jugadores o toma de decisiones estratégicas.

En este Trabajo de fin de grado se realizará una travesía completa a lo largo del ciclo de vida de un producto software, tratando temas como el análisis de mercado o tecnologías, diseño de interfaces, gestión de recursos hardware, recolección y normalización de datos en un sistema de información, diseño de base de datos y arquitecturas backend y frontend entre otros temas.

eSports Fantasy: Electronic sports team management game.

Alberto Plaza Montes

Keywords: fantasy, game, player, buy, sell, bid, team, score, electronic sport.

Abstract

Fantasy style games emerge with the aim of turning consumers into more than mere spectators of the competitions they follow. Instead, they aim to involve them actively in the experience. These video games allow participants to create and manage virtual teams composed of real players in sports such as football, basketball, rugby, and others.

However, this form of entertainment has not been adequately explored in the emerging world of esports. Major competitions of some of the most followed video games in history such as League of Legends, Counter-Strike, or Valorant lack platforms that provide the fantasy experience that users deserve.

I focus on developing a platform where users can enjoy these competitions, play in public or private leagues with their friends, and compete against each other to demonstrate their skills in player performance analysis or strategic decision making.

In this Bachelor's Thesis, a comprehensive journey will be undertaken throughout the lifecycle of a software product, addressing topics such as market or technology analysis, interface design, hardware resource management, data collection and normalization in an information system, database design, and backend and frontend architectures, among other subjects.

Yo, **Alberto Plaza Montes**, alumno de la titulación Ingeniería Informática de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 26828538G, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.



Fdo: Alberto Plaza Montes

Granada a 1 de septiembre de 2024 .

D. Juan Manuel Fernández Luna, Profesor del área de Ciencias de la Computación e Inteligencia Artificial del Departamento Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada.

Informa:

Que el presente trabajo, titulado *eSports Fantasy: Juego de gestión de equipos de deportes electrónicos.*, ha sido realizado bajo su supervisión por **Alberto Plaza Montes**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a 1 de septiembre de 2024 .

El director:

Juan Manuel Fernández Luna

Agradecimientos

Agradecer a mis amigos, los que venían conmigo y los que he hecho por el camino, por todo el apoyo que me han brindado de forma completamente desinteresada. He aprendido mucho de todos vosotros, Tener un buen amigo te hace, sin lugar a dudas, una mejor persona.

Agradecer a mis padres, por haberme inculcado siempre valores de interés, trabajo y dedicación. Por todo el apoyo incondicional que me han otorgado y todos los esfuerzos que han tenido que hacer, de los que me he enterado y de los que no, para conseguir que yo esté aquí.

Agradecer a mi tutor y a todos los buenos profesores que he tenido a lo largo de estos cuatro años en la ETSIIT. Gracias por compartir conmigo tanto conocimiento, experiencia y tanta calidad humana.

A todos de corazón, muchas gracias.

Índice general

1. Introducción	17
1.1. Motivación	17
1.2. Descripción del problema	18
1.3. Objetivos	18
1.3.1. Objetivos en la aplicación	18
1.3.2. Objetivos personales	19
1.4. Estudio de aplicaciones similares en el mercado	19
1.5. Descripción de la solución	22
1.6. Contenido de la memoria	26
2. Historias de usuario, planificación y presupuesto	27
2.1. Estudio de las metodologías existentes y elección	27
2.2. División del trabajo: Épicas	31
2.3. Descripción de perfiles de usuario	31
2.4. Identificación de roles	31
2.5. Historias de usuario	32
2.5.1. Épica 1: Gestión de Usuario	32
2.5.2. Épica 2: Gestión de ligas	34
2.5.3. Épica 3: Gestión de equipo	35
2.5.4. Épica 4: Gestión de datos generales	36
2.5.5. Épica 5: Administración de ligas públicas	37
2.6. Historias técnicas	37
2.7. Gestión de sprints y progresión	39
2.7.1. Sprint 1	39
2.7.2. Sprint 2	40
2.7.3. Sprint 3	41
2.7.4. Sprint 4	41
2.7.5. Sprint 5	42
2.7.6. Sprint 6	43
2.7.7. Progresión de sprints	43
2.8. Presupuesto	46

3. Implementación	51
3.1. Estudio de tecnologías existentes y elección	51
3.2. Gestión de tests unitarios y de integración	59
3.3. Sprint 1	60
3.3.1. Introducción al Sprint 1	60
3.3.2. HT 1: Configuración de despliegue en HW.	61
3.3.3. HT 2: Diseño e implementación de la base de datos. .	64
3.3.4. HT 3: Diseño de arquitectura y configuración del entorno Backend.	72
3.3.5. HT 4: Diseño de arquitectura y configuración del entorno Frontend.	74
3.3.6. Retrospectiva y retrasos	77
3.4. Sprint 2	77
3.4.1. Introducción al Sprint 2	77
3.4.2. Autenticación JWT	79
3.4.3. HU 1.1: Registro en la aplicación (correo)	79
3.4.4. HU 1.2.1 y 1.2.2: Inicio de sesión (Correo y Google) .	82
3.4.5. HU 1.3: Configuración de cuenta	86
3.4.6. Tests unitarios y de integración	87
3.4.7. Retrospectiva y retrasos	94
3.5. Sprint 3	94
3.5.1. Introducción al Sprint 3	94
3.5.2. HU 2.1 y 2.3: Creación y abandono de ligas	96
3.5.3. HU 2.2: Invitar a ligas	101
3.5.4. HU 2.4: Acceso a ligas asociadas	105
3.5.5. Tests unitarios y de integración	107
3.5.6. Retrospectiva y retrasos	111
3.6. Sprint 4	111
3.6.1. Introducción al Sprint 4	111
3.6.2. HT 5: Implementación del sistema recolector de datos	113
3.6.3. HT 8, HU 3.1 y 3.2: sistema rotativo de mercado y compraventa de jugadores	118
3.6.4. HU 3.4, 3.5: Clausulazos y aumentos de cláusula . .	122
3.6.5. HU 3.6: Alineamiento de jugadores en equipos . . .	124
3.6.6. HU 3.7: Consulta de historial	126
3.6.7. Tests unitarios y de integración	127
3.6.8. Retrospectiva y retrasos	131
3.7. Sprint 5	132
3.7.1. Introducción al Sprint 5	132
3.7.2. HT 6: sistema de ponderación de puntuaciones de desempeño	132
3.7.3. HT 7: Heurística de valores de mercado	135
3.7.4. Tests unitarios y de integración	135
3.7.5. Retrospectiva y retrasos	136

3.8. Sprint 6	137
3.8.1. Introducción al sprint 6	137
3.8.2. HU 4.1 y 4.2 : Mapeo y muestreo de datos en el frontend.	137
3.8.3. Tests unitarios y de integración	143
3.8.4. Retrospectiva y retrasos	143
4. Conclusiones y trabajos futuros	145
4.1. Emprendimiento y visión comercial	145
4.2. Conclusiones	146
4.3. Posibles trabajos futuros	146
4.4. Reflexión personal	146

Capítulo 1

Introducción

1.1. Motivación

El mercado de los deportes electrónicos ha crecido de forma desorbitada en los últimos 5 años, abriéndose paso entre otro tipo de competencias más tradicionales y acaparando la atención del público más joven. Con ello, crece alrededor de los deportes electrónicos todo un subsistema social y económico que no debe pasar desapercibido, desde prensa especializada hasta industria textil. Dentro de ese subsistema, destacaremos para este proyecto de fin de grado el ocio.

Los juegos de tipo fantasy son, en definitiva, juegos de gestión de gestión de recursos. Consisten en crear un equipo virtual conformado por jugadores de diferentes equipos de una determinada competición, pujando contra los demás usuarios por esos jugadores que mejor rendimiento tienen y que, por consiguiente, mejores puntuaciones otorgarán al usuario.

Estos juegos ponen a prueba tu conocimiento en la competición, capacidades de análisis en rendimiento de jugadores, estrategias de compraventa para maximizar beneficios y en definitiva, tu capacidad analítica para obtener una ventaja con respecto a tus oponentes.

Ya existen más que de sobra plataformas fantasy para deportes como el fútbol o el baloncesto, sin embargo, me ha sido prácticamente imposible encontrar ningún juego de este estilo que esté enfocado en los deportes electrónicos¹. Por consiguiente, he hecho de esta carencia mi proyecto de fin de grado, en el que me dispongo a crear yo mismo este videojuego.

¹Competiciones de videojuegos a nivel profesional.

1.2. Descripción del problema

Tal y como veremos en el estudio de aplicaciones similares realizado en este mismo capítulo, podremos observar cómo claramente el mercado de videojuegos estilo fantasy enfocándose en deportes electrónicos no ha sido lo suficientemente explotado en el sector. En el mejor de los casos hay algún juego fantasy especializado en una competición o videojuego en concreto pero que aún así por diferentes motivos no ha funcionado ni calando entre el público, ya sean por interfaces antiguas, malas decisiones creativas u otros motivos. La falta de variedad en estos juegos es además causa directa de la dificultad que crear una aplicación de estas características supone.

Esta dificultad es consecuencia de la falta de ecosistema en el medio. Normalmente, cuando se implementa una aplicación fantasy, es fácil abstraerse de cuestiones como la obtención de datos o la puntuación del desempeño de los jugadores gracias a las APIs o a las bases de datos de dominio público, permitiendo así a la aplicación fantasy enfocarse solamente en cuestiones propias como el manejo de ligas o de mercado. Sin embargo, para los deportes electrónicos no abundan fuentes de datos de las que obtener información de los jugadores y menos aún que nos permitan obtener ponderaciones de su desempeño en los partidos.

En este Trabajo de Fin de Grado se pretende dar fin a este problema gestionando nosotros mismos estas capas de las que normalmente los juegos fantasy pueden abstraerse. Obteniendo y normalizando datos, realizando las ponderaciones de desempeño mediante heurísticas de puntuación propias y además gestionar lo propio a un juego fantasy como las ligas, los usuarios, los mercados, etc. Aprovechando así además esta carencia de mercado para obtener un producto potencialmente innovador.

1.3. Objetivos

Antes de comenzar con el planteamiento general, dejaré claros cuales son los objetivos básicos de este TFG y de esta aplicación.

1.3.1. Objetivos en la aplicación

- **Crear una aplicación de escritorio:** Creación de la aplicación web del juego.
- **Desarrollar un sistema escalable a múltiples juegos:** Hacer la aplicación fácilmente escalable a nuevos videojuegos, haciendo que se tenga que programar lo mínimo posible, principalmente en el apartado de normalización de datos, para incluir nuevos deportes electrónicos.

- **Ejemplificar un sistema de puntuación:** Se debe implementar al menos una heurística de puntuación en función de desempeño en el partido para el videojuego que se vaya a utilizar como ejemplo. En este caso, League of Legends.
- **Automatizar sistema de mercado:** Autogestión por parte de la aplicación de la compraventa de jugadores, sistema de pago de cláusulas, pujas y rotación de jugadores libres.

1.3.2. Objetivos personales

- **Mejorar mi conocimiento en metodologías ágiles:** Familiarizarme con la que es la principal metodología en el mercado laboral actual, el desarrollo ágil.
- **Practicar nuevas tecnologías:** Mejorar mi capacidad de desarrollo en tecnologías novedosas y con prácticas modernas.

1.4. Estudio de aplicaciones similares en el mercado

Antes de comenzar tanto con el diseño como con la planificación de la aplicación, es importante realizar una investigación a las aplicaciones en el mercado que ofrecen competencia a la nuestra, con el objetivo de enfocarnos en utilizar buenos patrones, reforzando los puntos positivos de la aplicación y evitando los puntos negativos.

Comunio [02]

- **Información:** Se trata del primer juego “fantasy” en línea disponible en España, creado en 2002.
- **Descripción del estado:** Al ser el más antiguo es el que cuenta con una mayor comunidad de jugadores, sin embargo esto también hace que sus interfaces se vean desactualizadas, lo que lo convierte en un juego de nicho y poco amigable a nuevos jugadores.
- **Funcionalidad básica:**
 1. Compraventa de jugadores.
 2. Formación de equipo propio.
 3. Sistema de puntuación de jugadores propio.
 4. Sistema de cláusulas.
- **Ventajas:**

1. Comunidad amplia y activa.
2. Personalización de ligas muy flexible.
3. Sistema de puntuación transparente.
4. Amplia cobertura de ligas.

■ **Desventajas:**

1. Interfaz anticuada, menos atractiva e intuitiva que las de hoy en día.
2. Contiene publicidad.

Marca Fantasy [16]

■ **Información:** Se trata de la versión de juego lanzada por el diario “Marca”.

■ **Descripción del estado:** Fue el primero en aportar una interfaz amigable para móviles, y por tanto se quedó con una gran cantidad de jugadores como consecuencia del “boom” de los smartphones. Es el que usa la gran mayoría de gente en España. Además está asociado a múltiples sistemas de puntuación de jugadores reconocidos por los aficionados.

■ **Funcionalidad básica:**

1. Compraventa de jugadores.
2. Formación de equipo propio.
3. Sistema de puntuación de diferentes fuentes.

■ **Ventajas:**

1. Diseño atractivo.
2. Información actualizada en tiempo real.

■ **Desventajas:**

1. Menor flexibilidad en personalización de ligas.
2. Menor transparencia en la puntuación.

Mister Mundo Deportivo [19]

■ **Información:** Se trata de la versión de juego lanzada por el diario “Mundo deportivo”.

- **Descripción del estado:** No es tan conocido en España, pero es más reconocido a nivel internacional. Cuenta con una interfaz limpia y bastante simple, por lo que es difícil que no guste.
- **Funcionalidad básica:**
 1. Compraventa de jugadores.
 2. Formación de equipo propio.
 3. Sistema de puntuación de diferentes fuentes.
 4. Múltiples ligas posibles.
- **Ventajas:**
 1. Información actualizada en tiempo real.
 2. Diseño atractivo.
 3. Puntuación transparente.
- **Desventajas:**
 1. Menos conocida internacionalmente.
 2. Contiene publicidad.

Super Fantasy lol [20b]

- **Información:** Reinterpretación del juego creado por los responsables directos de los eSports de League of Legends.
- **Descripción del estado:** Como consecuencia de la reinterpretación del sistema “fantasy” clásico, el juego ha adaptado una jugabilidad extraña mediante un sistema de cartas, el cual no termina de convencer a los usuarios de “fantasy” habituales, pues lo que más suele atraer a estos de este tipo de juegos es el reto de gestión de recursos mediante compraventa.
- **Funcionalidad básica:**
 1. Compraventa de jugadores.
 2. Minijuegos que aportan dinamismo.
- **Ventajas:**
 1. Temática de eSports.
 2. Buen diseño visual.
- **Desventajas:**
 1. Funcionamiento mediante un sistema de cartas a modo de innovación que no terminó de cuajar en el público.

E-go fantasy [20a]

- **Información:** E-go es el principal y único juego fantasy de deportes electrónicos que he podido encontrar en la PlayStore.

- **Descripción del estado:** No parece tener buenas críticas en la app-store, pues cuenta con una puntuación de 2/5, parece ser consecuencia de múltiples bugs y diseños de interfaz no amigables con el usuario.

- **Funcionalidad básica:**

1. Compraventa de jugadores.
2. Formación de equipo propio.

- **Ventajas:**

1. Temática de eSports.
2. Buen diseño visual.
3. Variedad de eSports.

- **Desventajas:**

1. Problemas con la gestión de usuarios (quejas en los comentarios al respecto).
2. A pesar de que visualmente sea atractiva, la interfaz es confusa.
3. No tiene un sistema de mercado desarrollado.

Tras este estudio a la competencia, podemos obtener varias conclusiones las cuales vamos a intentar aplicar a lo largo de la planificación y el desarrollo del proyecto.

- Las interfaces atractivas aportan mucho valor para el jugador.
- Es fundamental crear las interfaces de forma intuitiva para el usuario.
- Las reformulaciones del juego parecen no funcionar, si bien se pueden incluir pequeñas innovaciones, estas solo deben ser extras que no alejen el juego del modelo más clásico de los fantasy.

1.5. Descripción de la solución

En este apartado de explicará a grandes rasgos cuál será el resultado directo de la solución implementada, explicando los puntos más clave del sistema.

Se trata de un videojuego online de administración de recursos y estrategia, el cual pone a prueba la capacidad de gestión del jugador para ser director de un equipo de deportes electrónicos. Este contará con compra/venta de jugadores, puntuación de jugadores en función de su desempeño en la vida real, rankings de jugadores, y más.

Es importante aclarar a qué hacemos alusión con «usuario» y a qué con «jugador». Este patrón se repetirá a lo largo de todo el trabajo de fin de grado.

- **Usuario:** Es la persona que utiliza/juega a la aplicación. Hace pujas sobre jugadores, tiene su propio equipo, etc.
- **Jugador:** Es el jugador profesional de deportes electrónicos. Salen a mercado de fichajes y los usuarios pueden pujar por ellos y alinearlos en sus equipos.

Funcionamiento

Todos los usuarios tienen la posibilidad de crear/unirse a una o varias ligas (las cuales están asociadas intrínsecamente a una liga real). Al entrar en esta liga, el usuario comenzará con una cantidad de dinero inicial y un equipo inicial formado de manera aleatoria por profesionales de deportes electrónicos del mundo real y que pertenecen a esa liga en concreto.

Mercado

Cada liga tendrá su propio mercado de jugadores el cual contará con dos tipos de ofertas (ver figura 1.1):

- **Las que hace el propio juego:** son de carácter rotativo. Irán apareciendo jugadores que estén libres y no pertenezcan a ninguna plantilla aún. Estos partirán de un valor base, y permitirán hacer pujas anónimas (es decir, nadie sabe ni si otros jugadores han pujado, ni cuánto por los jugadores del mercado). Una vez finalice el tiempo de puja, el jugador será adjudicado al usuario que más dinero haya pujado. Los jugadores que no han recibido pujas rotarán por una nueva tanda.
- **Las que hacen los usuarios:** Los usuarios podrán poner sus jugadores en el mercado. Estos podrán obtener pujas de otros usuarios. En caso de que ningún usuario haya pujado, cuando se acabe el tiempo de límite, la máquina pujará por el jugador su precio de mercado. El usuario puede rechazar o aceptar las ofertas que le hagan.

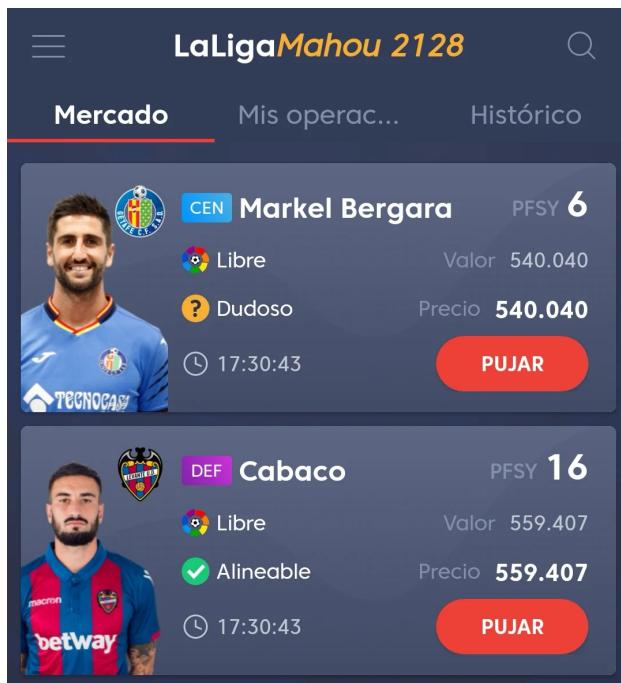


Figura 1.1: Mercado de jugadores en una liga del juego «La liga Fantasy» del diario Marca.

Ciclos de juego

Los jugadores están altamente ligados a su desempeño en la vida real. La mayoría de las ligas de videojuegos, independientemente del juego suele seguir un patrón general, muy parecido al de la liga de fútbol, por jornadas. Cada periodo de tiempo (1 semana, 10 días, 15 días, etc) se comienza una de estas jornadas, en la que a lo largo de algunos días, unos equipos se van enfrentando con otros. Si ganan puntuán más, si empatan puntuán menos, si pierden menos aún o en caso de que el desempeño haya sido demasiado malo, puntuarán en negativo. Se van repitiendo estos enfrentamientos hasta que todos los equipos hayan jugado con todos y finalmente, el que más puntos haya conseguido será el ganador de la liga.

De este modo, el juego solo se podrá jugar para determinadas ligas siempre y cuando estas estén operativas. En temporada activa, el juego tiene dos ciclos de juego claramente marcados “entre-jornada” y “jornada activa”. Entre jornadas es cuando se suelen hacer las gestiones en el equipo, compras, ventas, ofertas a usuarios, pagos de cláusula, etc. Mientras que con jornada activa lo que se suele hacer es ver como va la puntuación de tus jugadores durante esta jornada. Cuando esta finaliza, se realiza el reparto de puntos y de dinero asociado a esos puntos. Aún así, esto no necesariamente significa que no se puedan realizar compras, ventas y demás durante una

jornada, pero aplicarán a la siguiente jornada, aunque este aspecto aún está por determinar.

El equipo

Con los jugadores que comience, más los que vaya comprando en el mercado o de otros participantes de la liga, el usuario podrá conformar un equipo, el cual será el que puntúe a final de cada jornada.

En la figura 1.2 podemos apreciar los jugadores de distintos equipos conformando el equipo del usuario. Es jornada activa, pero acaba de comenzar, como aún no han terminado los partidos, no se sabe la puntuación de ninguno de los jugadores. Seguro vayan terminando, irán apareciendo sus puntuaciones.

El sistema de puntuación

Normalmente, este tipo de aplicaciones obtienen la puntuación de fuentes externas que califican el desempeño de los jugadores en cada partido, sin embargo, para este género me ha sido literalmente imposible encontrar un sistema similar, por lo que habrá que crear una heurística de puntuación de jugador en función de las estadísticas que obtengan en cada partido. Estas estadísticas son, en rasgos generales: asesinatos, muertes, asistencias, torretas destruidas, puntuación de visión, etc. Aunque pueden variar en función del deporte electrónico.

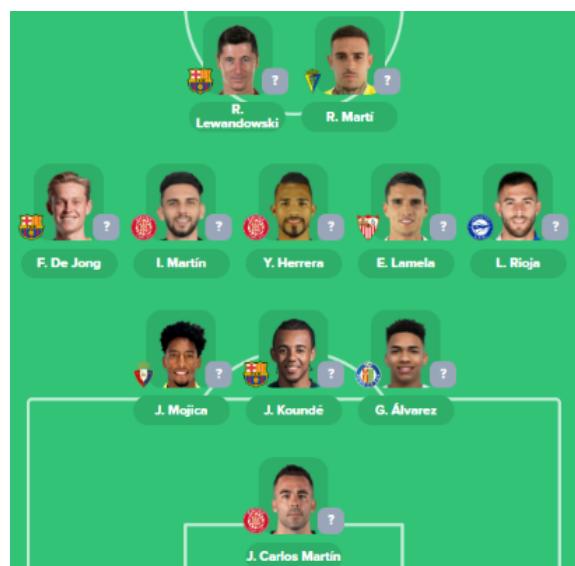


Figura 1.2: Equipo de jugadores de un usuario en el juego «Mister Fantasy» del diario mundo deportivo.

El objetivo es simple, gana el que tenga mayor puntuación total cuando finalice la liga a la que están asociados. En cada jornada, se recibe cierta cantidad de dinero en función del número de puntos obtenidos, lo que te permitirá mejorar tu equipo comprando nuevos jugadores o “robándoselos” a otros usuarios mediante el sistema de pagos de cláusula de rescisión de contrato del que hablaremos más adelante.

1.6. Contenido de la memoria

- **Capítulo 1:** Destinado a la presentación del Trabajo de Fin de Grado, describiendo el problema, los objetivos, realizando un análisis del estado de la competencia y describiendo el funcionamiento general de la aplicación.
- **Capítulo 2:** Destinado al análisis de requisitos mediante historias de usuarios, planificación temporal y estudio de presupuestos para el desarrollo.
- **Capítulo 3:** Destinado al análisis técnico, toma de decisiones y desarrollo de la aplicación.
- **Capítulo 4:** Destinado al desenlace del Trabajo de Fin de Grado, con conclusiones respecto al trabajo realizado, perspectiva futura y conclusiones finales.

Capítulo 2

Historias de usuario, planificación y presupuesto

En este capítulo vamos a realizar un estudio previo antes de comenzar con la programación o la maquetación de la aplicación. Profundizando en metodología, requisitos de usuario y presupuesto. Estos son algunos de los estudios que es fundamental llevar acabo, con el objetivo de no encontrar contratiempos mayores en el camino una vez se ha comenzado el desarrollo que puedan llevar a fallos catastróficos y de mejorar y optimizar el tiempo que invertimos en el proyecto.

2.1. Estudio de las metodologías existentes y elección

En este proyecto se va a realizar una aplicación web, la cual contará con backend, frontend y base de datos. Estas tecnologías están, en primera instancia, aisladas unas de otras, lo que dificulta su desarrollo y mantenimiento. Por ello es necesario utilizar una metodología que permita integrar estas tecnologías de forma eficaz.

En este apartado se realizará un estudio de las principales metodologías de desarrollo que se han tenido en cuenta para realizar este proyecto, con sus correspondientes ventajas y desventajas, y se dará una explicación a la elección de la decisión final.

Metodología en cascada [Jul20]

Se trata de una metodología del marco tradicional caracterizada por la planificación, concibiendo el proyecto como uno solo, con una estructura bien definida y que sigue un proceso unidireccional y secuencial.

En esta metodología, se definen una serie de etapas dependientes unas de otras. Cada una de estas etapas de desarrollo generan documentos o aplicaciones entregables al cliente, llegando al resultado final de forma incremental.

Ventajas

- **Fácil de administrar** gracias a sus entregables específicos y sencillos procesos de revisión.
- **Fácil de realizar la planificación** ya que los requisitos del proyecto se acuerdan en la primera fase.
- **Útil en proyectos pequeños.**
- **No hay retraso de ejecución de proyecto** ya que los clientes no pueden agregar nuevos requisitos constantemente.

Desventajas

- **No apto para modelos de gran tamaño.**
- **Poco efectivo si no** se da una clara especificación de requisitos desde el inicio.
- **Escasa flexibilidad** al ser una metodología rígida, no hay lugar a posibles sucesos inesperados.

Metodología ágil [Jul20]

Es la metodología de desarrollo más utilizada hoy en día en el mundo del software. Consiste en la entrega de producto software de manera temprana y constante, con el objetivo de recibir notificaciones de incidencias o gestiones del cambio mediante el contacto regular con el cliente y el equipo. Siempre manteniendo los principios de “agilidad, eficiencia y calidad”.

Ventajas

- **Cliente constantemente informado** y puede opinar sobre la evolución del proyecto.
- Ciclo de vida **simple** y de fácil comprensión.
- **Eliminación de trabajo innecesario.**
- **Retroalimentación** más rápida al usuario final.
- Permite un alto enfoque en la **agilidad** ya que el proceso está dividido en fases.

Desventajas

- **Alta necesidad de interacción** con el cliente, que podría no disponer del tiempo o interés suficiente.
- **Falta de límites** que puede llevar a constantes cambios y soluciones erróneas.
- **Falta de documentación.**

Metodología de cadena crítica [Jul20]

Esta metodología se basa en la teoría de las restricciones, que consiste en que “la productividad está limitada por las tareas de bajo rendimiento y se comprende como una línea de producción formada por estaciones de trabajo”. De este modo, el objetivo es priorizar estas tareas críticas y de bajo rendimiento para completarlas dentro del plazo establecido.

Se basa en tres principios:

1. **Identificar restricciones**, señalando el conjunto de tareas que definen la duración mínima del proyecto.
2. **Priorizar tareas**, focalizando la atención del equipo en la realización de las tareas identificadas anteriormente.
3. **Subordinar** el resto de **tareas** en la cadena crítica.

Ventajas

- Enfoque en tareas de mayor importancia.
- Reducción temporal y de recursos.
- Planificación de planes de acción de tareas con efecto en el producto final.

Desventajas

- Requiere de una **planificación más estructurada**.
- Requiere de **software específico** para la ejecución del proyecto.

Metodología iterativa [Jul20]

Chequeo y retroalimentación constante del producto aunque no esté finalizado. Dando lugar a la dinamización de los requisitos (y abriéndolos a posibilidad de modificación). Con posibles entregas parciales o simplemente una final.

Es, en esencia, una metodología de interacciones desglosada en secuencias específicas. Por ejemplo “Análisis, desarrollo, pruebas, retroalimentación, vuelta a empezar”, hasta finalmente hacer la entrega.

Ventajas

- **Reduce el riesgo de mala comunicación** con el cliente, gracias a la evaluación constante de resultados. Evitando así fallos acumulativos.
- Abierto a **correcciones y mejoras** del **cliente**.

Desventajas

- Posibilidad de **indecisión del cliente**, puede llevar a múltiples cambios.
- **Costoso**. Cada iteración puede suponer un aumento respecto al coste inicial.
- **No adecuada** para proyectos con **necesidades altamente estables** y bien definidas.

Decisión final

Tras la investigación de múltiples formatos de metodología de desarrollo y la investigación de sus múltiples ventajas y desventajas, además de cuestiones subjetivas, finalmente he decidido escoger la **metodología de desarrollo ágil** para este proyecto.

Al tratarse de un proyecto desarrollado por una única persona, la simplicidad de su ciclo de vida restará en cierto punto el apartado más burocrático de las metodologías, eliminando así trabajo innecesario. Además permitir un desarrollo de estas características, otorga una mayor libertad creativa, mientras que tampoco debo temer las principales desventajas como el movimiento de fechas límites o la alta necesidad de interacción ya que, la primera siempre sucederá mi favor y la segunda, sencillamente no tendrá lugar al ser un solo desarrollador.

2.2. División del trabajo: Épicas

Agrupamos las diferentes historias de usuario en épicas, lo que nos permitirá focalizar la atención en los diferentes conceptos clave del proyecto. Estas son:

- **Épica 1: Gestión de usuario.** Enfocada en los requerimientos de gestión que tienen los usuarios, como registros, autenticaciones o configuraciones respecto a sus preferencias.
- **Épica 2: Gestión de ligas.** Enfocada en los requerimientos de gestión respecto a las ligas a las que pertenece un usuario y que este puede configurar. Engloba múltiples parámetros al respecto. Creación, invitación, modificación de parámetros, etc.
- **Épica 3: Gestión de equipo.** Enfocada en la modificación del equipo y la compraventa de jugadores.
- **Épica 4: Gestión de datos generales.** Enfocada en la obtención de información de los deportes electrónicos ofrecidos, independientemente de pertenecer a una liga o no.
- **Épica 5: Administración de ligas públicas.** Enfocada en la gestión de ligas, pero esta vez a administradores generales del juego que puedan regular el comportamiento de los usuarios.

2.3. Descripción de perfiles de usuario

¿Quién es y qué hace cada persona?

- **Usuario:** Es la persona que accede a la aplicación por alguno de sus medios disponibles con el fin de obtener servicios de ella, y su rol en esta será el de Usuario (ya sea activo o espectador)
- **Administrador:** Es la persona que accede a la aplicación por algunos de sus medios disponibles con el fin de ofrecer servicios para ella, y su rol en esta será el de administrador.

2.4. Identificación de roles

De este modo, es importante identificar los posibles roles de usuario con los que podría contar la aplicación.

- **Usuario:** Dividido en dos tipos para un mejor desglose de requerimientos, que nos permite ver qué funcionalidades deben estar accesibles sin necesidad de pertenecer a ninguna liga y cuáles no. A efectos prácticos, un usuario activo y uno espectador tienen exactamente el mismo rol.

- **Usuario activo:** Es el perfil de usuario que jugará a la actividad principal de la aplicación, que es el modo fantasy.
- **Usuario espectador:** Es el perfil de usuario que no jugará al modo fantasy pero entra para consultar resultados, estadísticas y valoraciones de los partidos de deportes electrónicos.
- **Administrador:** Es el perfil de usuario que administra las ligas oficiales de la aplicación, regulando el comportamiento de los usuarios en la misma.

2.5. Historias de usuario

Plantilla:

Épica de -	Historia de Usuario
HU. 1.1	Nombre de la historia de usuario
Descripción:	-
Pruebas de aceptación:	-
Observaciones:	-

Esta plantilla se compone de los siguiente elementos:

- Épica correspondiente.
- Numeración de Historia de usuario con su respectivo título.
- Descripción de la historia de usuario.
- pruebas de aceptación.
- Posibles observaciones de carácter variado, como sugerencias, especificaciones, etc.

2.5.1. Épica 1: Gestión de Usuario

A nivel práctico, tanto “usuario activo” como “usuario espectador” serán exactamente el mismo perfil, la única diferencia es que el usuario espectador no se ha unido/creado ninguna liga, pero tiene la posibilidad de hacerlo en cualquier momento. Se han separado en dos subgrupos para una mejor separación de la funcionalidad requerida.

Épica 1: Gestión de usuario	Historia de Usuario
HU. 1.1	Registro mediante correo electrónico
Descripción:	Como usuario quiero poder registrarme mediante correo electrónico en la aplicación.
Pruebas de aceptación:	Se deben verificar los casos de registro exitoso, error por correo ya registrado, y error por formato incorrecto de correo o contraseña.
Observaciones:	Se implementará con Spring security.

Épica 1: Gestión de usuario	Historia de Usuario
HU. 1.2.1	Inicio de sesión (correo/usuario)
Descripción:	Como usuario quiero poder iniciar sesión mediante correo electrónico o nombre de usuario.
Pruebas de aceptación:	Se deben verificar los casos de inicio de sesión exitosa, error por correo no encontrado y contraseña incorrecta.
Observaciones:	Se implementará con Spring security.

Épica 1: Gestión de usuario	Historia de Usuario
HU. 1.2.2	Inicio de sesión (Google)
Descripción:	Como usuario quiero poder iniciar sesión mediante mi cuenta de Google sin necesidad de haberme registrado antes en la aplicación.
Pruebas de aceptación:	Se debe verificar que se trate de un email válido, que la contraseña sea inválida (ya que no tiene), y que el login con token sea correcto.
Observaciones:	Se utilizará la API de autenticación de Google basada en OAuth 2.0 y Spring Security OAuth2.

Épica 1: Gestión de usuario	Historia de Usuario
HU. 1.3	Configuración de cuenta
Descripción:	Como usuario quiero poder configurar los principales aspectos de mi cuenta.
Pruebas de aceptación:	Se deben verificar que al cambiar la configuración, el correo no esté ya en uso y la contraseña sea correcta.
Observaciones:	-

2.5.2. Épica 2: Gestión de ligas

Épica 2: Gestión de ligas	Historia de Usuario
HU. 2.1 Creación de ligas	
Descripción: Como usuario activo quiero poder crear ligas tanto públicas como privadas.	
Pruebas de aceptación: Se debe verificar que la liga se ha creado correctamente.	
Observaciones: -	

Épica 2: Gestión de ligas	Historia de Usuario
HU. 2.2 Invitar a ligas	
Descripción: Como usuario activo quiero poder invitar a un usuario a una liga de la que soy partícipe.	
Pruebas de aceptación: Se debe verificar que el código que el código se haya creado correctamente, la liga exista y el usuario no esté ya en esa liga.	
Observaciones: Se implementará como una pila de códigos temporales.	

Épica 2: Gestión de ligas	Historia de Usuario
HU. 2.3 Abandono de liga	
Descripción: Como usuario activo quiero poder abandonar una liga de la que soy miembro.	
Pruebas de aceptación: Se deben verificar que el usuario ya no es parte de esa liga.	
Observaciones: Preguntará con un modal de confirmación para evitar posibles errores humanos. Si el usuario es el administrador, otro usuario de la liga pasará a ser el nuevo administrador. Si se queda sin usuarios, se elimina.	

Épica 2: Gestión de ligas	Historia de Usuario
HU. 2.4 Acceder a ligas asociadas	
Descripción: Como usuario activo quiero ver un display que me permita acceder a las ligas de las que soy miembro.	
Pruebas de aceptación: Se deben verificar que se obtiene la lista de ligas asociadas.	
Observaciones: -	

2.5.3. Épica 3: Gestión de equipo

Épica 3: Gestión de equipo	Historia de Usuario
HU. 3.1 Compra de jugadores en mercado	
Descripción: Como usuario activo quiero poder comprar jugadores del mercado de fichajes.	
Pruebas de aceptación: Se deben verificar que el que el jugador es ahora propiedad del usuario comprador.	
Observaciones: Aplica tanto a jugadores libres (del sistema) como a jugadores de otros usuarios.	

Épica 3: Gestión de equipo	Historia de Usuario
HU. 3.2 Venta de jugadores en mercado	
Descripción: Como usuario activo quiero poder vender jugadores del mercado de fichajes.	
Pruebas de aceptación: Se deben verificar que el jugador esté ahora a la venta.	
Observaciones: Recibirá una lista de ofertas y podrá aceptarlas o declinarlas.	

Épica 3: Gestión de equipo	Historia de Usuario
HU. 3.3 Ofertas privadas	
Descripción: Como usuario activo quiero poder hacer ofertas privadas a usuarios de mi liga para jugadores que tengan otros usuarios en propiedad.	
Pruebas de aceptación: Se deben verificar que el usuario al que se le hace la oferta tiene al jugador especificado.	
Observaciones: Se puede hacer a cualquier jugador del equipo de otro usuario.	

Épica 3: Gestión de equipo	Historia de Usuario
HU. 3.4 Pago de cláusula de rescisión de contrato	
Descripción: Como usuario activo quiero poder pagar cláusulas de rescisión de contrato para hacerme con la propiedad de jugadores de otros usuarios.	
Pruebas de aceptación: Se debe verificar que el usuario al que se le paga la cláusula sea propietario del jugador determinado.	
Observaciones: Se puede hacer a cualquier jugador del equipo de otro usuario.	

Épica 3: Gestión de equipo	Historia de Usuario
HU. 3.5 Aumento de cláusula de rescisión de contrato.	
Descripción: Como usuario activo quiero poder pagar para aumentar el valor de cláusula de rescisión de contrato de un jugador.	
Pruebas de aceptación: Se debe verificar que el usuario sea propietario del jugador determinado.	
Observaciones: Se puede hacer a cualquier jugador de mi equipo.	

Épica 3: Gestión de equipo	Historia de Usuario
HU. 3.6 Alinear jugadores	
Descripción: Como usuario activo quiero poder alinear jugadores en mi equipo.	
Pruebas de aceptación: Se debe verificar que el usuario sea propietario del jugador determinado.	
Observaciones: Si ya hay un jugador alineado en esa posición, se sobrescribirá el nuevo jugador.	

Épica 3: Gestión de equipo	Historia de Usuario
HU. 3.7 Consultar historial	
Descripción: Como usuario activo quiero poder consultar mi historial de transacciones.	
Pruebas de aceptación: Se devuelve una lista con las transacciones de una liga.	
Observaciones: -	

2.5.4. Épica 4: Gestión de datos generales

Épica 4: Gestión de datos generales	Historia de Usuario
HU. 4.1 Revisar resultados de partidos	
Descripción: Como usuario espectador quiero poder consultar los resultados de un partido.	
Pruebas de aceptación: -	
Observaciones: -	

Épica 4: Gestión de datos generales	Historia de Usuario
HU. 4.2 Revisar listado de jugadores de equipos y ligas	
Descripción: Como usuario espectador quiero poder ver las diferentes ligas, equipos y jugadores junto a su información correspondiente.	
Pruebas de aceptación: -	
Observaciones: -	

2.5.5. Épica 5: Administración de ligas públicas

Épica 5: Administración de ligas públicas		Historia de Usuario
HU. 5.1 Gestionar liga pública		
Descripción: Como administrador quiero poder gestionar la configuración en las ligas oficiales.		
Pruebas de aceptación: Se debe verificar que el usuario es de tipo administrador.		
Observaciones: -		

Épica 5: Administración de ligas públicas		Historia de Usuario
HU. 5.2 Amonestar/banear usuario de una liga pública		
Descripción: Como administrador quiero amonestar o banear a un usuario de una liga pública por mal comportamiento.		
Pruebas de aceptación: Se debe verificar que el usuario es de tipo administrador.		
Observaciones: -		

2.6. Historias técnicas

		Historia Técnica
HT. 1 Configuración de Arquitectura Hardware		
Descripción: Diseño y configuración de las arquitecturas hardware que soportarán los servidores backend, frontend, y de base de datos.		
Pruebas de aceptación: Se debe verificar la correcta interconexión entre base de datos, backend y frontend.		
Observaciones: -		

		Historia Técnica
HT. 2 Diseño e implementación de la base de datos		
Descripción: Diseño, normalización, implementación y configuración de la base de datos.		
Pruebas de aceptación: -		
Observaciones: -		

Historia Técnica	
HT. 3	Diseño de arquitectura y configuración del entorno backend
Descripción: Diseño y puesta a punto de la arquitectura backend, de forma que esté lista para comenzar a programar funcionalidad cuando llegue el momento del desarrollo backend.	
Pruebas de aceptación: Se debe verificar el correcto funcionamiento de la arquitectura por capas mediante petición a endpoints de prueba.	
Observaciones: -	

Historia Técnica	
HT. 4	Diseño de arquitectura y configuración del entorno
Descripción: Diseño y puesta a punto de la arquitectura frontend, de forma que esté lista para comenzar a programar funcionalidad cuando llegue el momento del desarrollo frontend.	
Pruebas de aceptación: Se debe verificar el correcto funcionamiento de la arquitectura frontend mediante pantallas componente de prueba.	
Observaciones: -	

Historia Técnica	
HT. 5	Implementación del sistema recolector de datos
Descripción: Es necesario un sistema de recolección de datos de ligas, equipos, jugadores y partidos para al menos un e-sport, que nos permita demostrar el correcto funcionamiento de la aplicación.	
Pruebas de aceptación: Se debe verificar el correcto llenado de la base de datos en el proceso de recolección de datos.	
Observaciones: -	

Historia Técnica	
HT. 6	Diseño e implementación del sistema de ponderación de puntuaciones de desempeño
Descripción: Se debe crear una heurística de puntuación con respecto a las estadísticas que obtengan los jugadores en los partidos que disputen para el e-sport que se vaya a implementar como ejemplo.	
Pruebas de aceptación: Se debe verificar que los jugadores obtienen puntuaciones acordes a la heurística.	
Observaciones: -	

Historia Técnica	
HT. 7	Diseño de implementación heurística de valores de mercado para jugadores.
Descripción: Diseño e implementación de una heurística que otorgue a lo largo del tiempo un valor económico a los jugadores en función de sus estadísticas.	
Pruebas de aceptación: Se debe comprobar que los jugadores obtienen valoraciones acordes a la heurística.	
Observaciones: -	

Historia Técnica	
HT. 8	Diseño e implementación del sistema rotativo de mercado.
Descripción: Diseño e implementación de un sistema rotativo que pondrá a disposición de los usuarios jugadores nuevos por los que pujar cada cierto tiempo.	
Pruebas de aceptación: Se debe comprobar que se ofrecen jugadores para pujar y que estos no son de propiedad de ningún jugador.	
Observaciones: -	

2.7. Gestión de sprints y progresión

A continuación se da una descripción de la división de sprints que se asignará al proyecto para poder llevar una progresión clara del mismo.

2.7.1. Sprint 1

En este sprint se creará el ecosistema de trabajo del proyecto, además de la infraestructura necesaria para trabajar en el mismo.

- **HT 1:** Configuración de arquitectura hardware.
 - **Tarea 1:** Crear y configurar instancias docker para base de datos, frontend y backend.
- **HT 2:** Diseño e implementación de la base de datos.
 - **Tarea 1:** Diseño y normalización de base de datos.
 - **Tarea 2:** Creación de base de datos.
 - **Tarea 3:** Construcción de base de datos a partir del diseño.
- **HT 3:** Diseño de arquitectura y configuración del entorno backend.
 - **Tarea 1:** Diseño inicial del sistema de servicios backend.

- **Tarea 2:** Construcción inicial de la arquitectura backend, con conexión a base de datos y pruebas básicas.
- **HT 4:** Diseño de arquitectura y configuración del entorno Frontend.
 - **Tarea 1:** Diseño inicial del maquetado de la aplicación.
 - **Tarea 2:** Diseño inicial de la arquitectura frontend con pruebas de petición simple a backend.

2.7.2. Sprint 2

En este sprint se creará y gestionará el ecosistema de autenticación de usuario.

- **HU 1.1:** Registro en la aplicación (correo).
 - **Tarea 1:** Programación del servicio backend que introduce nuevos usuarios en base de datos.
 - **Tarea 2:** Maquetación en frontend de la pantalla de registro.
 - **Tarea 3:** Dinamización frontend para realizar petición de registro.
- **HU 1.2.1 y 1.2.2:** Inicio de sesión (Correo y Google).
 - **Tarea 1:** Programación del servicio backend que permite iniciar sesión mediante correo.
 - **Tarea 2:** Creación del servicio backend que permite iniciar sesión mediante google.
 - **Tarea 3:** Maquetación en frontend de la pantalla de inicio de sesión.
 - **Tarea 4:** Dinamización en frontend de la pantalla de inicio de sesión.
- **HU 1.3:** Configuración de cuenta.
 - **Tarea 1:** Programación de los servicios backend que supongan cambios de configuración de la cuenta. Cambio de nombre de usuario, cambio de contraseña, etc.
 - **Tarea 2:** Maquetación en el frontend de la pantalla de cambios de configuración en cuenta.
 - **Tarea 3:** Dinamización en frontend de la pantalla de cambios de configuración.

2.7.3. Sprint 3

- **HU 2.1 y 2.3:** Creación y abandono de ligas.
 - **Tarea 1:** Programación del servicio backend que crea una nueva liga.
 - **Tarea 2:** Programación del servicio backend que te desvincula de una liga.
 - **Tarea 3:** Maquetación frontend de la pantalla de liga (maquetación básica, se irá completando a lo largo del desarrollo).
 - **Tarea 4:** Dinamización de la pantalla de ligas (básica, como movimientos entre pestañas, se irá completando a lo largo del desarrollo).
- **HU 2.2:** Invitación a ligas.
 - **Tarea 1:** Programación servicio backend que devuelve un código de invitación a una liga.
 - **Tarea 2:** Maquetación frontend de la opción de invitación a liga.
 - **Tarea 3:** Dinamización frontend de la opción de invitación a liga.
- **HU 2.4:** Acceso a ligas asociadas.
 - **Tarea 1:** Programación del servicio backend que devuelve la lista de ligas asociadas.
 - **Tarea 2:** Maquetación frontend de la pantalla de ligas.
 - **Tarea 3:** Dinamización frontend de la pantalla de acceso a ligas.
- **HU 5.2:** Amonestación de jugadores
 - **Tarea 1:** Servicio backend que permite expulsar a un jugador de una liga a un administrador.
 - **Tarea 2:** Muestreo frontend a los usuarios administradores que permite ver y expulsar a los jugadores de una liga.

2.7.4. Sprint 4

En este sprint se creará la recolección de datos de jugadores y equipos y la puesta en mercado de los mismos.

- **HT 5:** Implementación del sistema recolector de datos.
 - **Tarea 1:** Programación de la tarea periódica backend que recogerá los datos de las fuentes de información y los normalizará en la estructura de datos de la aplicación.

- **HT 8:** Implementación del sistema rotativo de mercado.
 - **Tarea 1:** Programación del sistema backend que gestiona las rotaciones de jugadores disponibles en el mercado en las diferentes ligas.
- **HU 3.1 y 3.2:** Gestión de transacciones.
 - **Tarea 1:** Programación del servicio backend que gestiona transacciones (aplica a compras y ventas por igual).
 - **Tarea 2:** Maquetación frontend de las opciones de compra de un jugador.
 - **Tarea 3:** Maquetación frontend del historial de transacciones.
 - **Tarea 4:** Dinamización frontend de las operaciones de compra y venta de un jugador.
 - **Tarea 5:** Dinamización frontend del historial de transacciones
- **HU 3.6:** Alineamiento de jugadores en equipo.
 - **Tarea 1:** Programación del servicio backend que me devuelve los jugadores alineados de un usuario en una liga.
 - **Tarea 2:** Maquetación de la sección de alineación en una liga.
 - **Tarea 3:** Dinamización de la sección de alineación en una liga.
- **HU 3.7:** Consultar historial.
 - **Tarea 1:** Servicio backend que devuelve el historial de transacciones de una liga.
 - **Tarea 2:** Muestreo frontend del historial de transacciones de una liga.

2.7.5. Sprint 5

En este sprint se crearán los diferentes sistemas de ponderación de puntos que sean necesarios para cada uno de los deportes electrónicos que se ofrezcan en la aplicación.

- **HT 6:** Diseño e implementación del sistema de ponderación de puntuaciones de desempeño.
 - **Tarea 1:** Diseño y programación backend de las diferentes heurísticas de ponderación de puntos por acción (en caso de ser necesario para los juegos disponibles).
 - **Tarea 2:** Programación del servicio backend que devuelve la puntuación de un jugador en una jornada.

- **HT 7:** Diseño e implementación de la heurística de valores de mercado para jugadores.
 - **Tarea 1:** Diseño y programación de backend de la tarea periódica que irá recalcular el valor de mercado de los jugadores en función de determinados parámetros.

2.7.6. Sprint 6

Durante este sprint se realizarán principalmente labores de maquetación y desarrollo frontend que permitirán a los usuarios ver mediante una interfaz UX/UI un diseño y despliegue de los datos agradable.

- **HU 4.1:** Revisión de los resultados en los partidos
 - **Tarea 1:** Programación del servicio backend que obtiene información de los resultados de los partidos.
 - **Tarea 2:** Programación del muestreo de datos sobre partidos en frontend.
- **HU 4.2:** Revisión del listado de jugadores de equipos y ligas.
 - **Tarea 1:** Programación del servicio backend que me permite obtener información sobre ligas, equipos y jugadores.
 - **Tarea 2:** Programación del muestreo de datos en el frontend permitiendo ver los datos a usuarios no registrados.

2.7.7. Progresión de sprints

A continuación en este apartado se analizarán la estimación de tiempo a invertir en cada tarea y se asignará esa estimación a cada una de ellas con el objetivo de poder realizar finalmente un cálculo de la velocidad para determinar el ritmo y la duración del proyecto.

Estimación

Para determinar la estimación de cada HU/HT se ha creado la tabla 2.1, en la que se determinan y describen los 5 posibles estados que se han determinado para la estimación.

Nota: Es importante destacar que, de ahora en adelante, siempre que se haga referencia a «día», me referiré a un total de 4 horas, que es la máxima cantidad de tiempo diaria que puedo invertir en el Trabajo de Fin de Grado.

Se asignará una estimación a cada una de las historias de usuario, basándome en la experiencia que tenga realizando tareas similares, la cantidad de información que tenga que recabar al respecto para poder realizar la tarea y lo laborioso de la misma.

Estimación	Descripción	Ponderación de tiempo
1	Esfuerzo ligero	Pocas horas (1-4)
2	Esfuerzo leve	Pocos días (1-2)
3	Esfuerzo moderado	Algunos días (3-5)
4	Esfuerzo considerado	Algunos días (6-7)
5	Esfuerzo intenso	algunas semanas (1-2)

Tabla 2.1: Descripción de los estados de estimación posibles.

Desglose de estimación/prioridad para cada HU/HT

HU/HT	Nombre	Estimación	Prioridad	Sprint
HT 1	Configuración de arquitectura HW	4	Alta	1
HT 2	Diseño e implementación de la DB	5	Alta	1
HT 3	Diseño de arquitectura y configuración del entorno backend	3	Alta	1
HT 4	Diseño de arquitectura y configuración del entorno frontend	3	Alta	1
HU 1.1	Registro en la aplicación	3	Alta	2
HU 1.2.1	Inicio de sesión (correo/usuario)	3	Alta	2
HU 1.2.2	Inicio de sesión (Google)	3	Media	2
HU 1.3	Configuración de cuenta	3	Media	2
HU 2.1	Creación de ligas	4	Alta	3
HU 2.2	Invitar a ligas	2	Alta	3
HU 2.3	Abandono de liga	2	Alta	3
HU 2.4	Acceder a ligas asociadas	1	Alta	3
HU 5.1	Gestión de ligas	4	Media	3
HU 5.2	Amonestación de jugadores	2	Media	3
HT 5	Implementación del sistema recolector de datos	5	Alta	4
HT 8	Implementación del sistema de mercado rotativo	5	Alta	4
HU 3.1	Compra de jugadores en mercado	3	Alta	4
HU 3.2	Venta de jugadores en mercado	3	Alta	4
HU 3.3	Ofertas privadas	3	Baja	4
HU 3.4	Pago de cláusula de rescisión de contrato	3	Alta	4
HU 3.5	Aumento de cláusula de rescisión de contrato	3	Media	4
HU 3.6	Alinear jugadores	3	Alta	4
HU 3.7	Consultar historial	2	Media	4
HT 6	Heurística de ponderación de puntuación	5	Alta	5
HT 7	Heurística de valores de mercado	5	Alta	5
HU 4.1	Revisar resultados de partidos	3	Alta	6
HU 4.2	Revisar listado de jugadores, equipos y ligas	3	Alta	6

Tabla 2.2: Desglose de estimación, prioridad y sprint para cada una de las historias de usuario/técnicas.

Velocidad

Podemos calcular los puntos de velocidad de cada una de las historias como la suma de la estimación de cada una de sus tareas [Atl24]. De este modo obtenemos un resultado como el de la tabla 2.3:

Sprint	Velocidad
1	15
2	12
3	15
4	30
5	10
6	6

Tabla 2.3: Velocidad por Sprint.

Con este cálculo de velocidad y sabiendo la ponderación de tiempo que corresponde a cada estimación (ver tabla 2.2), podemos calcular una cantidad de tiempo por sprint como la división de la velocidad entre el mayor número de estimación posible. El cociente de esta división, lo multiplicamos por la máxima ponderación de tiempo posible y sumándole la máxima ponderación de tiempo posible correspondiente al resto de la división, obteniendo así la cantidad de tiempo por sprint en el peor de los casos.

Por ejemplo, para el sprint 2: $15 \text{ (velocidad)} / 5 \text{ (estimación)} = 2 \text{ (cociente), } 2 \text{ (resto). } 2 \text{ (cociente)} * 14 \text{ días (máxima estimación de tiempo posible)} + 2 \text{ (máximo número de días correspondiente a la estimación del resto)} = 30 \text{ días.}$

Siguiendo este ejemplo, se han calculado la estimaciones de los seis sprints existentes, cuyos resultados se pueden ver en la tabla 2.4.

Sprint	Tiempo (días)
1	42
2	30
3	42
4	84
5	28
6	16

Tabla 2.4: Cálculo temporal de sprints para el peor caso posible.

Como podemos observar, salen tiempos relativamente grandes, sin embargo cabe recordar que, siguiendo la filosofía del análisis algorítmico y con el objetivo de evitar sorpresas, nos hemos posicionado siempre en la peor opción posible.

La figura 2.1 muestra las tareas del proyecto, sus fechas de inicio y finalización, y las dependencias entre ellas [Ger12].

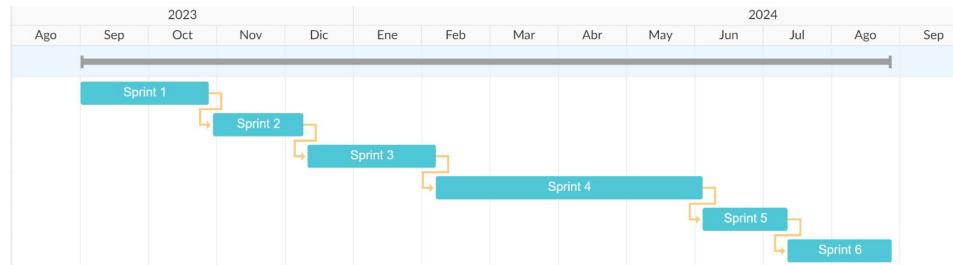


Figura 2.1: Progresión de los Sprints en diagrama de Gantt

2.8. Presupuesto

Para realizar la aplicación, se llevarán a cabo dos presupuestos. Un primer presupuesto durante la creación de la aplicación, y otro presupuesto acumulativo mensual para el mantenimiento de la aplicación. El presupuesto final contendrá el de mantenimiento dentro del marco de desarrollo que se haya tenido que mantener.

Presupuesto de desarrollo

El presupuesto de desarrollo software se basa en los siguientes ítems:

- **Backend:** El salario medio de un desarrollador Java en España es aproximadamente 2667€ mensuales o 16.41€ la hora [est24c].
- **Frontend:** El salario medio de un desarrollador Angular en España es aproximadamente 2500€ mensuales o 15.38€ la hora [est24b].
- **Base de datos:** El salario medio de un programador para tareas relacionadas con bases de datos en España es aproximadamente de 2646€ mensuales o 16.28€ la hora [est24a].
- **Devops:** El salario medio de un desarrollador DevOps en España es aproximadamente de 3750€ al mes o 23.08€ la hora [est24d].
- **Maquetación:** El salario medio de un maquetador en España es aproximadamente 2021€ al mes o 12.44€ la hora [est24e].

Sueldo desarrollador

Teniendo en cuenta la información recabada en el apartado anterior, y que el desarrollador será una única persona, se realizará un cálculo de sueldo medio como la suma de todos los salarios entre el número de trabajos (5) (ver tabla 2.5).

Trabajo	Salario medio
Backend - Java	16.41€/h
Frontend - Angular	15.38€/h
Base de datos	16.28€/h
DevOps	23.08€/h
Maquetación	12.44€/h
Total	16.718€/h

Tabla 2.5: Desglose de salario medio en función del campo de desarrollo.

De esta forma, podremos calcular el gasto en desarrollo en cada uno de los sprints y la progresión de presupuesto invertido a lo largo del tiempo. Calcularemos el gasto de cada sprint como salario medio total multiplicado por 4 horas multiplicado por número de días.

Sprint	Duración (días)	Gasto de Sprint	Gasto acumulado
SPRINT 1	42	2808.62 €	2808.62 €
SPRINT 2	30	2808.62 €	4814.78 €
SPRINT 3	42	1738.67 €	7623.40 €
SPRINT 4	84	5617.24 €	13240.65 €
SPRINT 5	28	1872.41 €	15113.07 €
SPRINT 6	16	1069.95 €	16183.02 €

Tabla 2.6: Desglose del gasto en desarrollo por sprint.

Calcularemos el gasto proporcional de cada tarea (ya sea historia técnica o de usuario), obteniendo en primer lugar el gasto por punto de estimación (gasto de sprint (ver tabla 2.6) / velocidad de sprint (ver tabla 2.3)). Para después multiplicar el gasto por punto por la estimación de la HU/HT determinada (estimaciones en tabla 2.1).

Sprint	HU/HT	Estimación	Gasto proporcional (€)
1	HT 1	4	748.96
1	HT 2	5	936.20
1	HT 3	3	561.72
1	HT 4	3	561.72
2	HU 1.1	3	702.15
2	HU 1.2.1	3	702.15
2	HU 1.2.2	3	702.15
2	HU 1.3	3	702.15
3	HU 2.1	4	403.95
3	HU 2.2	2	201.97
3	HU 2.3	2	201.97
3	HU 2.4	1	100.98
3	HU 5.1	4	403.95
3	HU 5.2	2	201.97
4	HT 5	5	1404.31
4	HT 8	5	1404.31
4	HU 3.1	3	842.59
4	HU 3.2	3	842.59
4	HU 3.3	3	842.59
4	HU 3.4	3	842.59
4	HU 3.5	3	842.59
4	HU 3.6	3	842.59
4	HU 3.7	2	561.72
5	HT 6	5	936.20
5	HT 7	5	936.20
6	HU 4.1	3	534.97
6	HU 4.2	3	534.97

Tabla 2.7: Desglose de gasto proporcional por sprint.

Mantenimiento online de la aplicación

Para poder desplegar la aplicación en la web, debo mantener, un servidor backend, un servidor frontend y un servidor de base de datos. Estos se harán a través del proveedor de microservicios en la nube Google Cloud, que ofrece un servicio de instancias de máquinas virtuales “Compute Engine”. Ubicamos tres máquinas virtuales, en Países Bajos, servidor europe-west4.

- **n1-standard-4** para el backend, ya que requerirá mayor potencia de cómputo al ser el que procesará los cálculos y transacciones.
- **n1-standard-2** para frontend y base de datos.

Servidor	Tipo máquina	CPUs virtuales	Memoria	Precio/h
BackEnd	n1-standard-4	4	15 GB	0.2 €/h
FrontEnd	n1-standard-2	2	7.5 GB	0.098 €/h
Base de datos	n1-standard-2	2	7.5GB	0.098 €/h

Tabla 2.8: Gasto de las máquinas virtuales ofrecidas por «Compute Engine» de Google [Goo24b].

La mayor parte del desarrollo de la aplicación se llevará a cabo de forma local, pero habrá que realizar pruebas en los servidores de producción ya que hay situaciones en las que es necesario. Mi experiencia personal en el campo laboral me ha dejado entrever que aproximadamente el 30 % del tiempo de un desarrollo, debe probar software en servicios reales, por lo que esa será la cifra que utilizaré.

Teniendo el cuenta la cantidad de días de desarrollo y precios de servicios de máquina virtual, podemos calcular cuánto gastaremos en estas durante el desarrollo de la aplicación:

$$158 \text{ días} * 4 \text{ horas} * 30 \% * (0.2 + 0.098 + 0.098) = \mathbf{75.0816 \text{ € en mantenimiento online}} \text{ durante el desarrollo.}$$

Una vez se ha finalizado el desarrollo, para lanzar la página a la web, además del mantenimiento de los servidores, hay que encontrar un dominio. Hostinger los ofrece por 9.99 € al año, lo que son 0.833 € al mes [Hos24].

Por lo que una vez se ha finalizado el desarrollo, el **mantenimiento de la web en servidores** es de $30.5 \text{ días} * 24 \text{ h} * (0.2 + 0.098 + 0.098) + 0.8325 = \mathbf{290.7045 \text{ €/mes}}$

Materiales

Ordenador Para realizar el proyecto se utilizará un Lenovo Legion Y520, cuyo precio es de 842.41 €, y teniendo en cuenta que el tiempo medio de vida de un ordenador es de 4 años, y que el proyecto durará aproximadamente unos 6 meses, podemos calcular el coste total como $842.41 \text{ €} / (12 \text{ meses} \times 4 \text{ años}) \times 6 \text{ meses} = 105.3 \text{ €}$

Internet En mi situación, teniendo un gasto mensual de 6.6€ en internet (en piso compartido) con la compañía DIGI. Sabiendo que en el piso en el que me encuentro hay de media 305.7 horas en las que hay un dispositivo conectado a internet, por lo que en total en los 158 días habrá habido una conexión de 778.5 horas. Podemos calcular el total como $(6.6 \text{ €} \times 79 \text{ días} \times 8 \text{ h} / 778.5 \text{ h}) \times 6 \text{ meses} = 32.14 \text{ €}$

$$\text{Total de materiales} = 105.3 \text{ €} + 32.14 \text{ €} = 136.44 \text{ €}.$$

Licencias y servicios externos No ha sido necesaria la compra de licencias para este proyecto, ya que tanto la base de datos en MariaDB, como los entornos de desarrollo, son de código abierto y gratuitos.

Resumen final

Categoría	Precio
Desarrollo	16183.02 €
Mantenimiento online	75.0816 €
Material	136.44 €
Licencias y servicios externos	0 €
Total	16394.52 €

Tabla 2.9: Total presupuesto resumido.

Capítulo 3

Implementación

3.1. Estudio de tecnologías existentes y elección

En este apartado se realizará un estudio de las principales tecnologías que se han tenido en cuenta para realizar este proyecto, con sus correspondientes ventajas y desventajas, y se dará una explicación de la decisión final en cada una de ellas. Cabe destacar que estas son las tecnologías más utilizadas y conocidas, pero el abanico de posibilidades en el mundo del desarrollo software es muy amplio.

Base de datos

Para este apartado se han tenido en cuenta dos posibilidades, las bases de datos relacionales y las bases de datos no relacionales.

Oracle - MySQL [Ora24]]

Oracle - MySQL se trata de un sistema de gestión de bases de datos E/R, de código abierto y propiedad de Oracle, que cuenta con una gran popularidad en el mundo de las bases de datos.

Ventajas

- **Amplia comunidad:** Cuenta con una gran comunidad de usuarios, lo que se traduce en recursos y soporte disponible.
- **Rendimiento:** El rendimiento de MySQL, junto a su rápida capacidad de respuesta la hace ampliamente utilizada en aplicaciones web y de comercio electrónico.
- **Facilidad de uso:** Fácil instalación y configuración, junto a una amplia variedad de herramientas de administración.
- **Buena escalabilidad** vertical y horizontal.

- **Seguridad:** Robustas características de seguridad, incluyendo autenticaciones y cifrado de datos.
- En caso de usar la versión de pago de Oracle, se ofrece asistencia técnica.

Desventajas

- **Propiedad de Oracle:** No es código libre, por lo que puede estar sujeto a derechos de autor/propiedad.
- **No ofrece todas sus características** en la versión de código abierto.
- Planes de precios muy **costosos**.

MariaDB

MariaDB es un sistema de gestión de bases de datos E/R, que se desarrolló como una bifurcación de MySQL, que ofrece varias ventajas y mejoras con respecto a su predecesor.

Ventajas

- **Alta compatibilidad con MySQL:** Lo que facilita enormemente las conversiones de un sistema a otro, además de facilitar también su uso.
- **Código abierto:** MariaDB cuenta con una licencia de software libre GLP, por lo que el acceso es gratuito y sin costes de licencia.
- **Amplia comunidad:** MariaDB también cuenta con una gran comunidad de desarrolladores que ofrecen soporte técnico.
- **Mejor rendimiento:** Muchas comparaciones entre los dos principales competidores de bases de datos relacionales (MySQL y MariaDB) concluyen que MariaDB suele obtener un ligero mejor rendimiento, especialmente en situaciones de concurrencia.
- **Escalabilidad:** Es escalable tanto vertical como horizontalmente.
- **Buena seguridad:** Cuenta con algunas características de seguridad avanzadas, como la autenticación PAM y la encriptación de datos en reposo.

Desventajas

- **Compatibilidad no garantizada al 100 %:** Aunque suele ser compatible con MySQL, no puede garantizar que se harán los traspasos de forma completamente efectiva.

- **Menor soporte:** Al ser un software de licencia libre y gratuito, no cuenta con un soporte especializado y que asegure la atención tal y como lo tiene Oracle.
- **Menor base de usuarios:** Aunque cuenta con una buena cantidad de usuarios, sigue sin llegar a la base de usuarios que ha obtenido MySQL.

MongoDB [Mon24]

MongoDB es un sistema de gestión de bases de datos NoSQL orientado a documentos, que se ha vuelto muy popular durante los últimos años en el mundo de las aplicaciones web y móviles.

Ventajas

- **Modelo flexible de datos:** Mongo permite agregar y eliminar campos a los documentos sin afectar a la estructura de la base de datos.
- **Rendimiento:** Cuenta con un gran rendimiento y eficiente gestión de consultas e inserciones. Especialmente en las aplicaciones que implican concurrencia.
- **Replicación y disponibilidad:** Mongo admite réplicas, lo que supone una alta disponibilidad, ofreciéndonos además la posibilidad de recuperación ante desastres.

Desventajas

- **Falta de unión (JOIN):** MongoDB no admite operaciones de unión entre colecciones, lo que puede requerir que los datos se diseñen de distintas forma y aumente la complejidad de ciertas consultas.
- **Consumo de recursos del servidor:** MongoDB puede ser intensivo en el uso de recursos del servidor. La correcta configuración y dimensionamiento de datos son esenciales.
- **Curva de aprendizaje:** La adopción de MongoDB puede requerir un considerable tiempo de aprendizaje para comprender el modelo de datos, consultas y configuraciones.
- **Seguridad:** Una mala configuración de MongoDB puede dar lugar a fallos de seguridad y vulnerabilidades importantes.
- **Débil consistencia:** La estrategia de consistencia eventual de MongoDB no garantiza datos coherentes.

Backend

PHP [Php24]

PHP es un lenguaje de programación de propósito general, pero que gracias a su funcionalidad, se ha utilizado históricamente en el contexto backend del desarrollo web. Es además uno de los lenguajes de programación web más populares y utilizados de la historia, siendo que aproximadamente el 79.2 % de la web tiene un backend en PHP [Kin24].

Ventajas

- **Amplia adopción:** PHP tiene un largo bagaje, por lo que hay una gran cantidad de recursos, bibliotecas y frameworks disponibles para este.
- **Velocidad de desarrollo:** PHP facilita el desarrollo mediante una sintaxis simple y muchas funciones incorporadas para tareas comunes.
- **Compatibilidad con Bases de datos:** PHP cuenta con una fácil y buena integración con bases de datos como MySQL, SQLite o PostgreSQL, lo que facilita la creación de backends que necesiten almacenamiento.

Desventajas

- **Rendimiento:** Suele tener un rendimiento inferior con respecto a otros lenguajes de programación como NodeJS o Java, especialmente en la gestión de concurrencia.
- **Historia de seguridad:** PHP ha tenido a lo largo de su historia, como consecuencia de la falta de cuidado en la escritura de código, múltiples brechas de seguridad.
- **Inconsistencias de sintaxis:** Como consecuencia de su largo bagaje y evolución.
- **Sin tipado estricto:** Al ser un lenguaje de tipado débil dinámico, las variables podrían cambiar durante la ejecución, lo que hace difícil la depuración en algunas ocasiones.

NodeJS [Nod24]

Se trata de una plataforma de desarrollo software en el lado del servidor, de código abierto basado en javascript y que está en alza, como consecuencia de la oleada de programación y ecosistemas Javascript en estos últimos años.

Ventajas

- **Velocidad de ejecución:** NodeJS compila el código JavaScript en código nativo, lo que lo hace extremadamente rápido en operaciones E/S.

- **Dirigido a eventos y no bloqueante:** Node utiliza una arquitectura de eventos que permite realizar operaciones no bloqueantes, lo que permite aumentar altamente la eficiencia de las aplicaciones en tiempo real.
- **Amplia comunidad y ecosistema de paquetes:** Ecosistema de módulos (npm) que facilita la integración de bibliotecas de terceros.
- **Ecosistema JavaScript:** En caso de tener un frontend en javascript, se puede generar un ecosistema en el que sea más fácil la comunicación entre entornos.

Desventajas

- **Hilo de ejecución único:** NodeJS usa un modelo de un único hilo de ejecución, lo que puede ser un problema en tareas intensivas de CPU.
- **Complejidad en el código asíncrono:** Se requiere un buen diseño y experiencia para manejar el llamado “promise hell” de manera efectiva.
- **Módulos no estables:** Los módulos de terceros no siempre son todo lo estables que nos gustaría.

Java - SpringBoot [Spr24b]

Java es un histórico lenguaje de programación de propósito general, compilado y orientado a objetos, que cuenta con una amplia comunidad. Además, el framework SpringBoot cuenta con soporte para java, potenciando la velocidad de creación de microservicios para el desarrollo web.

Ventajas

- **Facilidad de configuración:** Simplifica en gran medida la configuración de aplicaciones Java al proporcionarnos una configuración por defecto basada en convenciones.
- **Productividad:** Está diseñado para facilitar el desarrollo rápido ya que cuenta con un conjunto de herramientas y bibliotecas que permiten crear aplicaciones de forma eficiente.
- **Amplia Comunidad:** Que se traduce en un buen soporte técnico.
- **Despliegue sencillo:** Empaquetado en JAR/WAR autónomo facilitado por SpringBoot.

Desventajas

- **Complejidad:** A pesar de que SpringBoot simplifica la escritura de código, el ecosistema puede tener una alta curva de dificultad para desarrolladores novatos.

- **Consumo de recursos:** Las aplicaciones en SpringBoot pueden suponer un mayor gasto de recursos con respecto a su competencia.
- **Exceso de abstracción:** SpringBoot se asocia a una abstracción excesiva que dificulta las labores de depuración.
- Jars autogenerados de **alto peso**.

Posibilidad de sobrecarga: SpringBoot puede introducir sobrecarga en aplicaciones pequeñas.

Frontend

Flutter [Flu24]

Se trata de un framework de código abierto para DART, creado por Google, cuyo objetivo es facilitar el desarrollo frontend de aplicaciones para móviles y web. Es famoso por su buen rendimiento y su amigable interfaz de usuario.

Ventajas

- **Desarrollo multiplataforma:** Tanto en aplicaciones Android como aplicaciones IOS desde un único código base.
- **Rendimiento:** Utiliza el motor gráfico Skia para renderizar la interfaz, que proporciona un rendimiento rápido.
- **Hot reload:** Permite agilizar el proceso de desarrollo viendo los cambios en tiempo real.
- **Comunidad activa:** Con amplia cantidad de recursos y bibliotecas disponibles.

Desventajas

- **Tamaño:** Las aplicaciones flutter suelen tener un tamaño más grande al de las aplicaciones nativas.
- **Problemas de compatibilidad:** Puede tener dificultades en compatibilidad con las últimas plataformas nativas.
- **Módulos de tercero limitados:** Aunque tiene una gran biblioteca de paquetes, no es tan completa como la de las plataformas nativas.
- **No permite desarrollo web:** Es solo para aplicaciones móviles.

AngularJS [Ang24]

AngularJS es un framework para JavaScript, creado por google que se utiliza para crear aplicaciones frontend, en el lado del cliente. Y que es conocida por su buen rendimiento y su flexibilidad, además de su simplicidad en el uso.

Ventajas

- **Fácil aprendizaje:** Al utilizar HTML como plantilla y extender su sintaxis, se facilita el aprendizaje a las personas familiarizadas con HTML.
- **Enlace de datos bidireccional:** Los cambios en el modelo se reflejan automáticamente en la vista y viceversa. Simplificando así la actualización de la interfaz de usuario.
- **Manejo de eventos simplificado:** Cuenta con directivas personalizadas que facilitan la vinculación de eventos y acciones específicas a elementos HTML.
- **Arquitectura MVC:** Angular promueve una estructura bien organizada desde el

Desventajas

- **Rendimiento lento:** AngularJS puede tener un rendimiento lento en comparación a tecnologías más recientes, especialmente en aplicaciones grandes y complejas como consecuencia del sistema de detección de cambios.
- **Curva de aprendizaje:** Aunque tenga una fácil aprendizaje gracias a la extensión de HTML, tiene unos picos de aprendizaje altos como consecuencia de su modelo.
- **Escaso ecosistema:** En comparación con otros frameworks, AngularJS cuenta con un ecosistema de bibliotecas más limitado.

VueJS [Vue24]

Ventajas

- **Buena documentación:** documentación completa y amigable que facilita la compresión y la resolución de problemas.
- **Flexibilidad y escalabilidad:** Altamente modular, te permite integrarlo gradualmente en aplicaciones existentes.
- **Reactividad:** VueJS utiliza un sistema de reactividad que provoca una sencilla y eficiente actualización de interfaz.
- **Componentes:** Promueve la creación de componentes reutilizables, facilitando así la construcción de interfaces de usuario consistentes y mantenibles.

Desventajas

- **Menor adopción que React o Angular:** Aún no cuenta con tanta adopción como React o como Angular, por lo que la disponibilidad de recursos y bibliotecas de terceros es bastante limitada.
- **Menor flexibilidad en algunos aspectos:** Aunque ofrece mucha flexibilidad, hay casos en los que otras tecnologías como React o Angular pueden ser más adecuados debido a características específicas que no están en VueJS.

Decisión final de tecnologías

Tras plantearme las diferentes ventajas y desventajas de las diferentes tecnologías disponibles, entre ellas algunas tecnologías más conocidas por ser emergentes y otras más conocidas por su consistencia o bagaje, además de tener en cuenta cuestiones subjetivas como las que se detallarán más adelante, finalmente se han escogido las siguientes tecnologías:

Base de datos

MariaDB ha sido el sistema de gestión de bases de datos escogido.

El proyecto no requiere de almacenajes de información masivos y la información almacenada debe ser consistente por lo que lo ideal es un modelo relacional en lugar de un modelo NoSQL. Además este modelo facilita el diseño normalizado y el modelado de datos complejos en el servidor backend para su posterior gestión.

Dentro de los modelos relacional, nos hemos quedado con el sistema gestor de bases de datos de MariaDB, principalmente porque ofrece las mismas características y prestaciones que ofrece Oracle MySQL (teniendo en cuenta que no nos planteamos la opción de pago de esta), pero con licencias de uso libres y gratuitas.

Backend

Java - Springboot es el lenguaje (junto a un framework) escogido para realizar el backend (servicios) de la aplicación web.

Se ha escogido esta opción dado el intensivo uso que hará la aplicación de las conexiones a bases de datos para recopilar información, por lo que bibliotecas como JPA [Spr24a] aportan gran dinamismo a la programación del CRUD. Springboot nos permitirá implementar estas operaciones más fácilmente que otras tecnologías, y nos asegurará hacerlo de una forma segura. Hay que añadir la importancia que operar con un lenguaje robusto y seguro como lo es este, teniendo en cuenta que se va a hacer un uso bastante intensivo del procesamiento de datos durante la obtención y normalización de la

información en las tareas programadas, en las que se nos permite abstraernos del manejo de hilos para una fácil interacción con las mismas.

Frontend

Angular ha sido la tecnología escogida para desarrollar el frontend de la aplicación web.

Teniendo en cuenta que se trata de una aplicación de “gestión”, una estructura bien ordenada como la que ofrece angular va a venir perfecta. Además de que la gestión de contenidos bidireccional agiliza en gran medida la creación del frontend. Un punto importante también es mi experiencia personal como desarrollador frontend con este framework, el cual me va a permitir tener un buen nivel del mismo sin necesidad de recurrir a demasiado aprendizaje o recursos externos.

3.2. Gestión de tests unitarios y de integración

Los tests unitarios son desarrollos software cuya finalidad es verificar la correcta funcionalidad de diferentes partes del código aisladas, de manera independiente a sus dependencias y con posibilidad de automatización. Gracias a estos tests, podemos detectar errores o incongruencias en nuestro código de manera directa o protegernos ante posibles fallos a la hora de realizar cambios en fragmentos de código que han sido desarrollados con anterioridad [Pit24].

Se pueden hacer prácticamente tantos tests unitarios como se quieran hasta llegar incluso a obtener una cobertura frente al 100 % del código escrito, pero es importante destacar que no podemos obsesionarnos con ellos, ya que también toman una buena parte del tiempo de desarrollo. Por esto, hemos decidido crear los tests unitarios en función de las historias de usuario, de forma que los tests aseguren el correcto funcionamiento de todas y cada una de las historias de usuario, y por consiguiente, la totalidad de la lógica de negocio de la aplicación.

Por otro lado se encuentras los tests de integración. Estos consisten ya no en comprobar código de forma aislada como hacíamos en los tests unitarios, si no en todo lo contrario, comprobar la correcta cohesión, interacción y funcionamiento entre los módulos software que componen un proceso. Para ello, realizaremos peticiones a los principales endpoints de los controladores para cada uno de sus posibles cauces de conexión de módulos, lo que nos permitirá comprobar el correcto funcionamiento de la cohesión entre módulos [Pos24].

De este modo, para cada uno de los sprints, se incluirá un apartado de tests en el que se dará una lista de tests unitarios que cubren el código, asignándole a cada uno una o varias historias de usuario, y una lista de fotografías mostrando la correcta respuesta de los test de integración de los principales endpoints creados a lo largo de cada sprint en los que se trata de destacar la respuesta válida (o inválida en caso de ser necesario) de la petición y no estrictamente su contenido.

Aplicar una metodología TDD¹, ha sido una idea que se ha contemplado, sin embargo esta ha sido finalmente descartada dadas las conclusiones del artículo [Hak05], el cual dice que que, si bien es cierto que el TDD genera resultados más consistentes, más no resultados de mayor calidad técnica. Por tanto, al no ser este un proyecto que destaque especialmente por necesidad de robustez, se ha decidido priorizar la inversión de tiempo adicional que podría suponer seguir este enfoque en focalizar la calidad técnica del desarrollo.

Consideraremos correcto un sprint una vez que cumpla todas las pruebas de aceptación de sus correspondientes historias de usuario e historias técnicas, de modo que se crearán los tests unitarios y de integración al rededor de estas pruebas de aceptación; al final de cada apartado de tests, se incluirá un apartado en el que se aclarará que test unitario/integración, equivale a que prueba de aceptación.

3.3. Sprint 1

3.3.1. Introducción al Sprint 1

Durante este primer sprint procederemos con la puesta a punto del proyecto y sus configuraciones iniciales, con el objetivo de dejarlo preparado para la implementación de funcionalidades. Para ello se seguirá la estructura de Historias Técnicas del Sprint 1, qué es la siguiente:

- **HT 1:** Configuración de despliegue en HW.
 - **Tarea 1:** Crear y configurar instancias docker para base de datos, frontend y backend.
- **HT 2:** Diseño e implementación de la base de datos.
 - **Tarea 1:** Diseño y normalización de base de datos.
 - **Tarea 2:** Creación de base de datos.
 - **Tarea 3:** Construcción de base de datos a partir del diseño.

¹Test Driven Development.

- **HT 3:** Diseño de arquitectura y configuración del entorno Backend.
 - **Tarea 1:** Diseño inicial del sistema de servicios backend.
 - **Tarea 2:** Construcción inicial de la arquitectura backend, con conexión a base de datos y pruebas básicas.
- **HT 4:** Diseño de arquitectura y configuración del entorno Frontend.
 - **Tarea 1:** Diseño inicial del maquetado de la aplicación.
 - **Tarea 2:** Diseño inicial de la arquitectura frontend con pruebas de petición simple a backend.

3.3.2. HT 1: Configuración de despliegue en HW.

Durante esta historia técnica se ha realizado la Tarea 1, que consiste en la creación y configuración de las instancias docker que se utilizarán para el despliegue de la base de datos, el frontend y el backend.

Para ello, comenzaremos explicando qué es docker y su contenido relacionado. Llamamos contenedor a unidades de software estándar que empaquetan código junto a sus dependencias de manera que las aplicaciones funcionan de forma rápida e independiente al entorno en el que se estén ejecutando. Por otra parte, una imagen es una plantilla estática que contiene las definiciones necesarias para levantar ese contenedor; en la comunidad de docker se le suele referenciar con «una receta». De esta forma, la imagen pasa a ser un contenedor cuando instanciamos una copia de esta para ejecutarla, perdiendo así su inmutabilidad (ver figura 3.1). [Doc24].

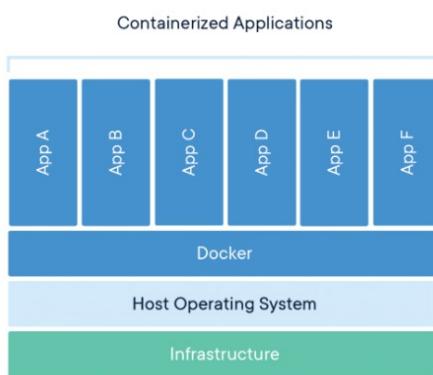


Figura 3.1: Infraestructura docker [Doc24].

Podría parecer que este es un trabajo innecesario, ya que podría simplemente desplegar las aplicaciones en un servidor de forma nativa y lista,

sin embargo es una tarea que aportará en el futuro un gran modularidad, escalabilidad y movilidad, permitiéndonos desplegar la página en servidores propios, AWS², Google Cloud o cualquier sistema de microservicios en muy pocos clicks y lo más importante, con la seguridad de saber que se ha hecho a partir de una estructura normalizada y completamente independiente al entorno de ejecución.

Creación de la arquitectura de contenedores

Contaremos con dos enfoques posibles a la hora de abordar la arquitectura de despliegue, ambos se prepararán y configurarán para en el futuro escoger una u otra versión:

- **Despliegue de cada contenedor en distintos servidores:** Esta es el modelo que se seguirá una vez la aplicación sea lanzada a producción, con el objetivo de maximizar la modularidad de la arquitectura y la eficiencia en el procesamiento.
- **Despliegue de los tres contenedores en el mismo servidor:** Este modelo se seguirá durante el desarrollo de la aplicación, facilitando así los procesos de actualización en el entorno.

De este modo, explicaré la lógica al rededor de cada uno de los contenedores de forma individual, lo cual nos permitiría desplegarlos de forma independiente en producción, y finalizaré explicando la unificación de contenedores que usaré durante el desarrollo.

Imagen de la base de datos³

Los principales puntos a destacar a la hora de la creación del Dockerfile de la base de datos son:

- La asignación de usuarios y contraseñas iniciales mediante variables de entorno.
- El guardado de archivos permanentes. Lo que supone persistencia en la base de datos si se transfiriese el contenedor de una máquina a otra o si este se apagase.
- Mapeo de puertos 3306 a 3306, lo que supone que seguimos manteniendo el mismo puerto para la base de datos.

²Amazon Web Services.

³Código de la imagen para la base de datos.

Imagen del backend⁴

Los principales puntos a destacar a la hora de la creación del Dockerfile del backend son:

- Copia del ejecutable del backend (archivo .war) al directorio de ejecución en el backend.
- La ejecución del conjunto de archivos .jar mediante el archivo .war.
- Mapeo de puerto 8080 al puerto deseado en el que se quiere exponer el backend.

Imagen frontend⁵

Los principales puntos a destacar a la hora de la creación del Dockerfile del frontend son:

- Copia del directorio de distribución que node genera para un proyecto angular al directorio de ejecución.
- La ejecución de un servidor NodeJS que levante el servicio web de la aplicación.

Arquitectura común⁶

En esta otra opción disponible, se ejecutarán todas las máquinas virtuales en un solo servidor host. Para ello he creado un archivo «Docker-compose.yml», el cual me permite configurar todas las instancia de los contenedores de una sola vez.

Además, como los contenedores estarán en un mismo host, será necesario configurar una conexión entre las instancias. Para ello he creado una subred, junto a su pasarela correspondiente (10.10.10.0/24), y he asignado a cada uno de los contenedores una dirección ip de esta pasarela.

- **Base de datos:** 10.10.10.2
- **Servidor Backend:** 10.10.10.3
- **Servidor frontend:** 10.10.10.4

⁴Código de la imagen para el backend.

⁵Código de la imagen para el frontend.

⁶Código del Docker-compose.

Podemos destacar que esta versión en una sola máquina host es especialmente útil para el periodo de desarrollo ya que esto nos permite tener un multientorno de trabajo bastante estable. Por ejemplo, durante el desarrollo del apartado frontend de un determinado sprint, podríamos tener a la misma vez ejecutándose una instancia docker del backend la cual sabemos que es una versión estable del sprint anterior y otra versión de desarrollo del nuevo sprint en local. De este modo, nos bastaría con ir cambiando en el desarrollo del frontend entre apuntar a la versión estable de docker o apuntar a la nueva versión del desarrollo, para depurar o encontrar fallos que se hayan podido introducir en el sprint que se está desarrollando actualmente.

3.3.3. HT 2: Diseño e implementación de la base de datos.

Para tratar de diseñar la base de datos de la forma más exhaustiva posible, lo haremos mediante un proceso dividido en tres secciones: Análisis de requisitos de información, diseño conceptual y diseño físico.

Análisis de requisitos de información

Los tres principales requisitos de información para este proyecto son los siguientes:

- **Información relacionada al usuario** como su usuario, correo, contraseña, etc.
- **Información fantasy** englobando aquí todo lo relacionado con la construcción de ligas; nombre, que usuarios están asociados a que liga, propiedad de jugadores, lista de mercado, historial de transacciones, etc.
- **Información del deporte electrónico** como las ligas reales, equipos, jugadores y partidos. Junto a la información que todo esto conlleva, como identificadores, nombres, fotografías y la relación entre ellos.

Esquema E/R conceptual y físico

El diagrama entidad relación es una representación gráfica que representa un diseño de base de datos; mostrándonos de forma clara que entidades debería modelar una base de datos, sus atributos y las relaciones entre estas entidades. Con el objetivo de acercarnos a un primer modelado de datos, se ha creado este diagrama ver figura 3.2. A continuación se desglosará que representan todas y cada una de las entidades, relaciones y atributos del modelo que se ha realizado:

Entidades:

- **User:** Almacena la información inherente al usuario.
 - **uuid:** Identificador único del usuario.
 - **mail:** Correo electrónico del usuario.
 - **username:** Nombre del usuario.
 - **pass:** Contraseña del usuario.
 - **admin:** Determina si el usuario es administrador o no.
- **League:** Almacena la información inherente a la liga de los usuarios.
 - **uuid:** Identificador único de la liga.
 - **name:** Nombre que el usuario le ha puesto a la liga.
 - **is public:** Determina si una liga es pública o no.
 - **start type:** Determina el modo de comienzo en una liga (con o sin reparto de jugadores).
 - **active clause:** Determina si el sistema de cláusulas está activo.
 - **real league:** Determina a qué evento deportivo corresponde esta liga.
- **Post:** Almacena información correspondiente a las entradas que irán apareciendo en la “feed” de cada liga, como pueden ser la compraventa de jugadores o los clausulazos.
 - **id:** Identificador único para cada post.
 - **prevowner:** Dueño anterior del jugador sobre el que recae la acción del post.
 - **player:** Jugador anterior.
 - **date:** Fecha en la que se publica el post.
 - **text:** Texto del post.
- **Player:** Almacena información inherente a los jugadores.
 - **uuid:** Identificador único de cada jugador.
 - **username:** Nombre de usuario.
 - **fullname:** Nombre real.
 - **role:** Posición que ocupa ese jugador en el equipo.
 - **value:** Valor de mercado por defecto.
- **Real Team:** Almacena información inherente a los equipos de las competiciones deportivas.

- **uuid:** Identificador único del equipo.
 - **name:** Nombre del equipo.
 - **short name:** Siglas.
 - **overviewpage:** Referencia de consulta sobre el equipo.
 - **game:** Videojuego al que compite este equipo.
- **Real League:** Almacena la información correspondiente al evento deportivo.
 - **uuid:** Identificador único de la liga.
 - **overview page:** Referencia de consulta sobre la liga.
 - **event reference:** Referencia de consulta sobre el evento general.
 - **api id:** Identificador de api y/o referencia cargo de la que se obtiene la información de esta liga.
 - **short name:** Siglas de la liga.
 - **is playoff:** Determina si una liga ha entrado en la modalidad de fase de grupos.
 - **current jour:** Determina la jornada actual en la que se encuentra una liga.
 - **Event:** Almacena información inherente a los eventos/partidos.
 - **uuid:** Identificador único del evento.
 - **team 1,2:** Referencia los equipos que se enfrentan.
 - **t1,2 score:** Resultado de cada equipo.
 - **mvp:** jugador que ha obtenido el «Most valued player» del partido.

Relaciones:

- **User in league:** Almacena los usuarios afiliados a cada liga.
 - **admin:** Determina si un usuario es administrador en determinada liga.
 - **money:** Determina el dinero que un usuario tiene en una liga.
- **Post in league:** Almacena que entrada corresponde a que liga.
- **Market:** Almacena qué jugadores hay disponibles en cada momento para compra en determinada liga:
 - **in sell:** Determina si el jugador está en venta.
 - **market value:** Precio de venta del jugador.

- **clause:** Precio de cláusula del jugador.
- **owner:** Propietario actual del jugador.
- **Team:** Almacena qué jugadores hay en que equipos de usuarios para determinadas ligas y si estos están alineados.
 - **jour:** Jornada en la que se tuvo a este jugador.
 - **aligned:** Determina si el jugador estaba alineado y en qué posición.
- **plays in:** Asocia jugadores con equipos reales.
- **plays ev:** Asocia que jugadores juegan en qué partidos.
 - **points:** Los puntos que realizaron en ese partido.
- **Corresp to:** Asocia equipos reales a determinadas ligas reales.
- **played for:** Asocia las ligas reales con sus eventos.
 - **jour:** Jornada en la que tuvo lugar este evento.
- **bidup:** Representa una puja de un usuario por un jugador.
 - **date:** Fecha en la que tuvo lugar la puja.
 - **state:** Estado de la puja (finalizada o no).
 - **bid:** Cantidad de dinero ofrecido por el usuario.
- **L-RL:** Relación entre una liga creada por los usuarios y una liga real de deporte electrónico.

El esquema físico⁷ es una representación directa de como se han almacenado definitivamente las entidades, relaciones y atributos en la base de datos habiendo hecho previamente un refinamiento del esquema E/R.

A continuación se realiza el paso de un esquema conceptual a un esquema lógico, más detallado y específico para hacer posible la implementación. En este, tal y como se puede apreciar en la figura 3.3, se estipulan restricciones de integridad referencial, cardinalidad, tipos de datos, etc.

A continuación realizaré un breve análisis en el que describiré las tablas que han sido fusionadas y comentaré todo aquello de la normalización 3FN⁸ que no sea evidente, además de responder algunas posibles preguntas que podrían surgir sobre el diseño.

⁷Código del esquema físico de la base de datos.

⁸Tercera forma normal.

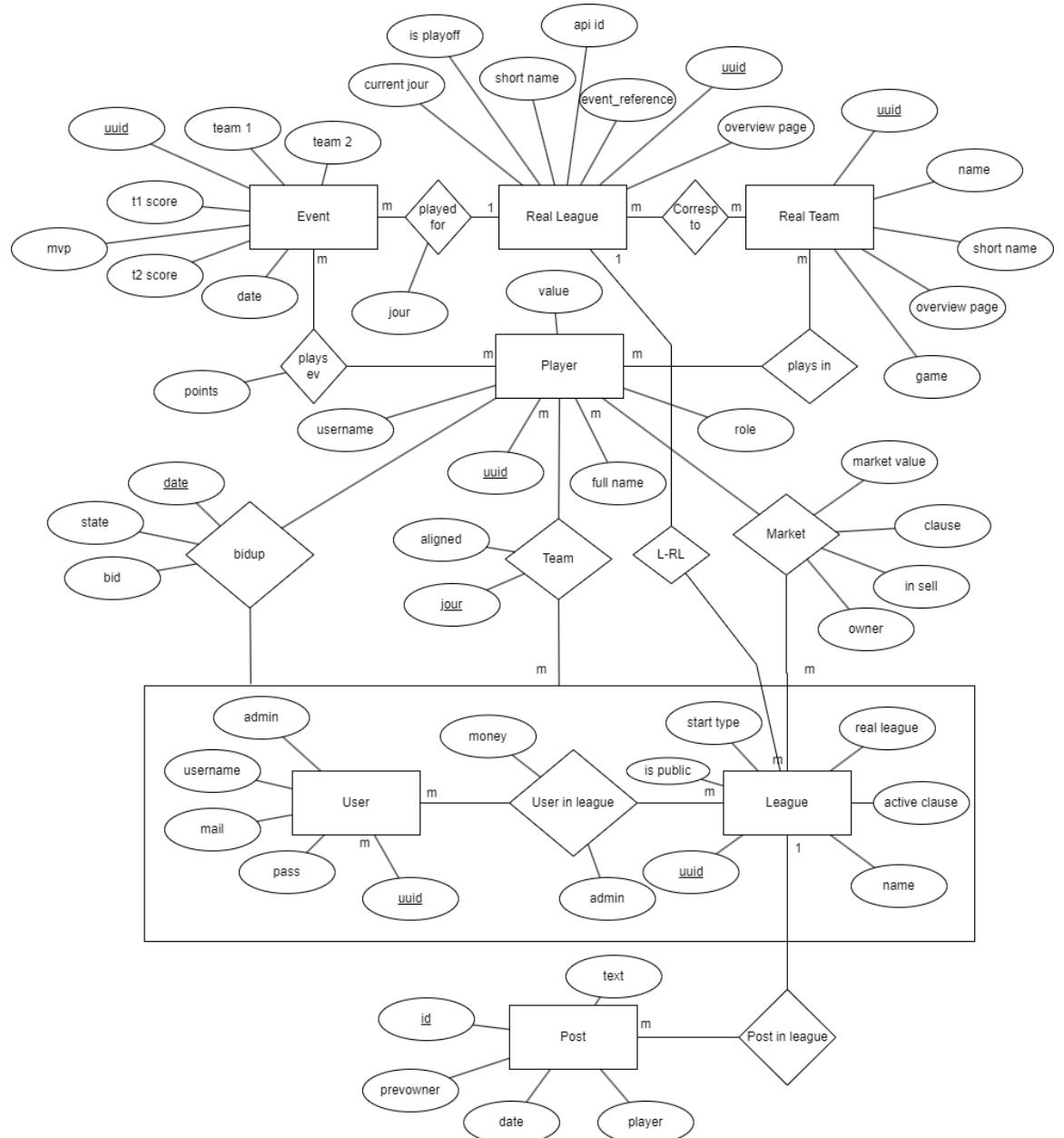


Figura 3.2: Esquema E/R conceptual.

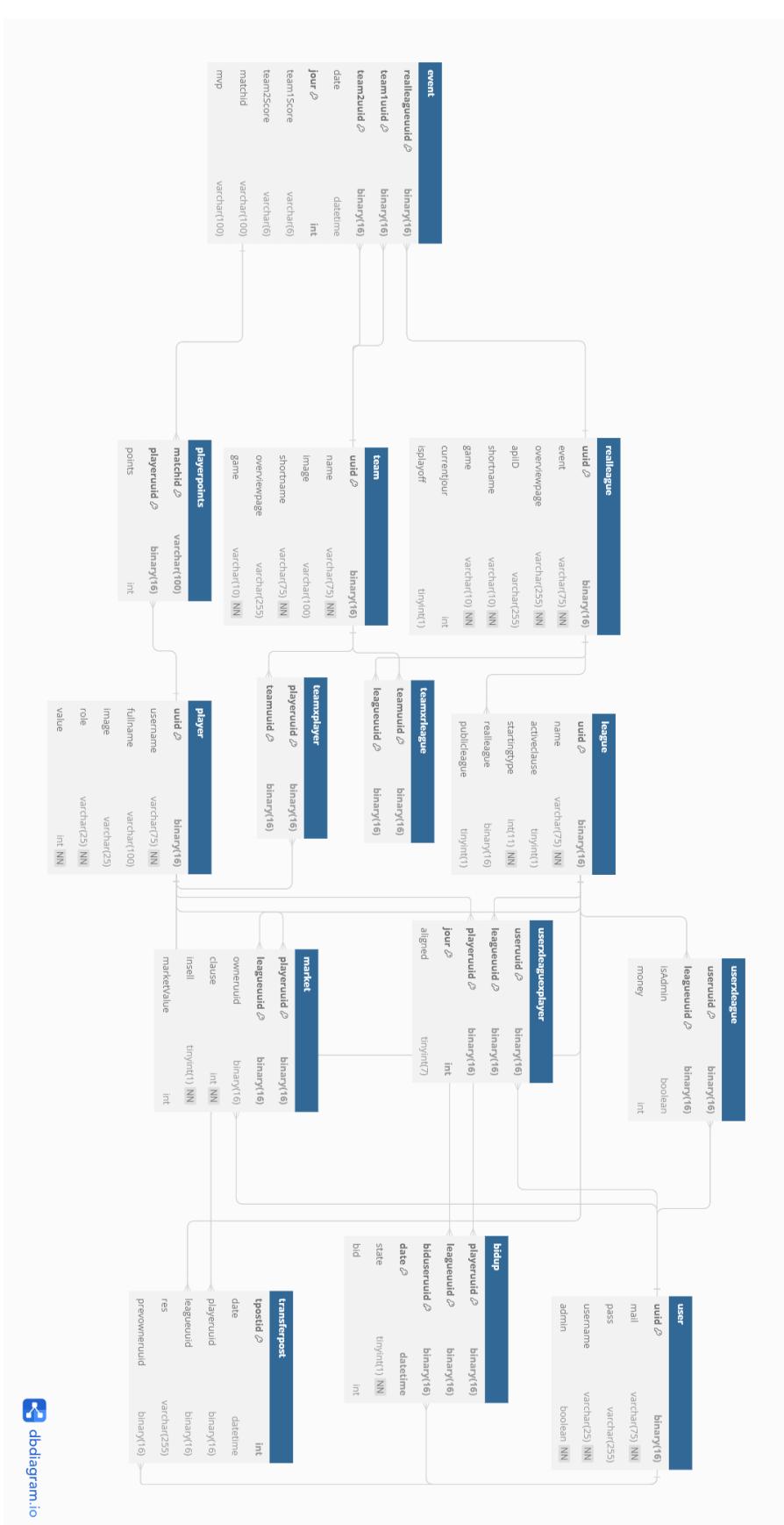


Figura 3.3: Esquema físico.

Fusiones

Fusionaremos tablas para relaciones 1:1 o relaciones 1:n que supongan una simplificación de la representación de la información.

- **Post in league** se ha fusionado dentro de **Post**, ya que el post siempre pertenecerá únicamente a una sola liga.
- **Played for** junto a su atributo jour se ha fusionado dentro de la tabla **Event**, ya que un evento solo puede suceder para una única competición de deportes electrónicos.
- **L-RL** se ha fusionado dentro de **League**, ya que una liga de usuarios tan solo corresponde a una única liga de deportes electrónicos.

Son considerados necesarios los siguientes puntos para que una base de datos esté en tercera forma normal:

- **1FN:**
 - **Atomicidad de valores:** No puede haber columnas con listas.
 - **Unicidad de filas:** Filas únicas.
- **2FN:**
 - **1FN:** Cumple la Primera forma normal.
 - **No hay dependencias parciales:** Un atributo no clave no puede depender de una sola parte de una clave primaria compuesta.
- **3FN:**
 - **2FN:** Cumple la Segunda forma normal.
 - **No hay dependencias transitivas:** Los atributos no clave deben depender exclusivamente de la clave primaria.

A continuación se realizará un desglose de tablas en el que podremos ver la normalización de las mismas:

Tabla	Clave Primaria	Atributos No Clave	Dependencias Transitivas	Cumple con 3NF
user	uuid	mail, pass username, admin	No	Sí
realleague	uuid	event, overviewpage apiID, shortname game, currentjour isplayoff	No	Sí
team	uuid	name, image shortname, overviewpage game	No	Sí
teamxrealleague	(teamuuid, leagueuuid)	Ninguno	No	Sí
league	uuid	name, activeclause startingtype, realleague publicleague	No	Sí
userxleague	(useruuid, leagueuuid)	isAdmin, money	No	Sí
player	uuid	username, fullname image, role value	No	Sí
teamxplayer	(playeruuid, teamuuid)	Ninguno	No	Sí
userxleaguexplayer	(useruuid, leagueuuid, playeruuid, jour)	aligned	No	Sí
market	(playeruuid, leagueuuid)	owneruuid, clause insell, marketValue	No	Sí
bidup	(playeruuid, leagueuuid, biduseruuid, date)	state, bid	No	Sí
event	(realleagueuuid, team1uuid, team2uuid, jour)	date, team1Score team2Score, matchid mvp	No	Sí
playerpoints	(matchid, playeruuid)	points	No	Sí
transferpost	tpostId	date, playeruuid leagueuuid, res prevowneruuid	No	Sí

Tabla 3.1: Tabla de dependencias transitivas 3FN.

Dado que todas las tablas cumplen la Tercera forma normal (ver tabla 3.1), podemos concluir que nos encontramos ante una base de datos correctamente normalizada en 3FN.

Posibles preguntas o dudas sobre diseño y normalización

¿Por qué la relación Player-Real Team no es n:1 y así se fusionaría otra tabla?

Dependiendo del deporte electrónico, hay competiciones que tienen lugar al mismo tiempo que los mundiales, por lo que podría suceder que un jugador esté asociado a dos equipos diferentes al mismo tiempo.

¿No es game una dependencia transitiva en la tabla team? No. Dado que hay modalidades de ligas que son la misma pero compiten en diferentes videojuegos, no se puede asociar game como atributo no clave de Real League, sin embargo, los equipos que compiten en la misma liga, pero en diferentes juegos si que son diferentes a pesar de tener mismo nombre y logo, pues cuentan con otros jugadores. De esta manera, podemos considerar game como un atributo no clave y sin dependencia transitiva con respecto al equipo.

¿No hay redundancia de datos con respecto al campo value y market value en las tablas Player y Market respectivamente?

No. el campo value de la tabla Player representa el valor base de mercado que ese jugador tiene y es inherente a él. El valor Market value representa el precio de venta del jugador en el mercado de una liga. Si bien es cierto que mientras ese jugador no haya sido puesto en venta por un usuario, el valor de value y market value será el mismo, una vez esto suceda el valor de market value pasará a ser la cantidad que el usuario pide por ese jugador.

¿Por qué Market está asociado a League directamente en lugar de a la agregación User-League si también cuenta con un campo owner perteneciente a la tabla User?

Porque el campo owner no es parte de la clave primaria. No necesariamente el jugador tiene porque estar asociado a un usuario, ya que este podría, por ejemplo, no haber sido comprado por nadie aún.

¿Cuál es la diferencia entre el campo admin de User y el campo admin de User in league? El primero hace referencia a si un usuario es administrador general del sitio web, mientras que el segundo hace referencia a si el usuario es administrador de la liga en la que se encuentra.

¿No es peligroso crear ciclos en los diseños de la base de datos como el que se aprecia en la parte superior de la figura 3.2? No, la creencia de que los ciclos en las bases de datos son peligrosos es un mito muy extendido entre las personas que se dedican al medio. Siempre y cuando se diseñe una correcta integridad referencial, no hay ningún problema con ellos. Aún así, no se genera ningún ciclo en la parte superior del esquema, ya que la tabla played for se fusiona dentro de Event.

3.3.4. HT 3: Diseño de arquitectura y configuración del entorno Backend.

Para la realización del apartado Backend de la aplicación utilizaremos Java con el Framework SpringBoot. Y crearé la inicialización mediante el IDE ItelliJ IDEA.

Las dependencias con las que se inicializará el proyecto son las siguientes:

- **spring-boot-starter-data-jpa:** Para el manejo de bases de datos relacionales.
- **mysql-connector-java:** Para la conexión a base de datos.
- **spring-boot-starter-web:** Para los mapeos de direcciones.
- **spring-session-core:** Para la gestión de sesiones de usuario.

Arquitectura inicial

Utilizaremos una arquitectura por capas en el backend, al ser esta la arquitectura más típica en este tipo de sistemas y la recomendada por la documentación oficial de Springboot [bae24], gracias a sus características principales:

- Separación de responsabilidades.
- Productividad y facilidad de desarrollo.
- Escalabilidad.
- seguridad.

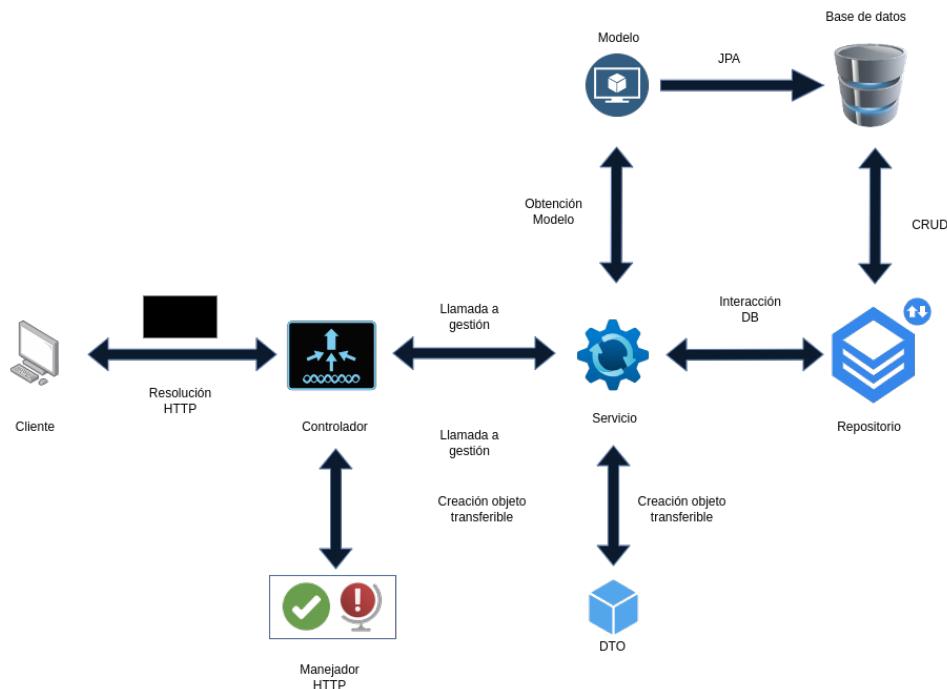


Figura 3.4: Arquitectura por capas backend [Elaboración propia].

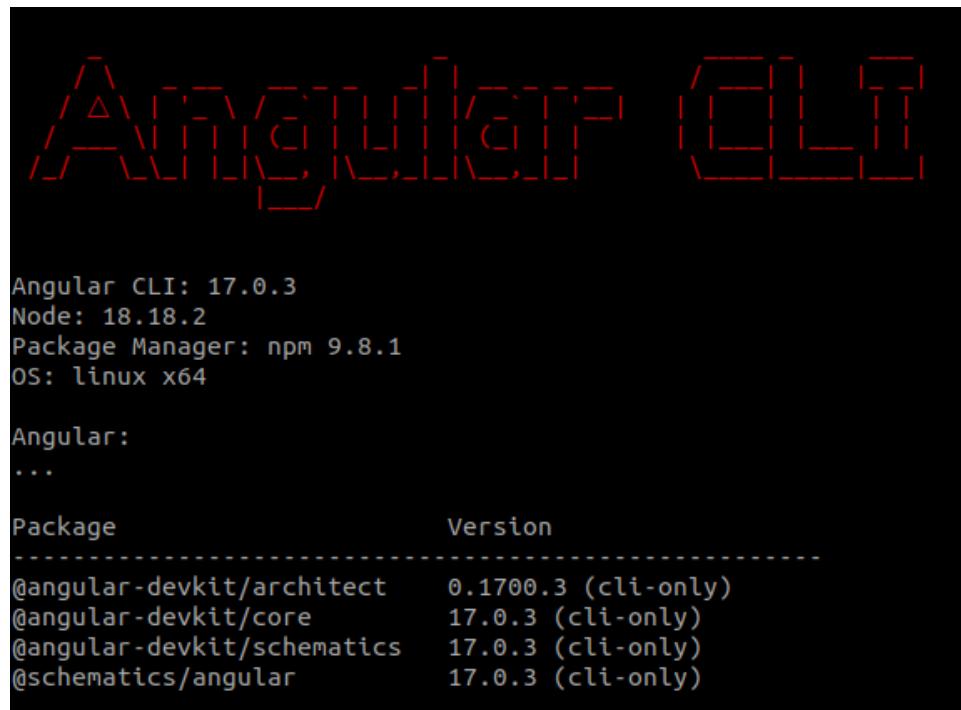
La división de la arquitectura quedará dividida en 5 capas principales, las cuales se retro-alimentarán unas a otras, ver figura 3.4.

- **Capa de entidad:** Donde asocia clases con las tablas de la base de datos.
- **Capa de controlador:** Gestión de las solicitudes HTTP, actúa como interfaz entre el frontend y el resto de la aplicación.

- **Capa de servicio:** Divide la lógica de negocio en servicios, realizando cada uno en una tarea específica.
- **Capa DTO:** Destinada a la transferencia y conversión de objetos en peticiones web a objetos.
- **Capa de repositorio:** Interfaz de JPA⁹ del acceso CRUD a la base de datos.

3.3.5. HT 4: Diseño de arquitectura y configuración del entorno Frontend.

Para la realización del apartado Frontend de la aplicación utilizaremos Angular (ver figura 3.5), el cual es un Framework de JavaScript que además da opción a crear y gestionar servidores mediante NodeJS, por lo que partiremos de la instalación de las bibliotecas NodeJS correspondientes. Para ello utilizaremos el gestor NVM¹⁰, el cual nos permite navegar a través de las versiones de node con mayor facilidad.



```

Angular CLI: 17.0.3
Node: 18.18.2
Package Manager: npm 9.8.1
OS: linux x64

Angular:
...
Package          Version
-----
@angular-devkit/architect    0.1700.3 (cli-only)
@angular-devkit/core         17.0.3 (cli-only)
@angular-devkit/schematics   17.0.3 (cli-only)
@schematics/angular          17.0.3 (cli-only)

```

Figura 3.5: Versión de Angular CLI.

⁹Java Persistence API.

¹⁰Node Version Manager.

Arquitectura inicial

A pesar de que no se vaya a usar Nx-Angular¹¹ en la aplicación, vamos a seguir su arquitectura típica, la cual aporta escalabilidad y facilidad en la comprensión. Esta arquitectura es una forma de organización que trata de dividir el código en los apartados más característicos de la aplicación, aportando a cada uno de estos módulos la mayor cantidad de características propias posibles.

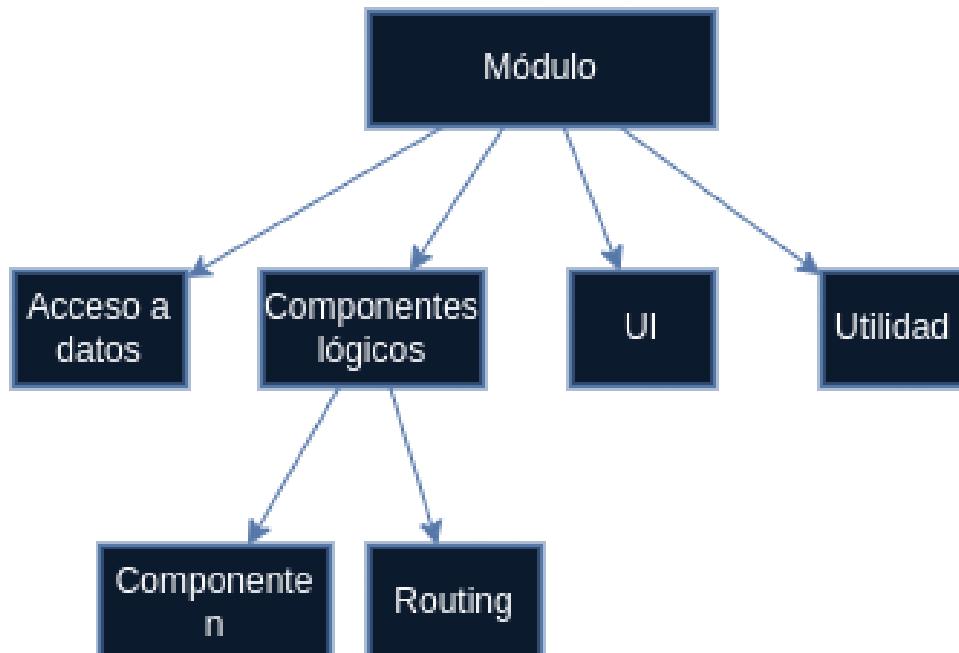


Figura 3.6: Estructurado de componentes en mi arquitectura Angular. [E elaboración propia].

Los módulos son los siguientes:

- **Acceso a datos:** para la gestión de llamadas a servicios.
- **Componentes lógicos:** para la implementación de las pantallas dinámicas. Además aquí también albergaremos el routing de cada componente dentro de este subsistema.
- **UI:** Para la gestión de «dumb components», que simplemente mostrarán información al usuario. Este es omitible si el módulo que se está implementando no lo requiere.
- **Utils:** para la gestión de funcionalidad común en los múltiples componentes.

¹¹Kit de herramientas para desarrollo monorepo [rsa20].

Configuración de diseño y paleta de colores

Se introdujo en el proyecto Angular Material¹². Esto es una biblioteca de componentes de UX/UI¹³ basada sobre «Material design¹⁴» y se definió una paleta de colores y pautas de interfaz sobre la que se estructuraría gráficamente el proyecto más adelante, poniendo el color corporativo como piedra angular. También se definió una hoja de estilos principal¹⁵, en la que se introdujeron algunas clases predefinidas para algunos elementos.

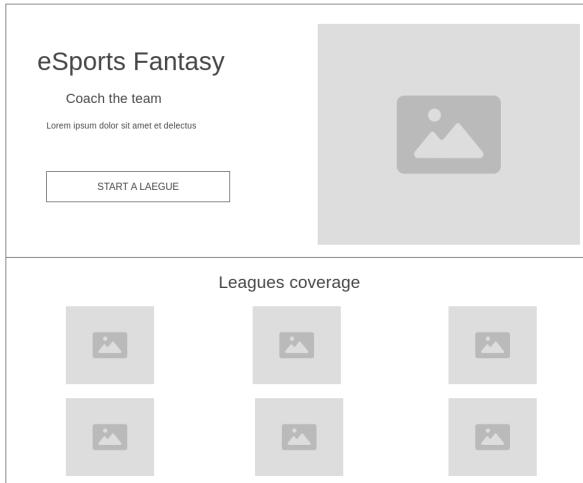


Figura 3.7: Wireframe pantalla de bienvenida

Una parte muy importante de un desarrollo web es la pantalla de bienvenida, la cual debe ser atractiva para poder atraer a los usuarios y representativa para que comprendan que es lo que hay en la web una vez que se registren como usuarios. Para ello he decidido dividir la pantalla en dos secciones, una con un carrusel de imágenes, que aporte dinamismos a la web, junto con un botón que de fácil acceso a la pantalla de registro. Mientras que en la parte inferior colocaré un escaparate con algunas de las ligas más importantes o populares y que por consiguiente puedan llamar más la atención de los usuarios. Para cada unas las pantallas que se quieran hacer, se

¹²Código de tema personalizado.

¹³User experience / User interface.

¹⁴Conjunto de pautas y buenas prácticas estipuladas por Google para obtener buenas interfaces de usuario.

¹⁵Código hoja de estilos principal.

realizará antes un diseño wireframe¹⁶, el cual me ayudará a pensar la estructura del maquetado y finalmente se realizará el diseño web. Se seguirá este esquema de wireframe seguido de pantalla durante el resto del TFG. Ver figura 3.7 y 3.8.

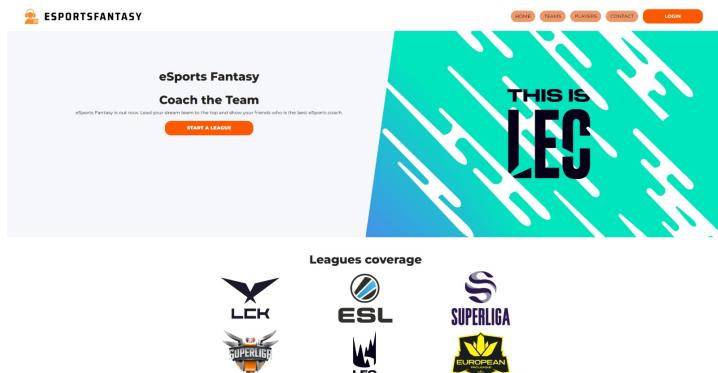


Figura 3.8: Pantalla de bienvenida.

3.3.6. Retrospectiva y retrasos

La dos principales dificultades encontradas en este sprint han sido el aprendizaje y montaje que la arquitectura de redes de una tecnología la cual era desconocida para mi como es Docker y el correcto diseño de la base de datos, la cual pasó por hasta 8 diseños anteriores antes de obtener el definitivo que podemos observar en la figura 3.2.

A pesar de las dificultades encontradas, en este sprint no tuvieron lugar retrasos con respecto a la programación estipulada.

3.4. Sprint 2

3.4.1. Introducción al Sprint 2

A lo largo del segundo sprint de desarrollo se programará todo lo relacionado con el manejo de usuarios, desde el registro pasando por el inicio de sesión, la gestión de credenciales JWT hasta la configuración.

En la figura 3.9 podemos ver un diagrama UML que representa las principales clases que tienen acción a lo largo de este sprint. Con él puede tomar una idea general de la estructura del sprint, a continuación, desglosaremos este diagrama mediante referencias al mismo en las explicaciones de los siguientes apartados.

¹⁶Representación visual básica de como se verá la pantalla.

3.4.2. Autenticación JWT

Para garantizar la seguridad de la aplicación, he decidido incluir un sistema de autenticación mediante tokens JWT. Un Json Web Token o JWT, es un mecanismo de propagación de información de forma segura el cual comúnmente utilizamos para propagar la información de la identidad del usuario. Este token va firmado mediante un «secreto de aplicación», de modo que cualquier tipo de modificación que sufra supondrá de manera inmediata la invalidación del token [Bel23].

En la figura 3.9 podemos ver las clases «JwtService»¹⁷ y «JwtAuthenticationFilter»¹⁸. En la primera, declaramos datos relevantes como la clave secreta del lado del servidor que es necesario estipular a la hora de crear un token JWT y el tiempo que tarda el token en expirar, el cual se ha estipulado en 24 días. Además se implementan los métodos correspondientes a la generación, la autenticación, la firma y la comprobación de si se trata de un token válido o expirado. Esta clase se incluirá en «JwtAuthenticationFilter», ya que es la utilizada por la biblioteca de seguridad web de Springboot y en ella sobrescribiremos el método «doInternalFilter» para que siga la lógica de verificación creada en el servicio de JWT descrito anteriormente.

De esta manera, cuando un usuario inicie sesión, se le devolverá automáticamente un token JWT asociado al mismo el cual almacenará en el almacenamiento local de su navegador. Cuando el usuario realice peticiones al backend que requieran uso de credenciales, se incluirá este token JWT como parte del header de la petición, para poder verificar en el backend la identidad del usuario.

Cabe destacar que por motivos de seguridad, las claves secretas declaradas en los archivos de configuración publicados en github no son las utilizadas realmente en la aplicación.

3.4.3. HU 1.1: Registro en la aplicación (correo)

En la figura 3.9 podemos observar la clase «User»¹⁹. Este es el principal modelo de datos de este sprint, el cual se encargará de representar la información correspondiente a los usuarios de la aplicación como nombre, email, identificador, contraseña, etc. También podemos observar «UserService»²⁰, este se encargará de gestionar toda la lógica de negocio correspondiente a los usuarios, por lo que podemos observar funciones de características «CRUD»,

¹⁷Código para JwtService.

¹⁸Código para JwtAuthenticationFilter.

¹⁹Código para el modelo de usuario.

²⁰Código para el servicio de usuario.

para la obtención de información del usuario, registro, inicio de sesión, etc.

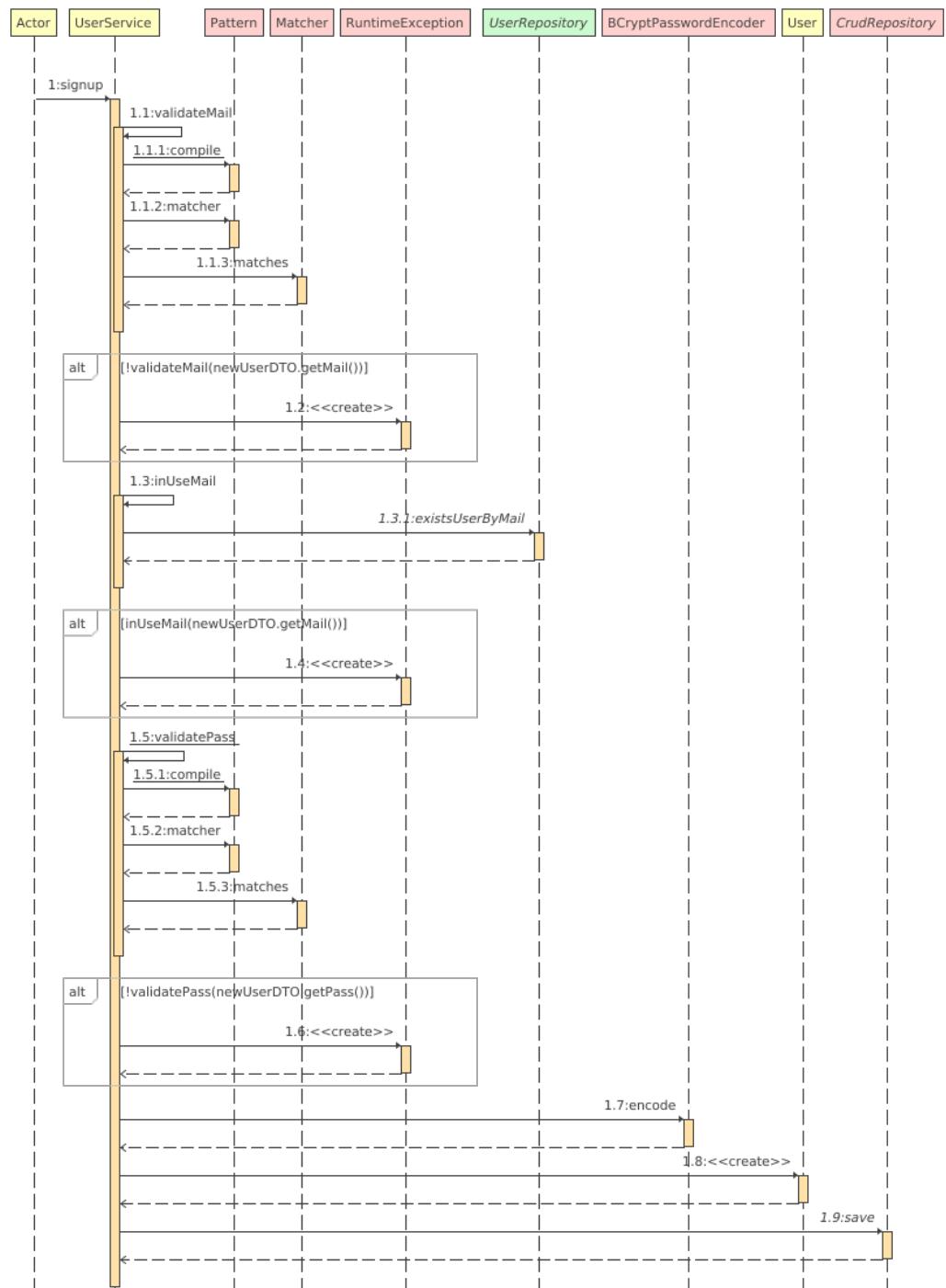


Figura 3.10: Diagrama de secuencia registro.

Podemos observar el flujo de actuación del registro del lado del servidor en la figura 3.10. En primer lugar se harán ciertas comprobaciones como validación del correcto formato de email (flujo 1.1), la posible existencia de email ya registrado (flujo 1.3) o la validación de formato de la contraseña (flujo 1.5), la cual además realiza su encriptamiento mediante el sistema de «BCrypt». Me he decantado por este algoritmo de encriptación por ser el recomendado por Springboot, el cual ofrece facilidades en su uso en el entorno de desarrollo en el que me encuentro y además es conocido por su resistencia a ataques de fuerza bruta. Finalmente, en caso de incumplir alguno de los requisitos, se lanzará una excepción correspondiente, mientras que si todo ha ido de manera exitosa, se creará y persistirá el nuevo usuario en la base de datos mediante el repositorio²¹.

Para el apartado del frontend, me he decantado por un formulario simple y minimalista (ver figuras 3.11 y 3.12), requiriéndole al usuario la información mínima indispensable para poder registrarse en la aplicación. De esta forma es fácil e intuitivo para el usuario crear un nuevo perfil en la aplicación, por lo que obtendremos usuarios con mayor facilidad. Como seguridad adicional, también se incluyen algunos algoritmos de validación de campos antes de enviar la petición mediante, como validación de email y contraseñas.

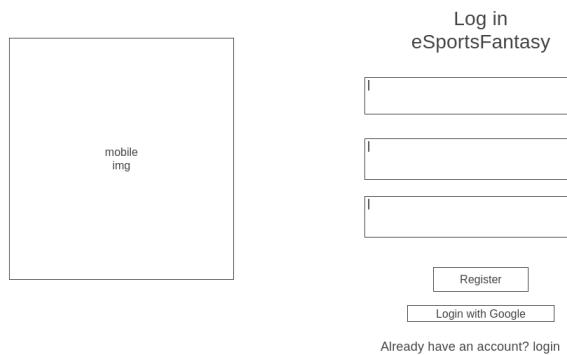


Figura 3.11: Wireframe pantalla de registro

²¹Código del repositorio de usuario.

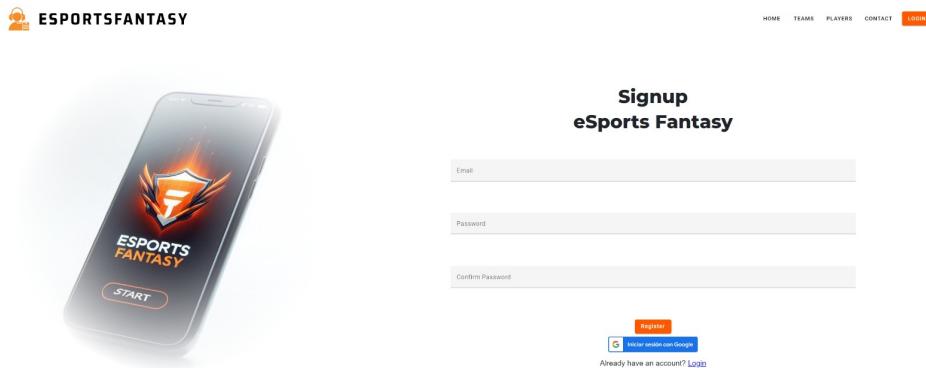


Figura 3.12: Registro.

3.4.4. HU 1.2.1 y 1.2.2: Inicio de sesión (Correo y Google)

Tanto si se trata de un inicio de sesión mediante correo electrónico como si es mediante Google, el proceso será casi el mismo. Podemos seguir el flujo a través de las figuras 3.13 y 3.14. El usuario introducirá las credenciales, a las cuales se le harán las comprobaciones pertinentes como comprobación de existencia de usuario y verificación de coincidencia en la contraseña (flujos 1.1 al 1.8). Una vez verificada la veracidad de los mismos, el servidor generará un token JWT el cual mandará al cliente. Este guardará en la memoria local del navegador, lo que facilita que el usuario pueda cerrar la pestaña y volver a entrar sin tener que volver a introducir los credenciales de nuevo la siguiente vez hasta que cierre su sesión o el token expire.

Para el inicio de sesión de Google, utilizaremos las bibliotecas que la compañía nos ofrece. En frontend realizaremos una solicitud a los servicios de Google los cuales, con la confirmación del usuario, nos dará un token JWT. Sin embargo, por cuestiones de seguridad este no es el token que utilizaremos en el backend. Confirmaremos la veracidad del token que se nos ha mandado mediante la biblioteca que nos ofrece Google para ello [Goo24a] (flujo 1.1) y una vez hecho, si este usuario no está registrado aún en base de datos, almacenaremos los datos del usuario en nuestra base de datos (flujo 1.3 al 1.11) pero sin contraseña. Esto nos permite unificar los usuarios que vienen de Google con los que vienen de forma convencional. Finalmente, por cuestiones de seguridad y privacidad, desecharemos el token JWT que nos ofrece Google y procederemos a generar un token propio.

Con respecto al apartado frontend, puede apreciarse en la figuras 3.15 y 3.16 como me he decantado por un diseño muy similar al del registro, de nuevo por las mismas razones antes descritas como son simplicidad y limpieza y eficiencia, con la diferencia de no tener el campo de confirmación de contraseña y añadir el botón de inicio de sesión con Google.

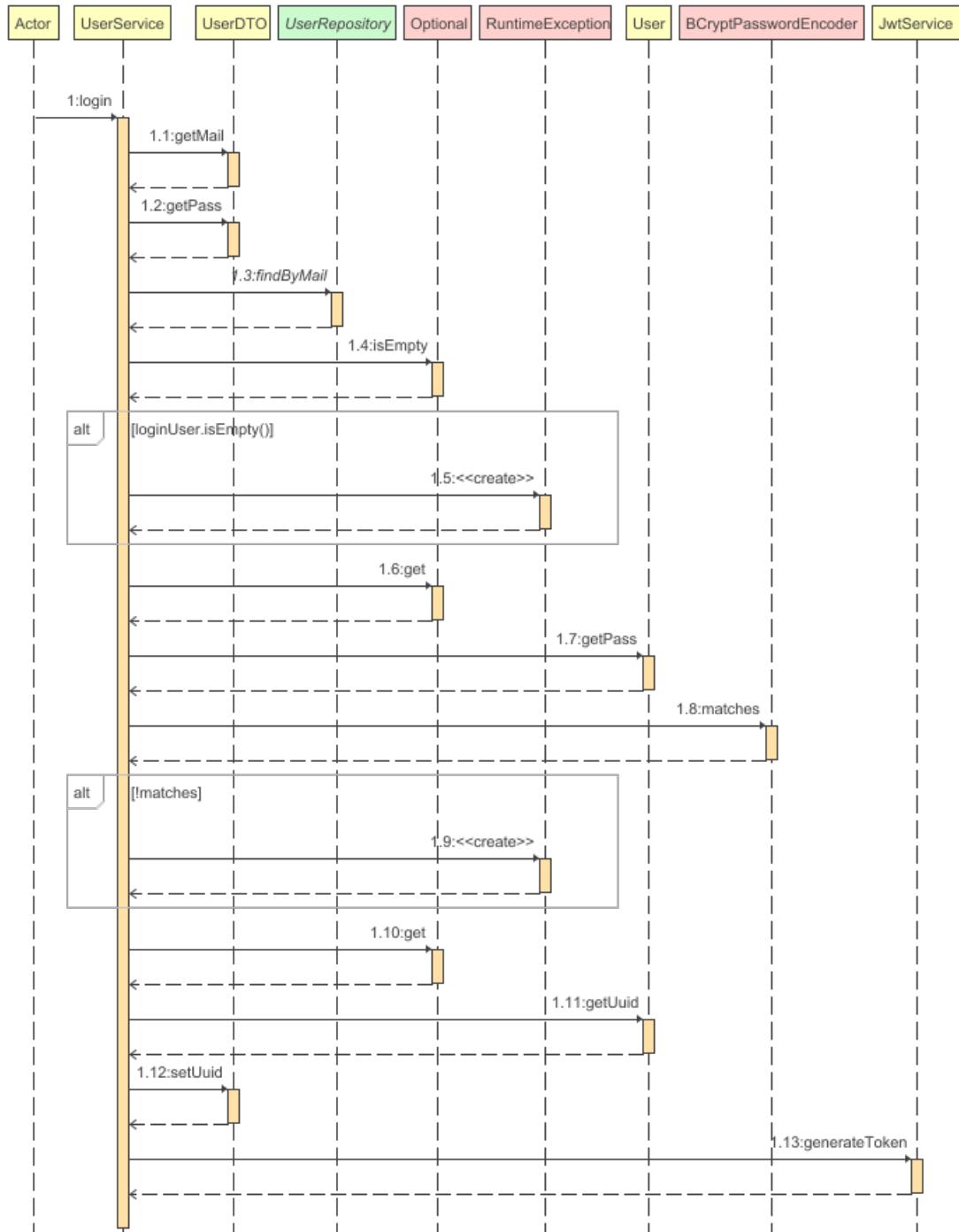


Figura 3.13: Diagrama de secuencia inicio de sesión.

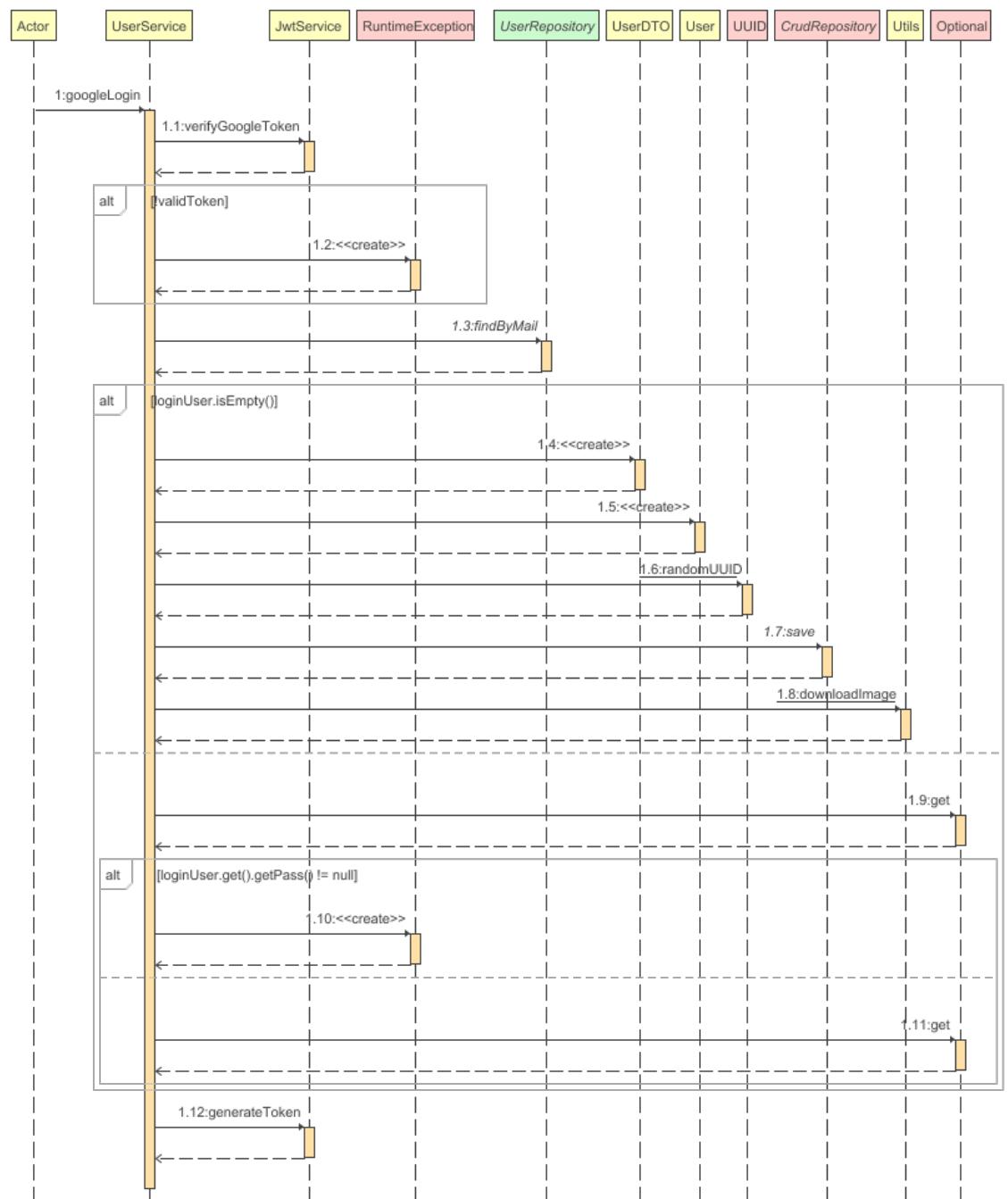


Figura 3.14: Diagrama de secuencia inicio de sesión mediante Google.

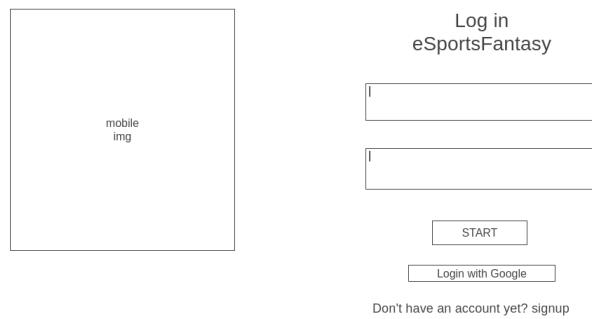


Figura 3.15: Wireframe pantalla de inicio de sesión.

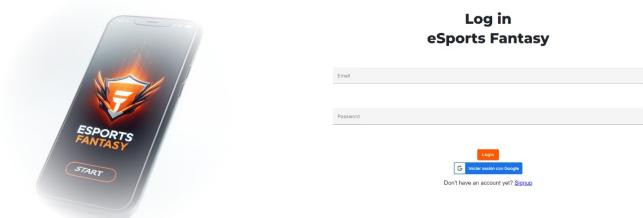


Figura 3.16: Inicio de sesión.

3.4.5. HU 1.3: Configuración de cuenta

Para el apartado de configuración de cuenta se han creado múltiples «endpoints» para el servicio de usuarios. Estos aplicarán modificaciones sobre el modelo de datos correspondiente, permitiéndote hacer algunas configuraciones en tu cuenta como cambio de foto de perfil, nombre de usuario o correo electrónico. De nuevo, al igual que realizábamos en el apartado de registro, se realizan exactamente los mismos flujos de verificación para cada uno de los campos que hemos descrito en los puntos anteriores, tanto a nivel de cliente como a nivel de servidor, con el objetivo de evitar brechas de seguridad a la hora de modificar el modelo de datos del usuario.

The wireframe shows a user interface for editing user information. At the top, there's a placeholder for a profile picture with a camera icon. Below it, there are fields for 'username' and 'mail'. A button labeled 'Edit User Information' is positioned above a group of input fields. These fields include 'Email' (with a placeholder 'Email'), 'Username' (placeholder 'Username'), 'Icon' (placeholder 'Icon'), and 'Password' (placeholder 'Password'). To the right of these input fields is a 'save changes' button. At the bottom of the form are two buttons: 'edit' and 'Logout'.

Figura 3.17: Wireframe configuración de usuario.

Siguiendo la estela de minimalismo y manteniendo la estética y colores de la aplicación, se ha creado un escaparate de información, ocultando el formulario de modificación (ver figuras 3.17 y 3.18). Fácilmente puede desplegarse este formulario mediante el botón de edición. Al aplicar los cambios se envían todas las peticiones «backend» necesarias, para comprobar que los cambios hayan surtido efecto correctamente.

User Information



RedRiot 

admin@gmail.com

Edit User Information

Email:

Username:

Icon:

Ningún archivo seleccionado

Password:

Save Changes

Edit **Logout**

Figura 3.18: Configuración de usuario.

3.4.6. Tests unitarios y de integración

Para este Sprint se han realizado un total de 18 tests unitarios²² y 6 tests de integración.

²²Código para los tests de usuario.

Desglose tests unitarios

Test	HU	Descripción	Código de error	Resultado
testValidateMailValidEmail	1.1 - 1.2.1 - 1.2.2	Validación email.	-	✓
testValidateMailInvalidEmail	1.1 - 1.2.1 - 1.2.2	Validación email incorrecto.	-	✓
testInUseMail	1.1 - 1.3	Email en uso.	-	✓
testNonInUseMail	1.1	Email no en uso.	-	✓
testValidatePassInvalidPass	1.1 - 1.2.1 - 1.2.2	Contraseña inválida.	-	✓
testValidatePassValidPass	1.1 - 1.3	Contraseña válida.	-	✓
testValidSignup	1.1	Registro correcto.	-	✓
testInvalidSignupMail	1.1	Registro incorrecto.	1005	✓
testInvalidSignupMailInUse	1.1	Registro incorrecto (email en uso).	1006	✓
testInvalidSignupPass1	1.1	Registro contraseña sin números.	1007	✓
testInvalidSignupPass2	1.1	Registro contraseña sin minus.	1007	✓
testInvalidSignupPass3	1.1	Registro contraseña sin mayus.	1007	✓
testValidLogin	1.2.1	Login válido.	-	✓
testNotFoundMailLogin	1.2.1 - 1.2.2	Login inválido (no email).	1001	✓
testInvalidPassLogin	1.2.1	Login contraseña equivocada.	1002	✓
testValidLoginWithToken	1.2.1 - 1.2.2	Login con token.	-	✓
testGoogleLoginUnvalidGoogleToken	1.2.2	Login token inválido.	1003	✓
testChangeUser	1.3	Modificación usuario.	-	✓

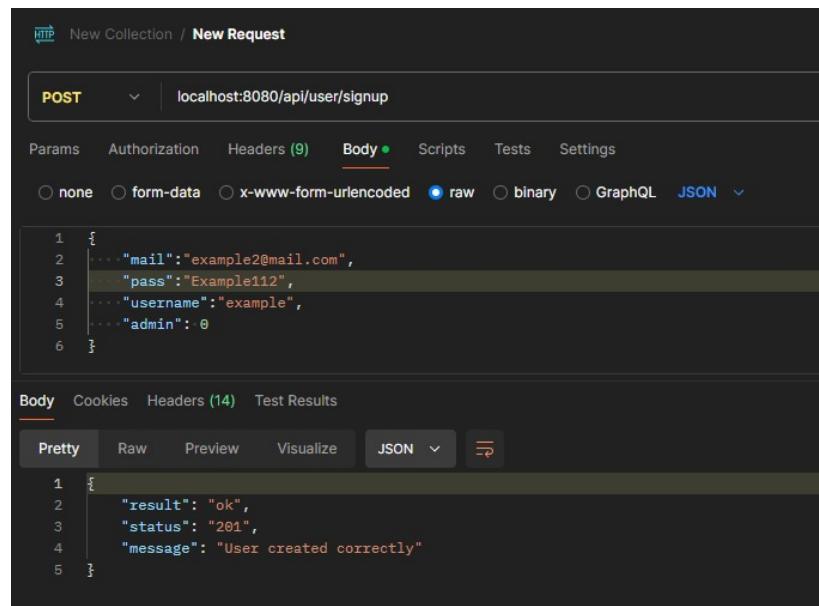
Tabla 3.2: Listado de tests unitarios.

Descripción de tests unitarios

- **testValidateMailValidEmail:** Verifica que un correo electrónico con formato válido pase la validación correctamente.
- **testValidateMailInvalidEmail:** Comprueba que un correo electrónico con formato inválido sea rechazado durante la validación.
- **testInUseMail:** Valida que el sistema identifique cuando un correo electrónico ya está en uso.
- **testNonInUseMail:** Verifica que un correo electrónico no registrado sea marcado como disponible.
- **testValidatePassInvalidPass:** Comprueba que una contraseña inválida (no cumple con los requisitos) sea rechazada.
- **testValidatePassValidPass:** Verifica que una contraseña válida (que cumple con los requisitos) pase la validación.
- **testValidSignup:** Simula un registro exitoso y comprueba que todos los datos válidos sean aceptados.
- **testInvalidSignupMail:** Simula un registro fallido debido a un correo inválido y asegura que se devuelva el error correspondiente.

- **testInvalidSignupMailInUse:** Valida que el sistema rechace un registro si el correo ya está en uso.
- **testInvalidSignupPass1:** Verifica que el registro falle si la contraseña no contiene números.
- **testInvalidSignupPass2:** Comprueba que el registro falle si la contraseña no contiene letras minúsculas.
- **testInvalidSignupPass3:** Asegura que el registro falle si la contraseña no contiene letras mayúsculas.
- **testValidLogin:** Simula un inicio de sesión válido con credenciales correctas y asegura que el login sea exitoso.
- **testNotFoundMailLogin:** Comprueba que el sistema rechace el inicio de sesión si el correo electrónico no está registrado.
- **testInvalidPassLogin:** Valida que el sistema rechace el inicio de sesión si la contraseña es incorrecta.
- **testValidLoginWithToken:** Simula un inicio de sesión exitoso utilizando un token de autenticación válido.
- **testGoogleLoginUnvalidGoogleToken:** Verifica que el sistema rechace el inicio de sesión si el token de Google es inválido.
- **testChangeUser:** Simula la modificación de un usuario existente y verifica que se realice correctamente si los datos son válidos.

Desglose tests integración



The screenshot shows a Postman collection named "New Collection". A new request is being created for the endpoint "localhost:8080/api/user/signup" using the "POST" method. The "Body" tab is selected, showing a raw JSON payload:

```

1 {
2   "mail": "example2@mail.com",
3   "pass": "Example112",
4   "username": "example",
5   "admin": 0
6 }

```

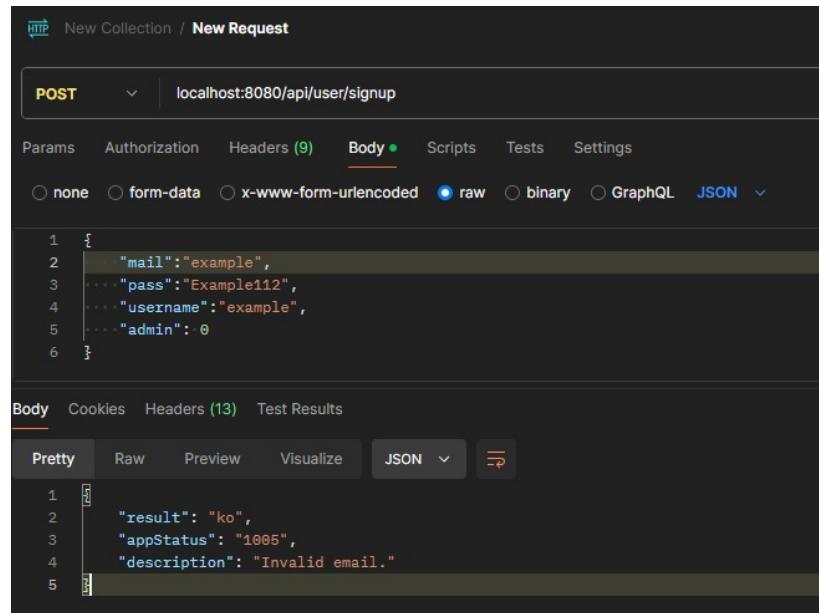
The response body is displayed in the "Pretty" tab:

```

1 {
2   "result": "ok",
3   "status": "201",
4   "message": "User created correctly"
5 }

```

Figura 3.19: Petición a endpoint /signup con credenciales válidos.



The screenshot shows a Postman collection named "New Collection". A new request is being created for the endpoint "localhost:8080/api/user/signup" using the "POST" method. The "Body" tab is selected, showing a raw JSON payload:

```

1 {
2   "mail": "example",
3   "pass": "Example112",
4   "username": "example",
5   "admin": 0
6 }

```

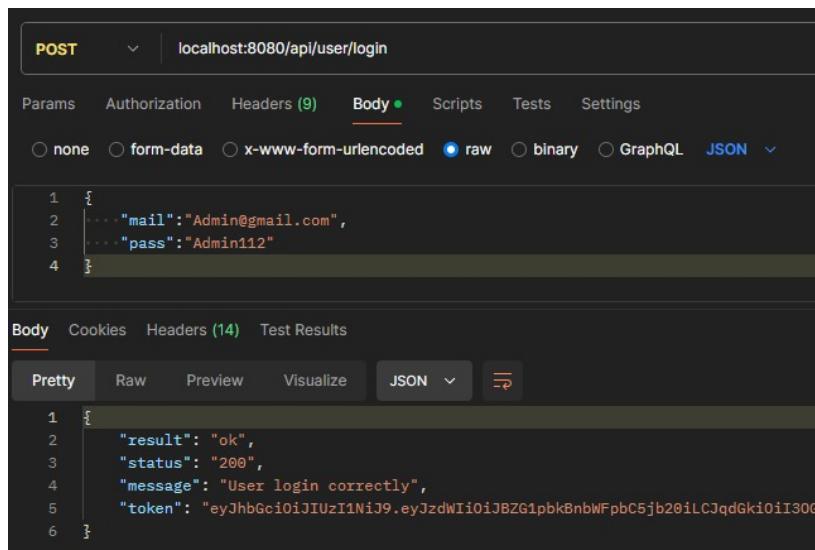
The response body is displayed in the "Pretty" tab:

```

1 {
2   "result": "ko",
3   "appStatus": "1005",
4   "description": "Invalid email."
5 }

```

Figura 3.20: Petición a endpoint /signup con credenciales inválidos.

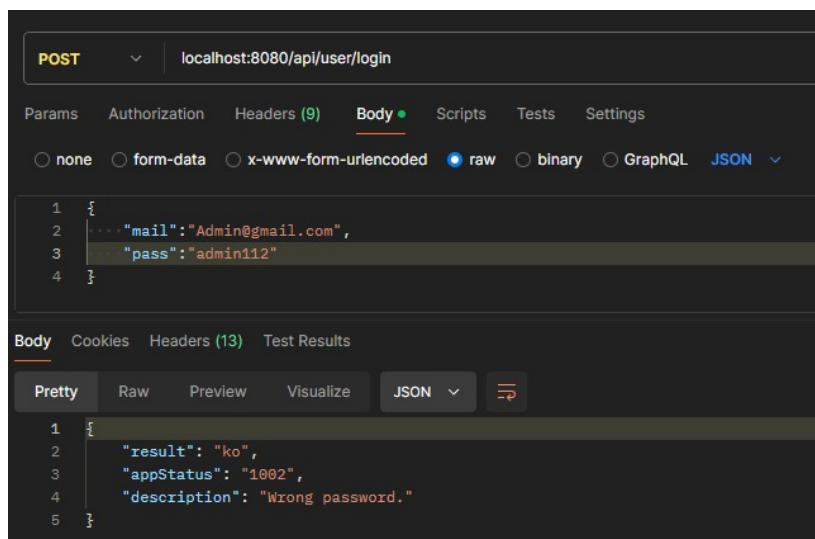


The screenshot shows a Postman request to the endpoint `localhost:8080/api/user/login`. The request method is POST. The body is set to raw JSON with the following content:

```
1 {
2   ...
3   "mail": "Admin@gmail.com",
4   ...
5   "pass": "Admin112"
6 }
```

The response status is 200, and the message is "User login correctly". A token is also provided.

Figura 3.21: Petición a endpoint /login con credenciales válidos.



The screenshot shows a Postman request to the endpoint `localhost:8080/api/user/login`. The request method is POST. The body is set to raw JSON with the following content:

```
1 {
2   ...
3   "mail": "Admin@gmail.com",
4   ...
5   "pass": "admin112"
6 }
```

The response status is 401, and the message is "Wrong password."

Figura 3.22: Petición a endpoint /login con credenciales inválidos.

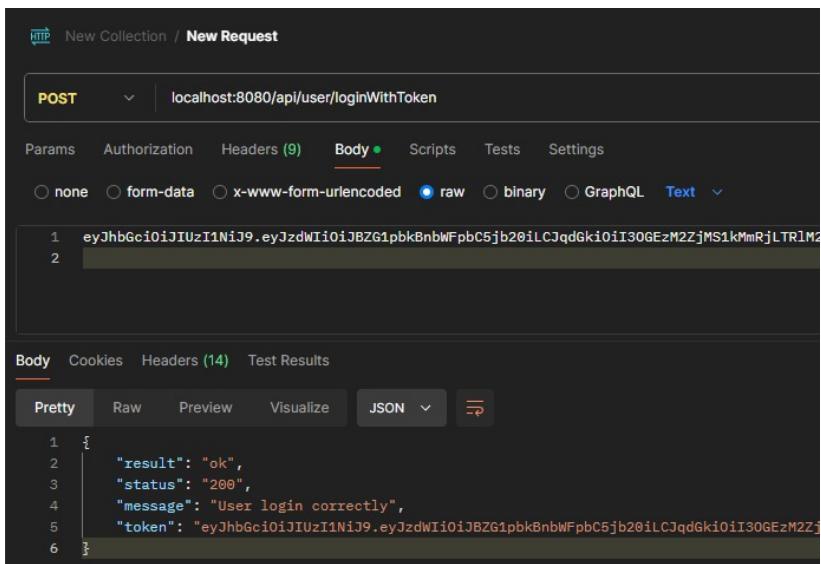


Figura 3.23: Petición a endpoint /loginwithtoken con credenciales válidos.

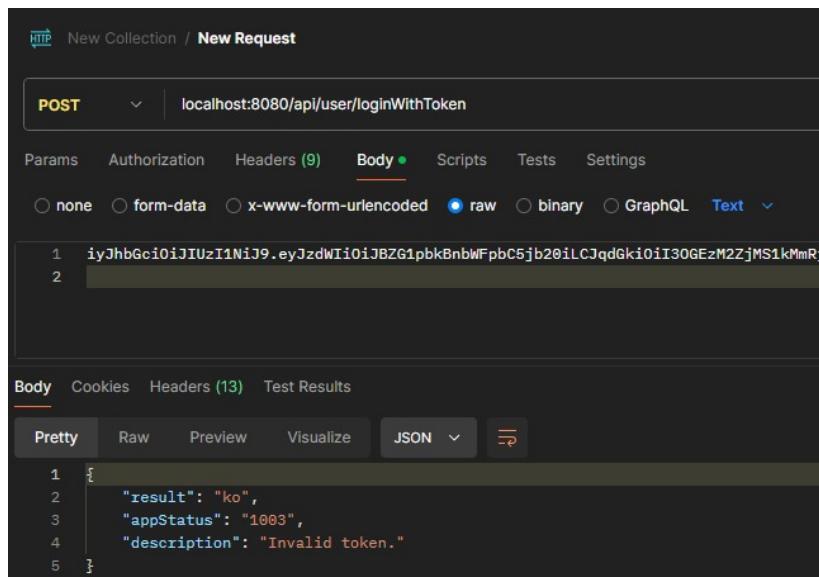


Figura 3.24: Petición a endpoint /loginwithtoken con credenciales inválidos.

Equivalencias HU/HT a pruebas de aceptación

- **HU 1.1:** Se deben verificar los casos de registro exitoso, error por correo ya registrado, y error por formato incorrecto de correo.

- **TU:** testValidateMailValidEmail.
- **TU:** testValidateMailInvalidEmail.

- **TU:** testInUseMail.
 - **TU:** testNonInUseMail.
 - **TU:** testValidatePassInvalidPass.
 - **TU:** testValidatePassValidPass.
 - **TU:** testValidSignup.
 - **TU:** testInvalidSignupMail.
 - **TU:** testInvalidSignupMailInUs.
 - **TU:** testInvalidSignupPass1.
 - **TU:** testInvalidSignupPass2.
 - **TU:** testInvalidSignupPass3.
 - **TI:** Petición credenciales válidos. Ver 3.19.
 - **TI:** Petición credenciales inválidos. Ver 3.20.
- **HU 1.2.1:** Se deben verificar los casos de inicio de sesión exitosa y error por correo no encontrado.
 - **TU:** testValidateMailValidEmail.
 - **TU:** testValidateMailInvalidEmail
 - **TU:** testValidatePassInvalidPass.
 - **TU:** testNotFoundMailLogin.
 - **TU:** testValidLogin.
 - **TU:** testInvalidPassLogin.
 - **TU:** testValidLoginWithToken.
 - **TI:** Petición credenciales válidos. Ver 3.21.
 - **TI:** Petición credenciales inválidos. Ver 3.22.
 - **HU 1.2.2:** Se deben verificar el correcto token JWT otorgado por Google.
 - **TU:** testValidateMailValidEmail.
 - **TU:** testValidateMailInvalidEmail.
 - **TU:** testValidatePassInvalidPass.
 - **TU:** testValidLoginWithToken.
 - **TI:** Petición credenciales válidos. Ver 3.23.
 - **TI:** Petición credenciales inválidos. Ver 3.24.
 - **HU 1.3:** Se deben verificar que al cambiar la configuración, el correo no esté ya en uso y la contraseña sea correcta.
 - **TU:** testInUseMail.
 - **TU:** testValidatePass.

3.4.7. Retrospectiva y retrasos

La principal dificultad a lo largo de este sprint fue mi desconocimiento con respecto al funcionamiento de los tokens JWT, añadido a que, a diferencia de otros frameworks como podrían ser NodeJS u otras distribuciones backend de javascript, springboot no ofrece manejos tan sencillos.

La problemática de comprender los tokens JWT correctamente, sumada a las dificultades que suponía programar este sistema en springboot llevaron al proyecto a un retraso de hasta una semana para este sprint.

3.5. Sprint 3

3.5.1. Introducción al Sprint 3

Durante el tercer sprint de la aplicación se llevará a cabo el desarrollo de funcionalidad correspondiente a las ligas, englobado su creación, asociación con usuarios, métodos de acceso y su gestión y configuración.

Puede observar la figura 3.25, la cual es un diagrama UML con las principales clases partícipes a lo largo de este sprint. Lo utilizaremos ahora para obtener una idea general del sprint y se irá desglosando poco a poco a lo largo de las diferentes explicaciones.

Destacar que tanto para este diagrama UML de la figura 3.25 como para los posteriores, se suprimirán de la imágenes de forma sucesiva clases que no aporten gran relevancia a las explicaciones ya que, si bien se usa por ejemplo la entidad del usuario a lo largo de este sprint y los siguientes, este no aporta mayor información al diagrama. Esto se hace con el objetivo de hacer los diagramas lo más comprensible que se pueda, ya que su escalado en complejidad y cantidad es alto.

Estructuras de datos principales

Para comenzar con la programación correspondiente a las ligas, he creado dos modelos de datos «League»²³, para representar a las ligas que crean los usuarios, con datos como el nombre, el tipo de comienzo, la liga real asociada o si los clausulazos están activos o no, y «UserXLeague»²⁴ para representar la información que relaciona a un usuario con una liga, como el dinero que tiene el usuario en esa liga o si es o no administrador de la misma.

Además, como hemos mencionado antes, las ligas tienen que estar asociadas a una liga real. Por ejemplo, la liga «Amigos de Alberto» debe estar

²³Código para el modelo de datos de las ligas.

²⁴Código para el modelo de datos de la asociación entre usuarios y ligas.

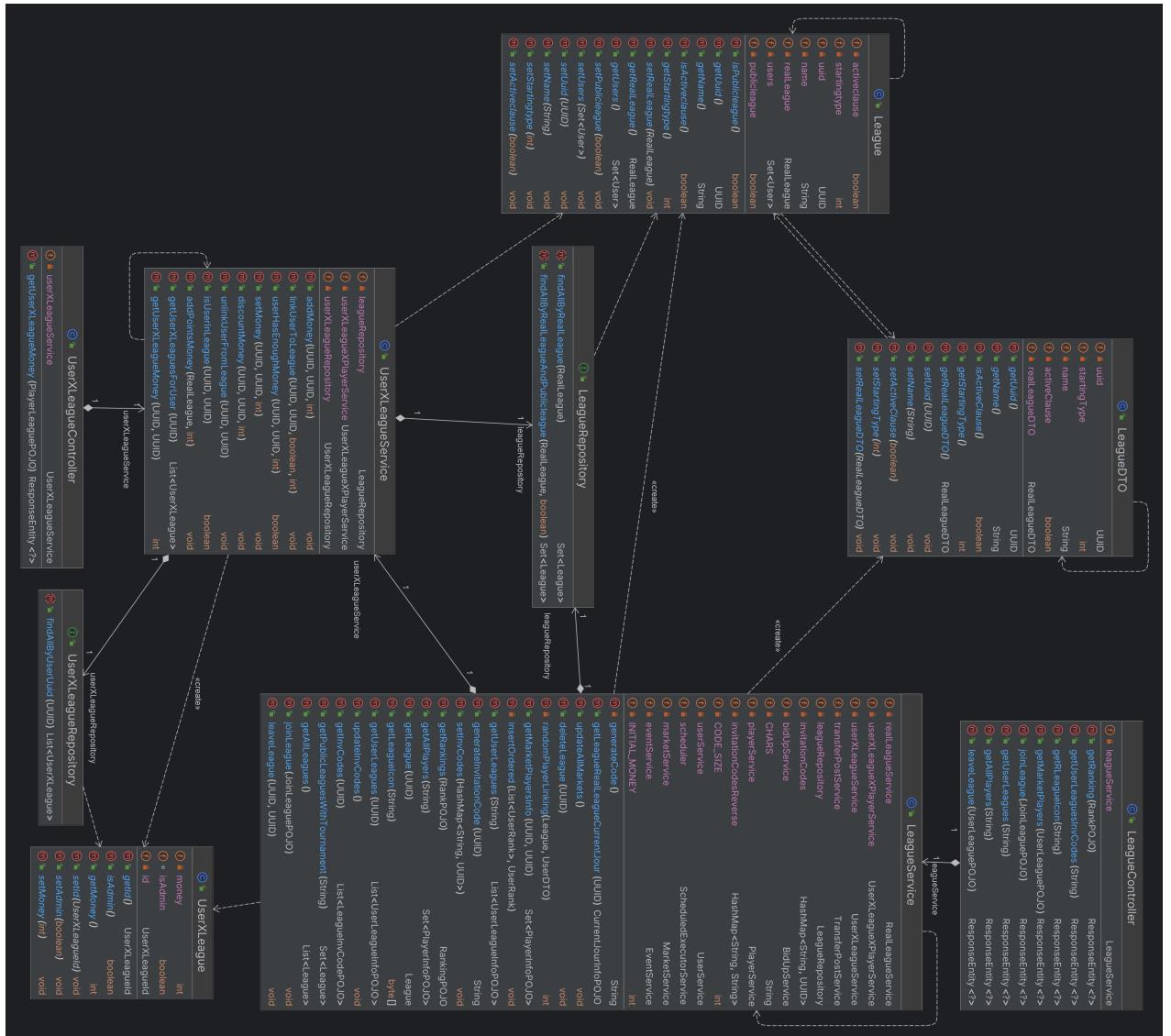


Figura 3.25: Diagrama UML del sprint 3.

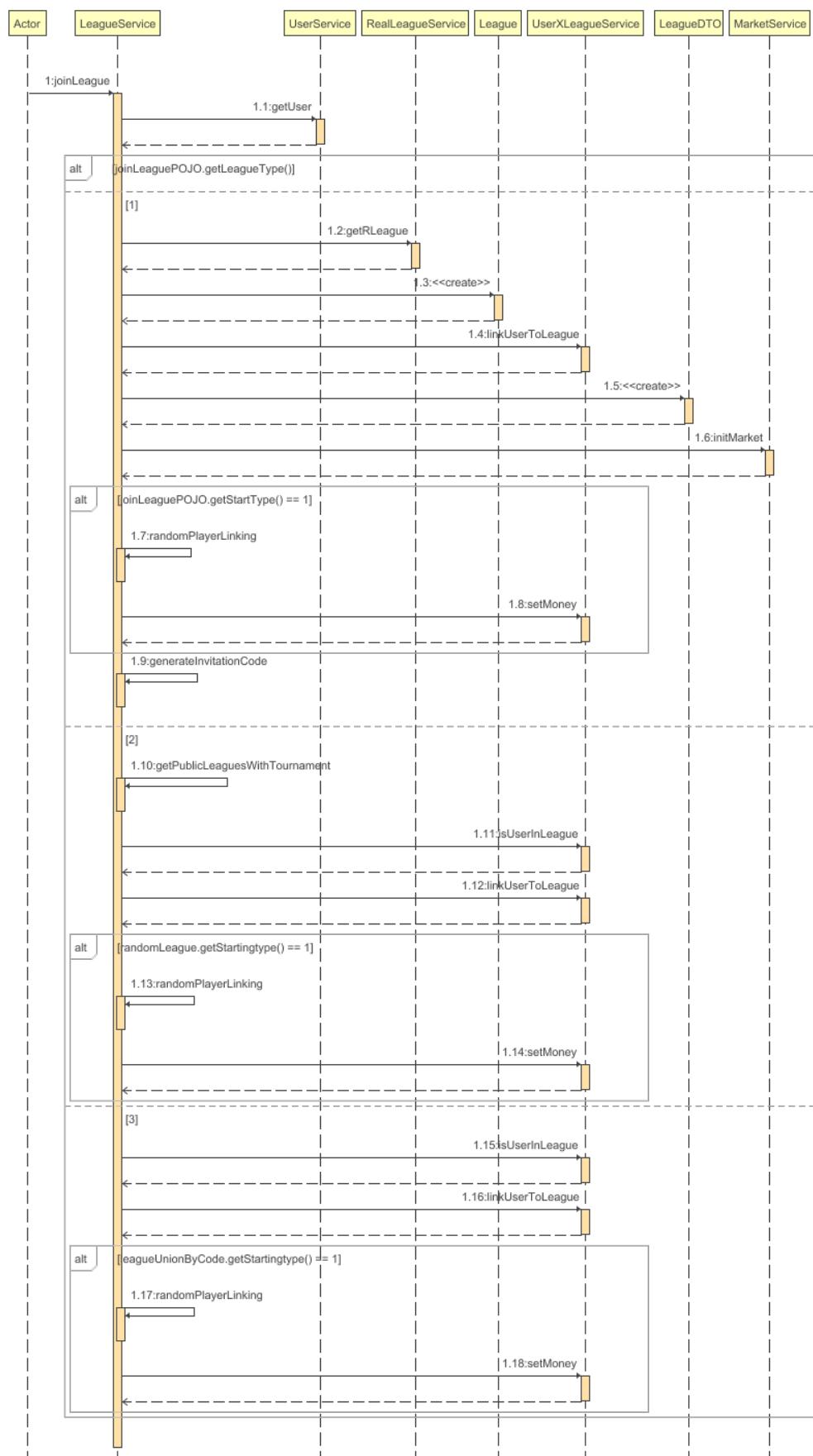
asociada a un liga de deportes electrónicos real, como por ejemplo la liga «superliga», que es la principal competición del videojuego League of Legends en España. Ya que sobre los datos de la liga seleccionada (jugadores, partidos, equipos) será sobre los que se desarrollará el juego. Sin embargo en este punto del desarrollo, no tenemos acceso a estos datos ya que aún no se ha desarrollado la HT 5 (implementación del sistema recolector de datos). Por lo que se operará de forma temporal sobre ficheros en formato «JSON», los cuales ofrecerán datos simulados para poder realizar el desarrollo de este sprint.

3.5.2. HU 2.1 y 2.3: Creación y abandono de ligas

En el diagrama UML de la figura 3.25, podemos observar dentro del servicio de liga²⁵ el método «joinLeague». Este método es el que convierte el servicio en el responsable de crear las ligas tal y como podemos ver en la asociación «create» del diagrama. Fijándonos en el primer condicional del diagrama de secuencia de la figura 3.26, podemos ver como el flujo es el siguiente:

1. Obtención de la liga real correspondiente.
2. Creación de la instancia de liga (junto a sus respectivas configuraciones).
3. Asociación como administrador entre el usuario creador y la liga.
4. Inicialización del mercado (se desglosa en sprint 4).
5. Asignación aleatoria de jugadores para el usuario creador si la opción está activa en la liga.
6. Asignación de dinero correspondiente al usuario en la liga.
7. Generación de código de invitación (desglosado en el apartado HU 2.2 de este capítulo).

²⁵Código para el Servicio de liga.

Figura 3.26: Diagrama de secuencia del procedimiento `joinLeague`.

Con respecto al apartado visual, se trató de aportar al usuario un entorno que fuese fácil, cómodo y dinámico, escondiendo la diferente configuración mediante el uso de un acordeón²⁶, y desplegando y cerrando automáticamente las pestañas, de forma que el usuario vea la mínima configuración necesaria.

The wireframe illustrates a user interface for selecting a league. It features a vertical stack of four expandable sections:

- Select League type**: Contains three options: "Create league" (with placeholder text "Lorem ipsum dolor sit amet et delectus"), "Join public league" (with placeholder text "Lorem ipsum dolor sit amet et"), and "Join friends league" (with placeholder text "Lorem ipsum dolor sit amet").
- Select a game**: Displays four small square icons, each containing a diagonal cross (X).
- Select a league**: Displays four small square icons, each containing a diagonal cross (X).
- Configuration**: A group of input fields:
 - "League name": An input field with a placeholder "I".
 - "Active clause option": A dropdown menu.
 - "Starting Type": A dropdown menu.
 - "Make public league": A dropdown menu.
- Introduce a code**: A section with a placeholder "Lorem ipsum dolor sit amet et delectus" and an input field.

Figura 3.27: Wireframe pantalla de selección de liga.

De esta forma conseguimos guiar al usuario en su proceso de toma de decisiones a la hora de crear la liga y no lo apabullamos con información que podría ser irrelevante para él. En el wireframe de la figura 3.27 podemos ver la pantalla con el acordeón completamente desplegado. La primera opción corresponde a la selección de tipo de unión, permitiéndonos escoger entre crear nuestra liga propia, unirnos una pública o unirse a una liga de amigos mediante código (ver figura 3.28), el segundo apartado del acordeón nos permite escoger para qué videojuego queremos la liga (ver figura 3.29), la tercera opción del acordeón permite seleccionar un liga de la lista de competiciones disponibles (ver figura 3.30). La cuarta opción del acordeón (solo

²⁶Acordeón angular material.

disponible si se escogió la creación propia de liga), nos permite realizar la configuración de la liga, como colocar el nombre, seleccionar si queremos las cláusulas activas, el modo de comienzo o si es una liga pública (ver figura 3.31).

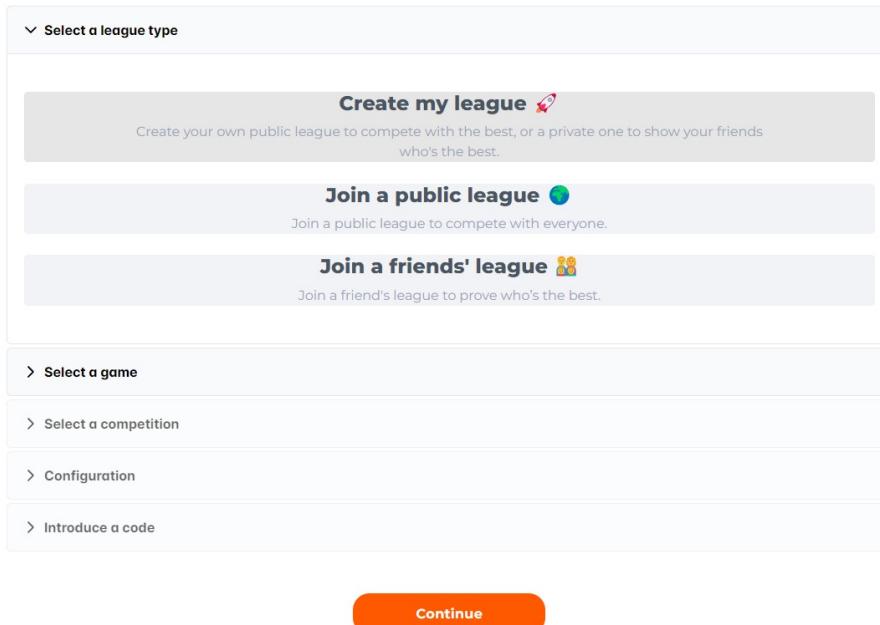


Figura 3.28: Selección tipo de unión a liga.

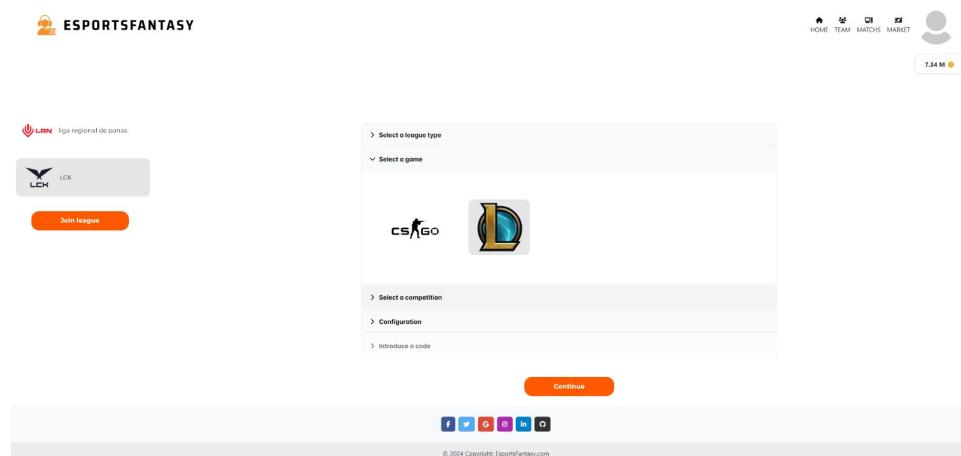


Figura 3.29: Selección de juego.

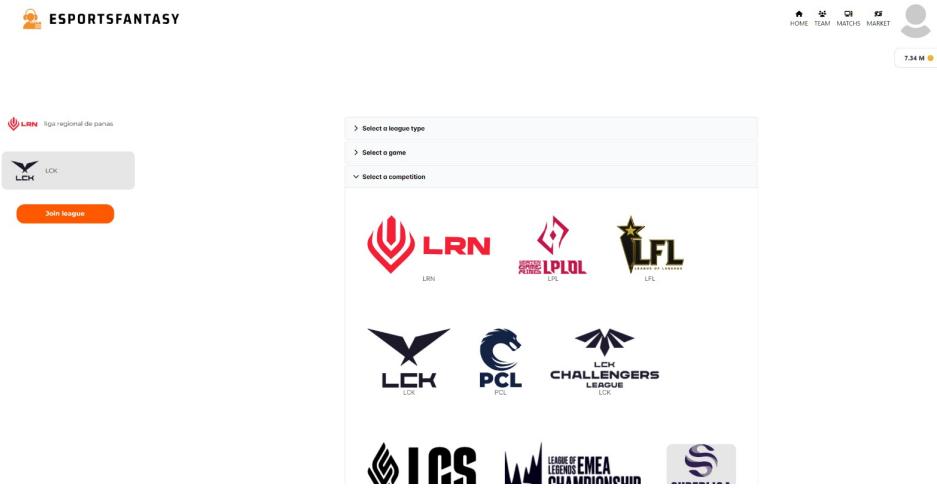


Figura 3.30: Selección de liga real.

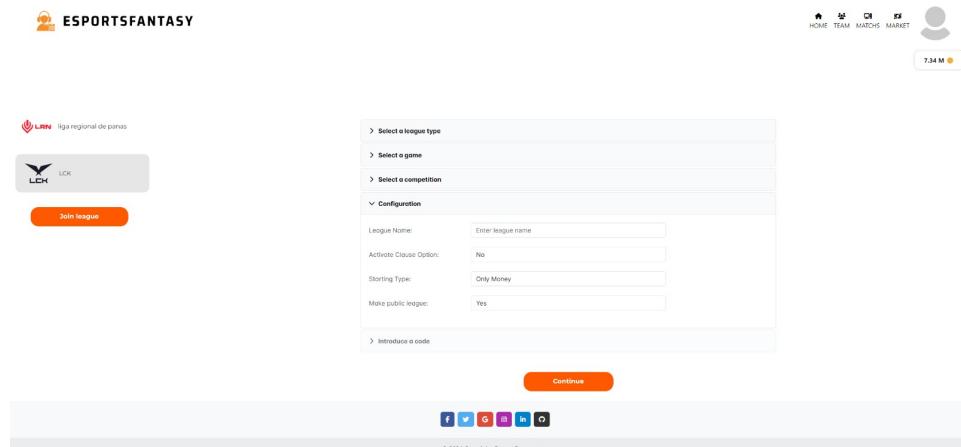


Figura 3.31: Configuración de liga a crear

Con respecto al abandono de ligas habrá que ser cuidadoso, ya que es importante no dejar tuplas en la base de datos que puedan dar lugar a problemas de integridad referencial más adelante y por consiguiente, desligar todos los datos del usuario a la liga y sus derivados. Podemos seguir el flujo de la figura 3.32.

1. Desvinculación del historial de equipo de jugadores con respecto al usuario.
2. Eliminación de la propiedad de los jugadores en el mercado. Y puesta de los jugadores como "disponibles para venta".
3. Desvinculación del usuario con la liga.

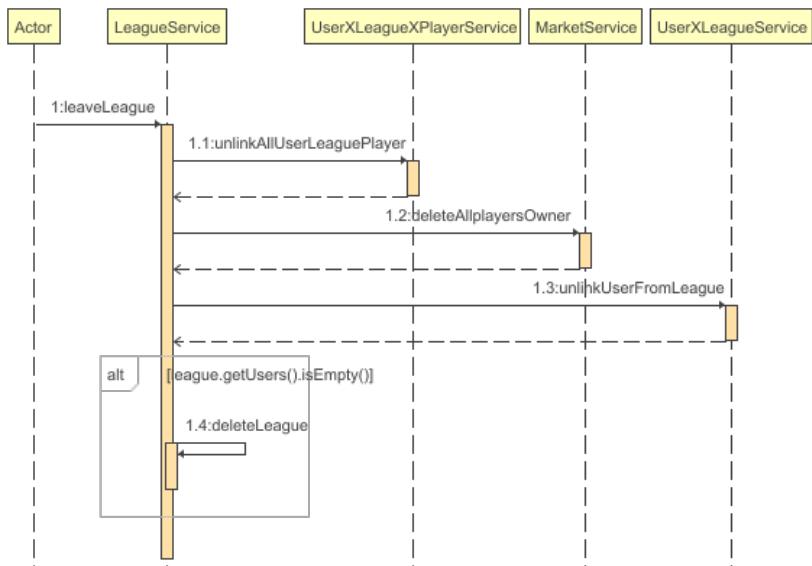


Figura 3.32: Diagrama de secuencia, abandono de liga.

Podemos destacar que no se ha hecho una eliminación de datos ni para el historial de transacciones ni para el registro de pujas. Esto se debe a que no hay restricciones de integridad referencial de estas dos tablas con respecto a tablas que impliquen propiedad (como la de market o la de UserXLeagueX-Player), por lo que he decidido mantener la información a modo de histórico.

Con respecto al frontend, he decidido introducir la lista de ligas del usuario en su perfil a modo de escaparate. A estas, les he añadido aquí una opción de abandono. Dado que no es una funcionalidad grande y puede casar con la información del usuario, he decidido introducir esta opción aquí con el objetivo de ahorrar costes y tiempo de programación en crear un nuevo componente Angular únicamente para la eliminación de ligas. Además he añadido un modal de confirmación que además explica las posibles consecuencias de abandonar una liga, para evitar accidentes y concienciar sobre las consecuencias directas de abandono dado que no hay posibilidad de recuperar la información una vez se abandona una liga.

3.5.3. HU 2.2: Invitar a ligas

Los códigos de invitación se crearán generando cadenas de 6 caracteres de forma aleatoria. Estos se introducirán en una pila de códigos de invitación, de la que expirarán a las 24 horas. Bajo ninguna circunstancia podrá haber dos códigos iguales ya que, aunque se generen de forma aleatoria (lo cual ya lo hace poco probable), antes de que se introduzca el código en la pila, se comprueba que no exista ya, en cuyo caso generará un nuevo código

hasta introduzca uno que no exista.

Con este sistema lo único que podría llegar a preocupar es que el volumen de ligas creadas llegase a un punto tal que pudiese llegar a crear ciertas ralentizaciones a la hora de generar códigos por encontrar códigos repetidos, sin embargo, analicemos las posibilidades brevemente: Teniendo en cuenta que los códigos se generan con letras mayúsculas, entre letras y dígitos tenemos un total de 36 caracteres disponibles por cada posición. Siendo 6 posiciones, obtendremos un total $36^6 = 2.176.782.336$ posibles códigos. Suponiendo que alcanzásemos un 0.01 % de el índice de capacidad, nos daría un total de 217.678 códigos (por consiguiente ligas generadas). Teniendo en cuenta que las expectativas volumen de ligas no alcanzan este número ni como ligas totales, cuanto menos si tenemos en cuenta que las ligas van desapareciendo según sus contra partes de la vida real van finalizando. De este modo la posibilidad de ralentización por duplicidad de códigos no es algo de lo que preocuparse para este proyecto. Aún así, si es que llegase a serlo en algún momento, bastaría con añadir letras minúsculas como posibilidad o incluso un carácter más para aumentar drásticamente las opciones de códigos posibles.

A la hora de solicitar una unión mediante código, simplemente se buscará este en un «HashMap» que contenga un mapeo entre el código y el identificador de la liga. Esto nos permitirá encontrar la liga asociada al código de manera rápida y eficiente. A continuación se procederá a la unión del jugador en la liga tal y como se describe en el apartado anterior a partir de la asignación aleatoria.

Con respecto al aspecto visual, mantenemos la unión mediante código escondida en el acordeón para no dar información irrelevante a los usuarios que simplemente quieren crear sus propias ligas o unirse a ligas públicas. La opción del acordeón para unirse a un código mediante liga deshabilita automáticamente todas las demás, ya que no será necesario especificar más información y solo estará disponible si el usuario escogió en la primera opción la unión mediante código (ver 3.33).

En el mismo momento de generación de código, además de tener un «Hashmap» de tupla «código-uuidliga», también se tendrá otro de tupla «uuidliga-código» que se irá llenando a la misma vez que su contra-parté. De esta manera, tal y como podemos ver en las figuras 3.34 y 3.35 en la pantalla de usuario en la que desplegamos la lista de ligas y junto a sus botones de abandono de liga, también incluiremos el código de acceso correspondiente a esa liga, ya que será fácilmente accesible mediante la uuid de la misma gracias al «hash» reverso que se ha mencionado anteriormente.

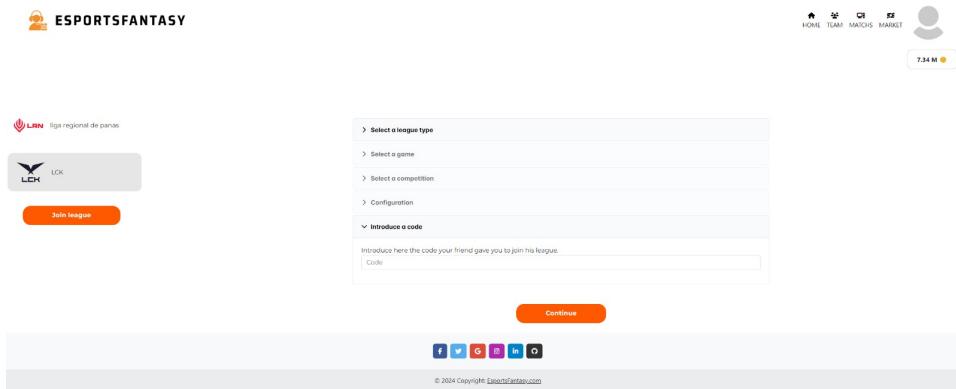


Figura 3.33: Unión a liga mediante código.



Figura 3.34: Wireframe de configuración de liga.

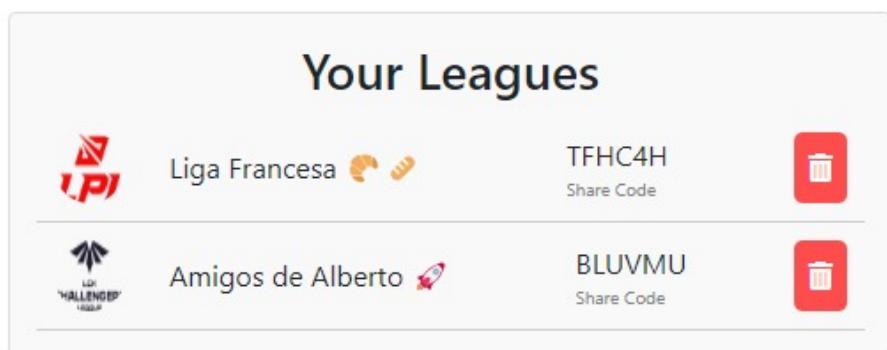
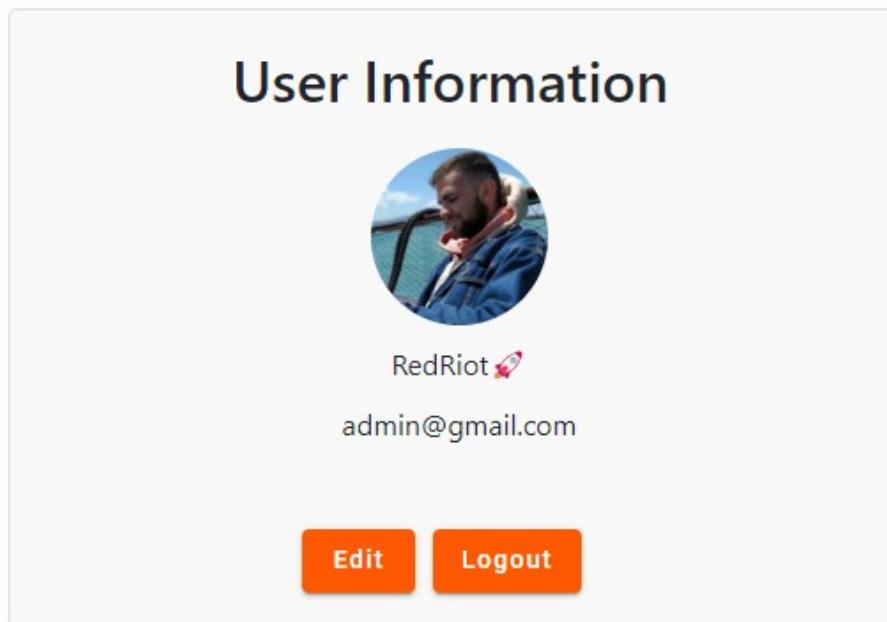


Figura 3.35: Lista de ligas de un usuario con códigos y opción de salir de la liga.

3.5.4. HU 2.4: Acceso a ligas asociadas

Dado que un usuario puede estar asociado a múltiples ligas, será necesaria una manera de obtener la información indispensable de estas ligas. En la figura 3.25 podemos observar que el controlador de ligas²⁷ contiene un endpoint llamado «getUserLeagues». Este endpoint, a través del servicio unificará los datos correspondientes con la información mínima indispensable de las ligas en las que está un usuario (como son el nombre, el ícono y el UUID).

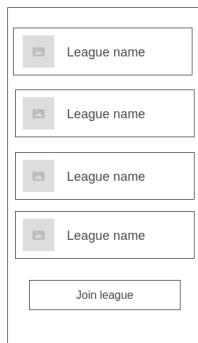


Figura 3.36: Wireframe lista de ligas.

Con esta información, en frontend se ha creado el aside que podemos ver en las figuras 3.36 y 3.37 que mostrará las diferentes ligas, manteniendo como activa siempre una de ellas, de esta forma el usuario podrá de un único golpe de vista ver en qué liga está situado actualmente y navegar entre ellas.

Un servicio frontend (el cual cabe destacar que no es lo mismo que los servicios a los que se ha estado haciendo alusión hasta ahora para denominar a los servicios del backend [Ang23]) se encargará de gestionar la información interna de liga seleccionada. El cambio de ligas provocará unas u otras llamadas en función de la pantalla que el usuario tenga seleccionada (equipo, mercado, ranking, etc), actualizando esta información interna. Además esto facilitará la gestión de llamadas a los diferentes servicios, pues será siempre fácil acceder al identificador de la liga de la que se desea obtener información.

Podemos hacernos una idea de como quedará cualquier pantalla de un usuario que haya iniciado sesión, mostrando a su izquierda la lista de ligas y a la derecha la pantalla principal que corresponda en cada momento (ver 3.38).

²⁷Código para el controlador de ligas.

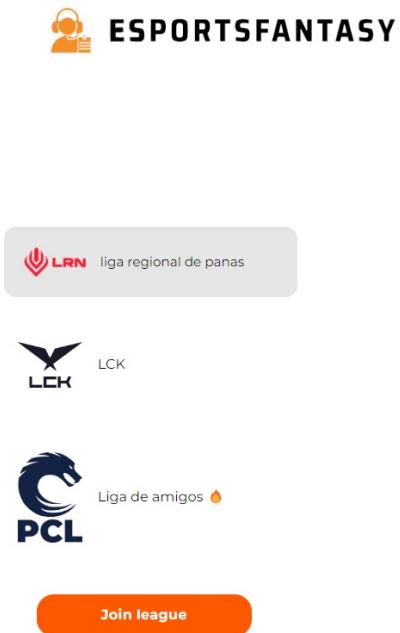


Figura 3.37: Menú de lista de ligas para un usuario.

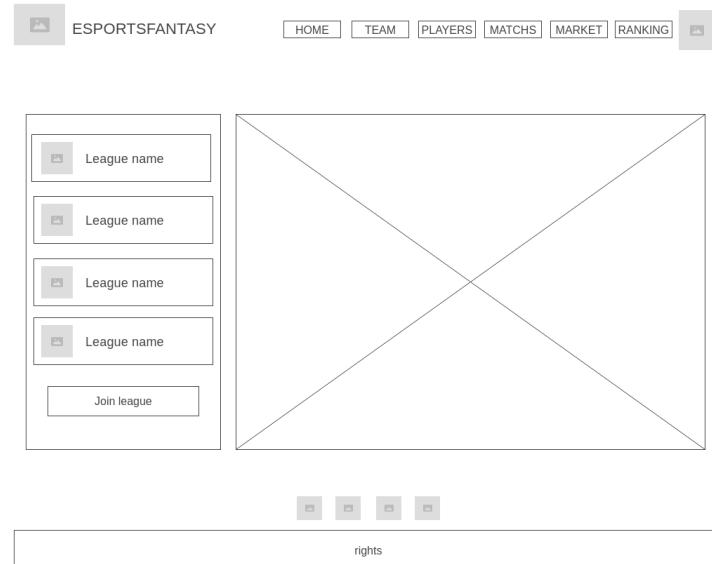


Figura 3.38: Wireframe de muestra de estructura de pantallas.

3.5.5. Tests unitarios y de integración

Para este sprint se han realizado un total de 9 tests unitarios²⁸ y 6 tests de integración.

Desglose tests unitarios

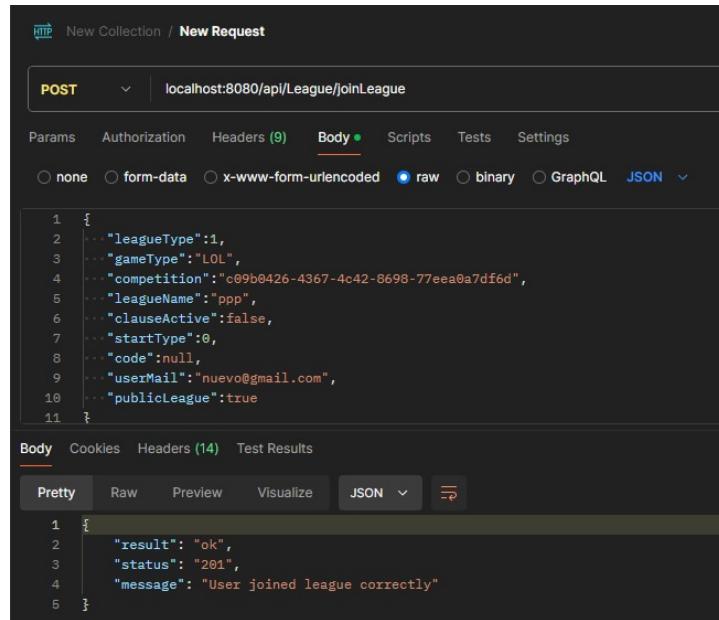
Test	HU	Descripción	Código de error	Resultado
testJoinLeague_Case1	2.1 - 2.2	Comprueba que se cree una liga y genere código de invitado.	-	✓
testJoinLeague_Case2	2.2	Comprueba que se pueda acceder a ligas públicas.	-	✓
testJoinLeague_Case3	2.2	Comprueba que se pueda acceder a ligas por código de invitado.	-	✓
testJoinLeague_ErrorInvitationCode	2.2	Comprueba que el código de invitación exista en la pila de códigos.	1012	✓
testJoinLeague_ErrorLeagueExists	2.2	Comprueba que el código esté asociado a una liga.	1013	✓
testJoinLeague_ErrorUserInLeague	2.2	Comprueba que el usuario no esté ya en la liga.	1014	✓
testJoinLeague_ErrorLeagueWithSpecifiedTournament	2.2 - 2.4	Comprueba que la competición especificada exista en liga.	1010	✓
testJoinLeague_ErrorNoPublicLeague	2.2 - 2.4	Comprueba si la liga pública especificada existe.	1011	✓
testRemovePlayerFromLeague	2.3	Comprueba si se ha eliminado correctamente a un usuario de una liga.	-	✓

Tabla 3.3: Listado test unitarios sprint 3.

- **testJoinLeague_Case1:** Comprueba que se cree una liga y que se genere un código de invitado correctamente.
- **testJoinLeague_Case2:** Verifica que un usuario pueda unirse a ligas públicas sin problemas.
- **testJoinLeague_Case3:** Valida que un usuario pueda acceder a ligas mediante un código de invitado.
- **testJoinLeague_ErrorInvitationCode:** Comprueba que el código de invitación proporcionado exista en la lista de códigos disponibles. Devuelve el error 1012 en caso de falla.
- **testJoinLeague_ErrorLeagueExists:** Verifica que el código de invitación esté efectivamente asociado a una liga existente. Devuelve el error 1013 si no se encuentra una liga asociada.
- **testJoinLeague_ErrorUserInLeague:** Asegura que el usuario no esté ya registrado en la liga. Devuelve el error 1014 si el usuario ya es miembro.
- **testJoinLeague_ErrorLeagueWithSpecifiedTournament:** Comprueba que la competición especificada exista en la liga. En caso de fallo, devuelve el error 1010.
- **testJoinLeague_ErrorNoPublicLeague:** Verifica la existencia de una liga pública especificada. Devuelve el error 1011 si la liga pública no existe.
- **testRemovePlayerFromLeague:** Comprueba que un usuario haya sido eliminado correctamente de una liga.

²⁸Código para los tests de servicios de liga.

Desglose tests integración



The screenshot shows a Postman interface with the following details:

- Method:** POST
- URL:** localhost:8080/api/League/joinLeague
- Body (raw JSON):**

```

1 {
2   "leagueType":1,
3   "gameType":"LOL",
4   "competition":"c9b0426-4367-4c42-8698-77eea0a7df6d",
5   "leagueName":"ppp",
6   "clauseActive":false,
7   "startType":0,
8   "code":null,
9   "userMail":"nuevo@gmail.com",
10  "publicLeague":true
11 }

```

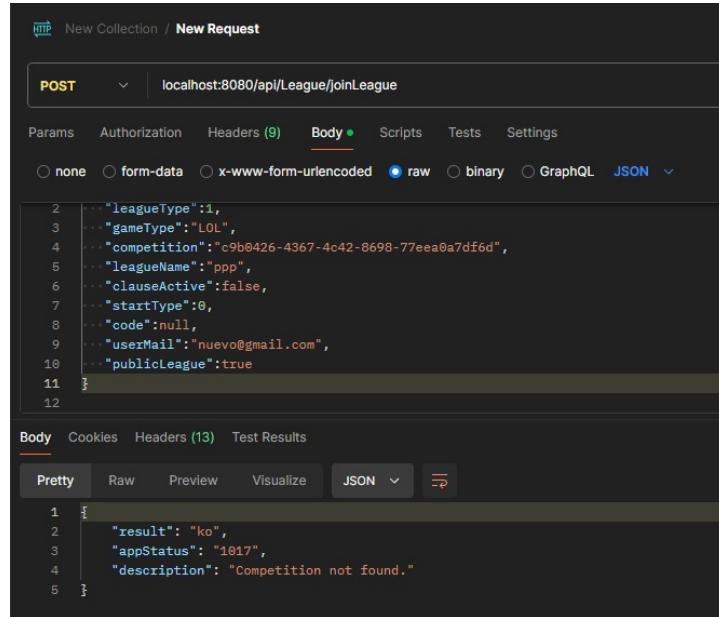
- Response (Pretty JSON):**

```

1 {
2   "result": "ok",
3   "status": "201",
4   "message": "User joined league correctly"
5 }

```

Figura 3.39: Petición a endpoint /joinLeague con credenciales válidos.



The screenshot shows a Postman interface with the following details:

- Method:** POST
- URL:** localhost:8080/api/League/joinLeague
- Body (raw JSON):**

```

2 {
3   "leagueType":1,
4   "gameType":"LOL",
5   "competition":"c9b0426-4367-4c42-8698-77eea0a7df6d",
6   "leagueName":"ppp",
7   "clauseActive":false,
8   "startType":0,
9   "code":null,
10  "userMail":"nuevo@gmail.com",
11  "publicLeague":true
12 }

```

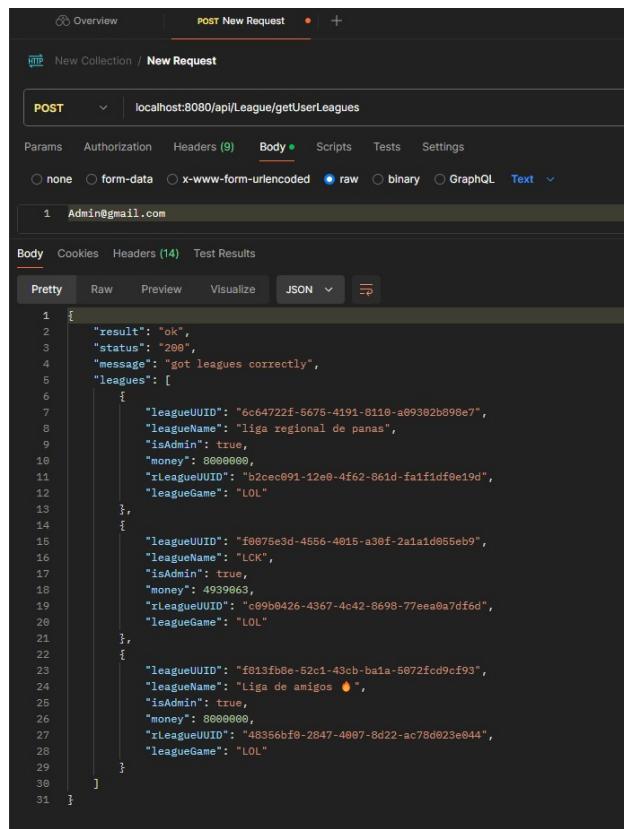
- Response (Pretty JSON):**

```

1 {
2   "result": "ko",
3   "appStatus": "1017",
4   "description": "Competition not found."
5 }

```

Figura 3.40: Petición a endpoint /joinLeague con credenciales inválidos.

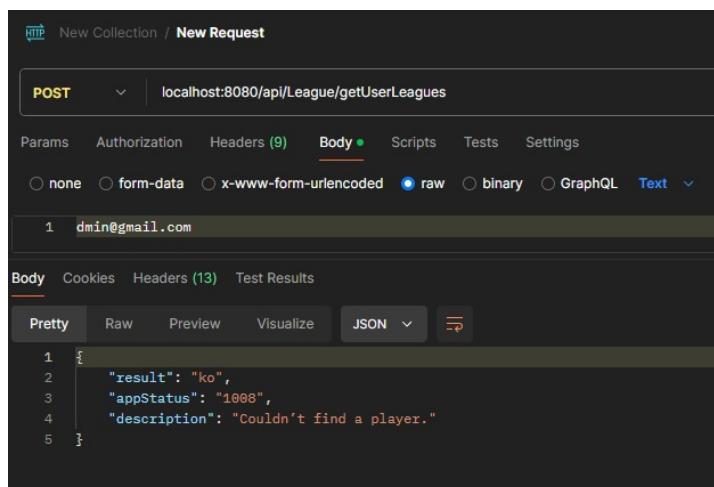


The screenshot shows a Postman interface with the following details:

- Method:** POST
- URL:** localhost:8080/api/League/getUserLeagues
- Body:** Raw JSON response (Pretty printed)
- Response Body:**

```
1 {
  "result": "ok",
  "status": "200",
  "message": "got leagues correctly",
  "leagues": [
    {
      "leagueUUID": "6c64722f-5675-4191-8110-a69382b898e7",
      "leagueName": "liga regional de panas",
      "isAdmin": true,
      "money": 8000000,
      "rleagueUUID": "b2cec091-12e0-4f62-861d-fa1f1df0e19d",
      "leagueGame": "LOL"
    },
    {
      "leagueUUID": "f0875e3d-4556-4015-a30f-2a1a1d055eb9",
      "leagueName": "LCK",
      "isAdmin": true,
      "money": 4939963,
      "rleagueUUID": "c69b9426-4367-4c42-8698-77eea0a7df5d",
      "leagueGame": "LOL"
    },
    {
      "leagueUUID": "f013fbbe-52c1-43cb-ba1a-5072fc09cf93",
      "leagueName": "Liga de amigos 🎉",
      "isAdmin": true,
      "money": 8000000,
      "rleagueUUID": "48356bf0-2847-4007-8d22-ac78d023e044",
      "leagueGame": "LOL"
    }
  ]
}
```

Figura 3.41: Petición a endpoint /getUserLeagues con credenciales válidos.

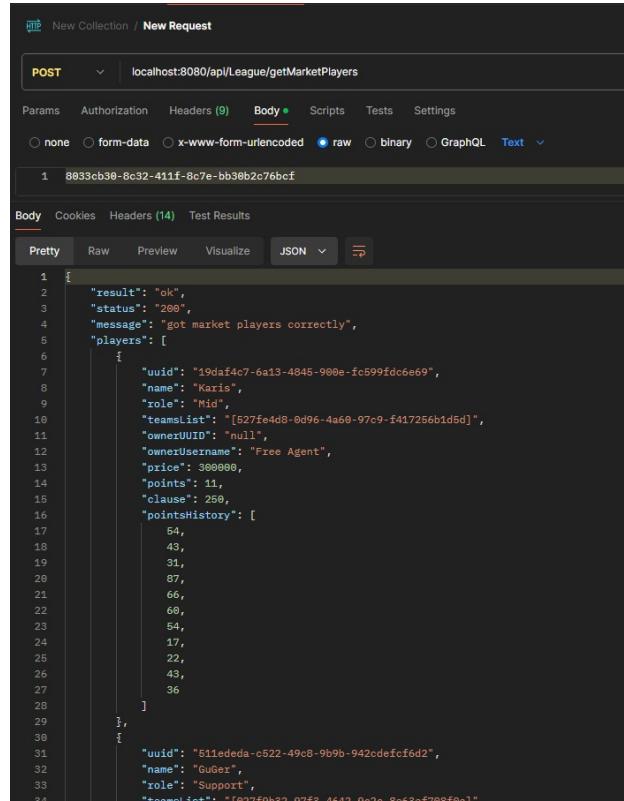


The screenshot shows a Postman interface with the following details:

- Method:** POST
- URL:** localhost:8080/api/League/getUserLeagues
- Body:** Raw JSON response (Pretty printed)
- Response Body:**

```
1 {
  "result": "ko",
  "appStatus": "1008",
  "description": "Couldn't find a player."
}
```

Figura 3.42: Petición a endpoint /getUserLeagues con credenciales inválidos.

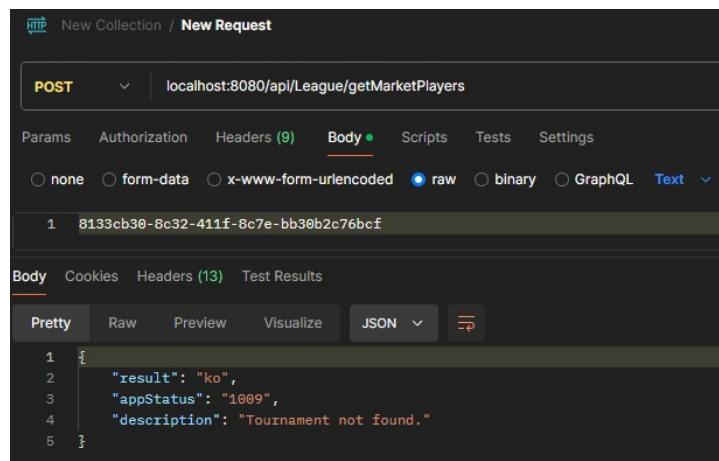


The screenshot shows a Postman collection named "New Collection" with a "New Request". The request is a POST to "localhost:8080/api/League/getMarketPlayers". The "Body" tab is selected, showing the raw JSON response. The response is a success (status 200) with the message "got market players correctly". It contains a "players" array with two objects, each representing a player with fields like uuid, name, role, teamsList, ownerUserID, ownerUsername, price, points, clause, and pointsHistory.

```

1 {
2   "result": "ok",
3   "status": "200",
4   "message": "got market players correctly",
5   "players": [
6     {
7       "uuid": "19daf4c7-6a13-4845-900e-fc599fdc6e69",
8       "name": "Karlis",
9       "role": "Mid",
10      "teamsList": "[527fe4d8-0d96-4a60-97c9-f417256b1d5d]",
11      "ownerUserID": "null",
12      "ownerUsername": "Free Agent",
13      "price": 300000,
14      "points": 11,
15      "clause": 250,
16      "pointsHistory": [
17        54,
18        43,
19        31,
20        87,
21        66,
22        66,
23        54,
24        17,
25        22,
26        43,
27        36
28      ],
29    },
30    {
31      "uuid": "511ededa-c522-49c8-9b9b-942cdefcf6d2",
32      "name": "GuGex",
33      "role": "Support",
34      "teamsList": "[527fe4d8-0d96-4a60-97c9-f417256b1d5d]"
35    }
36  ]
37}
  
```

Figura 3.43: Petición a endpoint /getMarketPlayers con credenciales válidos.



The screenshot shows a Postman collection named "New Collection" with a "New Request". The request is a POST to "localhost:8080/api/League/getMarketPlayers". The "Body" tab is selected, showing the raw JSON response. The response is a failure (status 401) with the message "Tournament not found." and an appStatus of "1009".

```

1 {
2   "result": "ko",
3   "appStatus": "1009",
4   "description": "Tournament not found."
5 }
  
```

Figura 3.44: Petición a endpoint /getMarketPlayers con credenciales inválidas.

Equivalencias HU/HT a pruebas de aceptación

- **HU 2.1:** Se debe verificar la liga se crea correctamente.
 - **TU:** testJoinLeagueCase1.
 - **TI:** Petición a /joinLeague con credenciales válidos. Ver figura 3.39.
 - **TI:** Petición a /joinLeague con credenciales inválidos. Ver figura 3.40.
- **HU 2.2:** Se debe verificar que el código que el código se haya creado correctamente, la liga exista y el usuario no esté ya en esa liga.
 - **TU:** testJoinLeagueCase1.
 - **TU:** testJoinLeagueCase3.
 - **TU:** testJoinLeagueErrorInvitationCode.
 - **TU:** testJoinLeagueErrorLeagueExists.
 - **TU:** testJoinLeagueErrorUserInLeague.
 - **TU:** testJoinLeagueErrorNoPublicLeague.
 - **TU:** testJoinLeagueErrorLeagueWithSpecifiedTournament.
- **HU 2.3:** Se debe verificar que el usuario ya no es parte de la liga.
 - **TU:** testRemovePlayerFromLeague.
- **HU 2.4:** Se debe verificar que se obtiene la lista de ligas asociadas.
 - **TI:** Petición a getUserLeagues. Ver figura 3.41.
 - **TI:** Petición inválida a getUserLeagues. Ver figura 3.42.

3.5.6. Retrospectiva y retrasos

Durante este sprint no se han encontrado dificultades más allá de que era un sprint denso en cuestión de picar mucho código.

No se produjeron retrasos con respecto a la programación estimada originalmente.

3.6. Sprint 4

3.6.1. Introducción al Sprint 4

En este cuarto sprint nos encontramos ante el principal grueso del proyecto. Nos centraremos en todo lo relacionado con los jugadores. La obtención y normalización de datos, el sistema de actualizaciones de mercado, las

transferencias, compraventa y sistema de cláusulas. Además de gestionar la alineación de los jugadores por parte de los usuarios en las diferentes etapas del juego.

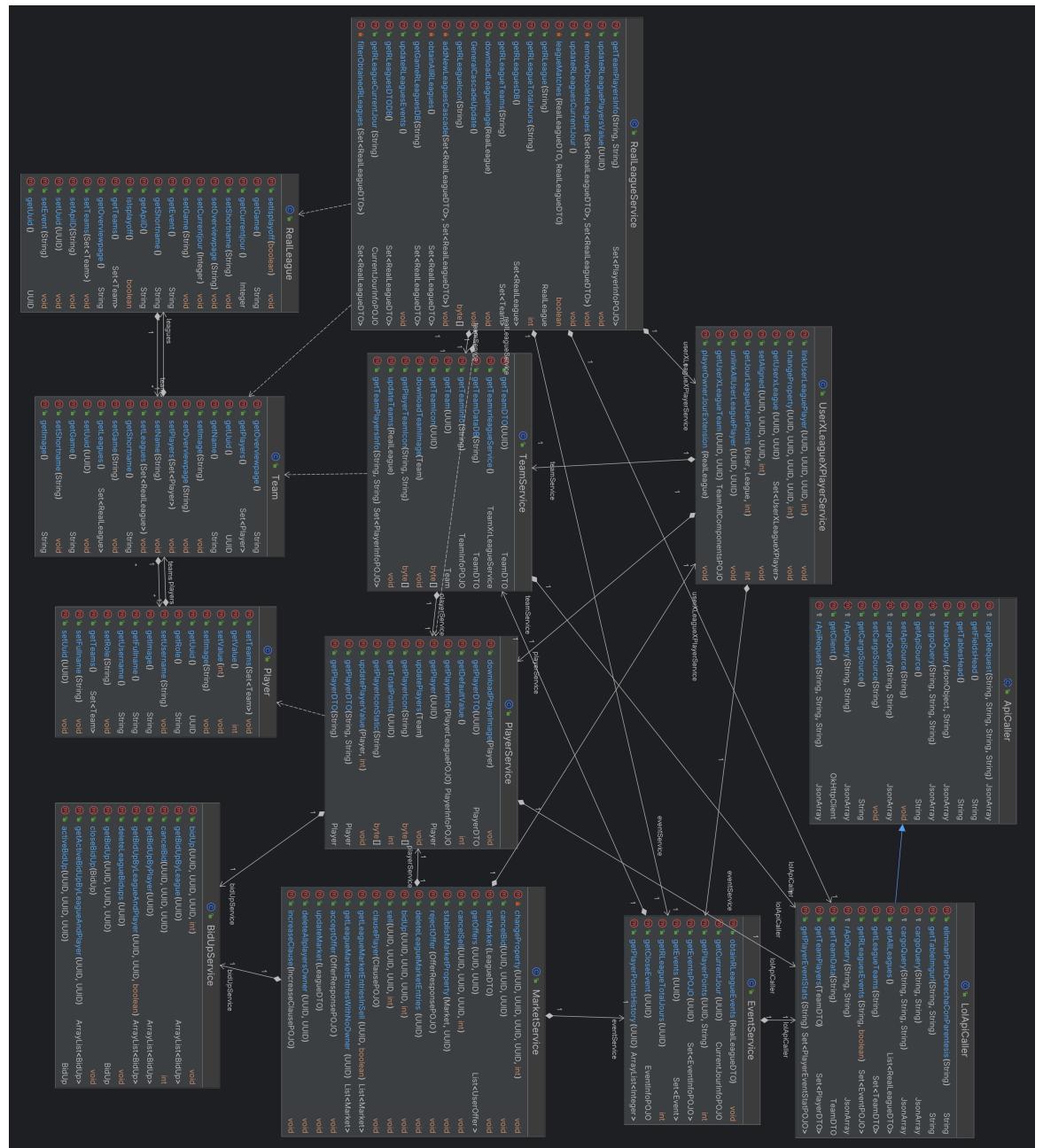


Figura 3.45: Diagrama UML del sprint 4.

En la figura 3.45 podemos obtener una visión general de la estructura del sprint. Dado el volumen creciente del diagrama, a diferencia de los anteriores, se han eliminado componentes como los controladores o los repositorios, ya que a estas alturas del trabajo se sobreentiende su comprensión y dificultaría la legibilidad del diagrama. A continuación procederemos a desgranarlo en las explicaciones pertinentes a cada clase.

3.6.2. HT 5: Implementación del sistema recolector de datos

Tal y como habíamos mencionado anteriormente en el sprint 3, se había estado trabajando sobre datos de jugadores, equipos, ligas y partidos simulados. Durante esta historia técnica se explicará el proceso de obtención y normalización de datos.

eSportsFantasy está pensado para ser muy escalable hacia múltiple variedad de deportes electrónicos, esto conlleva obtener la información necesaria de múltiples y variadas fuentes de datos, por consiguiente deberán ser normalizada para alcanzar una estructura común sobre la que poder operar de forma generalizada. En la parte superior central y superior derecha de la figura 3.45 podemos observar dos clases, «*ApiCaller*»²⁹ y «*LolApiCaller*»³⁰. «*ApiCaller*» es el servicio de peticiones, este permitirá a los diferentes servicios de obtención de datos específicos de cada videojuego abstraerse del modelo de solicitud, ya sean llamadas de tipo cargo, a una api o incluso scrapping; unificando así el sistema de llamadas en cada uno de los diferentes deportes electrónicos. De esta clase, heredaran todas las clases de solicitud de datos.

De esta manera, se ha implementado «*LolApiCaller*» a modo de ejemplo de sistema de obtención de datos. En este caso se obtienen datos para las competiciones del videojuego League of Legends. Para ello se han usado principalmente dos de los tres sistemas de peticiones que heredamos de «*ApiCaller*», siendo estos las queries de tipo cargo y el scrapping web para las imágenes.

Aclararemos antes de continuar el concepto de cargo query. Llamamos «cargo query»³¹ a la posibilidad que ofrece MediaWiki³² de consultar los datos de su base de datos de forma relativamente sencilla y estructurada. Así pues, las páginas soportadas sobre MediaWiki, como pueden ser Lea-

²⁹Código para el sistema de llamadas.

³⁰Código para el sistema de obtención de información del videojuego League of Legends.

³¹Extensión cargo.

³²MediaWiki.

guepedia³³, soportan este tipo de solicitudes, devolviéndonos la información en formato json. Aprovechar además para aclarar con respecto al scrapping web que se realiza a dicha página para obtener las diferentes imágenes, que se cuenta con el total permiso de la misma, siendo el scrapping el formato recomendado por los moderadores para obtener las imágenes en los canales oficiales de soporte.

Así pues, un ejemplo simple de cargo query construido a partir de las clases «ApiCaller» y «LolApiCaller» podría ser la siguiente: <https://lol.fandom.com/api.php?action=cargoquery&format=json&tables=CurrentLeagues-CL&fields=Event,OverviewPage&limit=500&offset=0>, en el que obtenemos la lista de ligas que se están disputando actualmente. A partir del listado de tablas³⁴ realizando diferentes operaciones de base de datos, se puede obtener prácticamente la totalidad de información de la web.

Si bien se podría pensar de primeras que implementar este sistema como una interfaz en vez de como herencia podría facilitar el trabajo futuro al no tener que diferenciar en el código entre los diferentes deportes electrónicos en las llamadas del resto del código, finalmente decidí que no, ya que los diferentes juegos y ligas son demasiado diferentes entre sí, cuentan con unos requerimientos en muchas ocasiones demasiado dispares como para agruparlos a todos en la misma interfaz. No tendría caso implementar una interfaz si solo se va a aportar una o dos implementaciones a un método en el mejor de los casos.

Una vez contamos con un sistema normalizado de peticiones y un ejemplo de obtención de datos para un deporte electrónico, se procede a crear un sistema común de obtención de datos. Este es el que he llamado «general-CascadeUpdate», para el cual podemos observar un diagrama de secuencia en la figura 3.46 dentro del servicio «RealLeagueService». El proceso es el siguiente:

1. Obtención de las ligas que actualmente están teniendo lugar.
2. Filtrado de las ligas para eliminar segundas divisiones o ligas desconocidas que pueden generar problemas por falta de normalización de información en las bases de datos de los recursos.
3. Eliminación de ligas obsoletas.
4. Inclusión de la información de las nuevas ligas en la base de datos.

³³Leaguepedia.

³⁴Listado de tablas Leaguepedia.

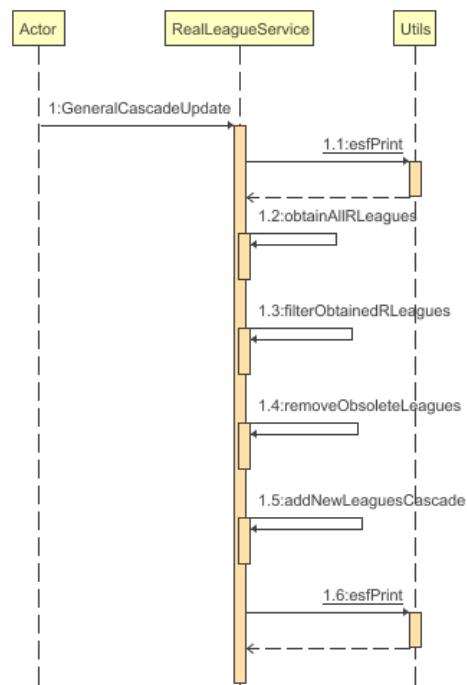


Figura 3.46: Diagrama de secuencia del general cascade update.

Haremos hincapié en este último punto de la inclusión de nuevas ligas por ser el menos trivial. Podemos consultar el diagrama de secuencia en la figura 3.47. El proceso será el siguiente:

1. Creación del evento (lo que en el tfg llamamos, liga real) en base de datos.
2. Creación de los equipos en base de datos.
3. Creación de los jugadores en base de datos.
4. Obtención de la imagen de liga real.
5. Asociación entre los equipos y las ligas.
6. Obtención de la imagen de cada equipo.
7. Asociación entre los equipos y los jugadores.
8. Obtención de la imagen de cada jugador.

Con respecto a los eventos (esta vez sí, usando nuestra nomenclatura para referirnos con la palabra eventos a los diferentes partidos que suceden en una liga) se ha creado una tarea programada aparte, ya que necesitan una frecuencia de refresco mucho mayor que el resto de datos pues es importante que la posibilidad de colocar jugadores en tu equipo como usuario quede cerrada antes de que una jornada comience para evitar así posibles cambios en las decisiones que los usuarios toman como entrenadores.

Recordamos además que, en función de a qué deporte electrónico pertenezca cada liga, se hará llamando a uno u otro sistema recolector de datos tanto para el recolector general de cascada como para el de eventos. En este caso, como solo se obtienen datos de League of Legends, se hace mediante «LolApiCaller».

Este proceso de actualización de datos se realizará una vez al día, con el objetivo de obtener información para nuevas ligas, eliminar ligas obsoletas o incluso actualizar información de las ligas actuales en casos como posibles bajas de jugadores.

3.6.3. HT 8, HU 3.1 y 3.2: sistema rotativo de mercado y compraventa de jugadores

Para la gestión del mercado de jugadores tenemos, principalmente dos modelos de datos de especial relevancia, «market»³⁵ y «bidup»³⁶.

Market es el encargado de almacenar el estado de un jugador en una liga, con atributos como el precio de venta, el precio de clausula, el propietario y su estado de mercado (si está en venta o no). En una primera instancia podríamos pensar que no es necesario almacenar una tupla por cada jugador en cada liga, sin embargo, si lo pensamos de una forma más detenida, esta es una forma muy efectiva de almacenar información que es cambiante entre cada liga para cada uno de los jugadores como es el precio de venta o el precio de cláusulas, por lo que tener esta tabla nos permite abordar estos múltiples problemas a la misma vez.

Por otro lado tenemos la tabla «bidup», la cual almacena la lista de pujas que realiza cada usuario a cada jugador. También podría parecer inútil ya que, por ejemplo, podríamos introducir en market un campo «maxbidup», en el que almacenar siempre el máximo pujador, sin embargo, esto eliminaría la posibilidad de poder observar el historial de pujas que recibe un jugador, lo cual permitirá a los usuarios estudiar a su competencia o incluso provocaría problemas más críticos como no tener una forma de almacenar cuánto dinero había pujado un usuario por un jugador que ha perdido la puja.

De esta forma utilizaremos un servicio updateMarket³⁷, presente en la clase MarketService que vemos en la figura 3.45 y desglosado en su diagrama de secuencia en la figura 3.48. Este se ejecutará de manera periódica para cada una de las ligas, provocando la actualización del mercado de jugadores. El flujo es el siguiente:

1. Obtención de dos listas de jugadores, con y sin propietario.
2. Resolución de las pujas de esta tanda de mercado, asociando el jugador al usuario ganador y devolviendo el dinero a los usuarios perdedores.
3. Randomización de la lista de jugadores que no está en venta y establecimiento como jugadores en venta.
4. Establecimiento de la lista anterior de jugadores en venta que no han resuelto una puja como no en venta.

³⁵Código del modelo de mercado.

³⁶Código del modelo de pujas.

³⁷Código del servicio de mercado.

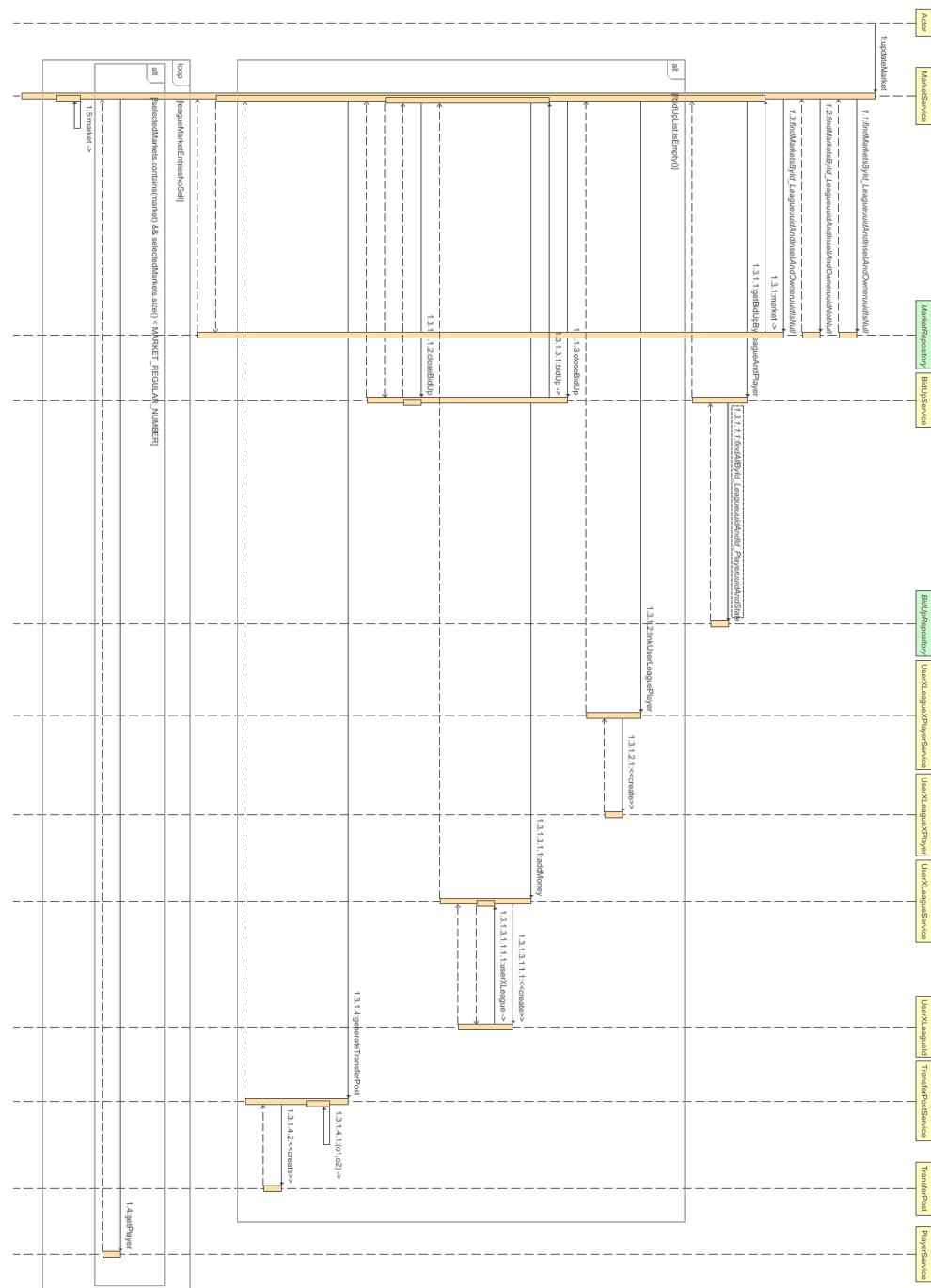


Figura 3.48: Diagrama de secuencia de la actualización de mercado

Además también tienen algunos procesos secundarios más como el relleno del historial de transferencias y pujas para mostrárselo a los usuarios una vez resuelto.

Para el frontend, se ha creado una pantalla (ver figuras 3.49 y 3.50) en la que aparecerá una lista con los diferentes jugadores, mostrando a los usuarios alguna información relevante de estos como el equipo, la posición o sus últimas rachas de puntuación, lo cual les ayudará a evaluar si deberían pujar o no por un jugador.

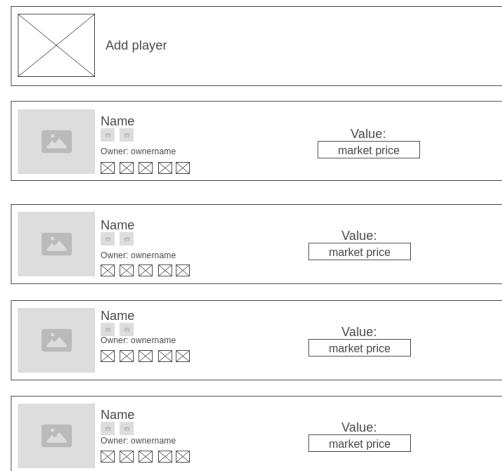


Figura 3.49: Wireframe pantalla de mercado.

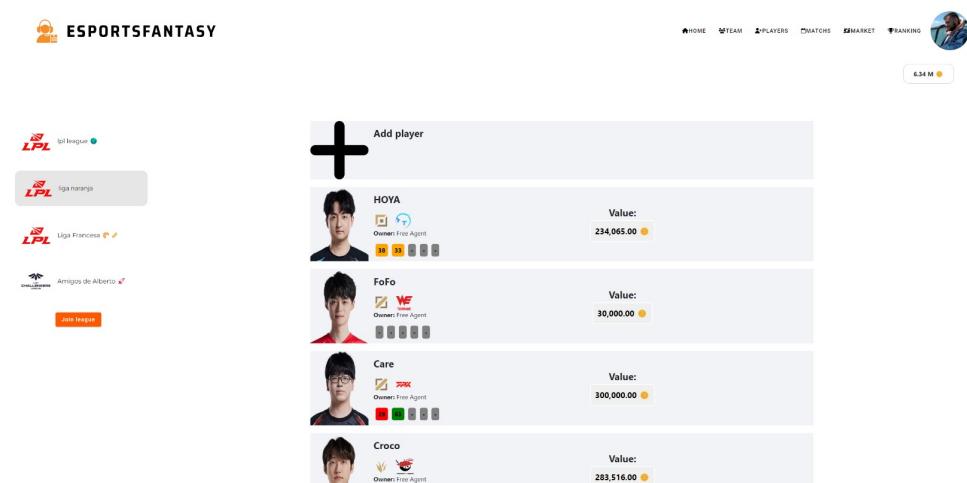


Figura 3.50: Mercado de jugadores.



Figura 3.51: Wireframe pantalla de venta de jugadores.

Para pujar por un usuario se ha creado una ventana modal, la cual mantiene la estética de la web e incluye tanto un deslizador, como una cantidad fraccionada por decimales, todo con el objetivo de facilitar la usabilidad y legibilidad de la puja para el usuario. Puede observar como se visualizaría una puja en las figuras 3.51 y 3.52.

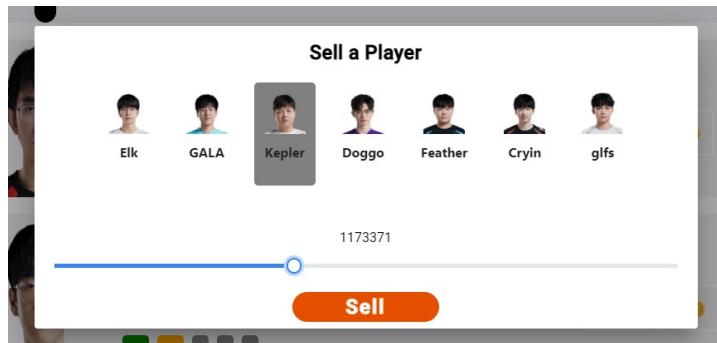


Figura 3.52: Modal de venta de un jugador.

Con respecto a colocar jugadores en venta, se realiza de forma muy sencilla simplemente modificando su entrada correspondiente en la tabla «Market» a en venta. Después, al obtener la lista de jugadores que están en venta, a los que no tienen dueño otorgados por la actualización de mercado, se le sumarán todos aquellos que los usuarios hayan puesto en venta. De la lista de pujas que se reciben, el usuario podrá escoger cual aceptar, las demás quedarán automáticamente invalidadas. En frontend esto se representará

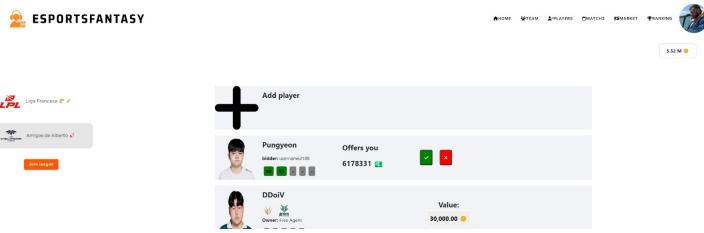


Figura 3.53: Oferta realizada a un jugador puesto en venta por otro usuario.

mostrando una entrada de mercado por cada una de las ofertas que recibe el jugador tal y como podemos ver en la imagen 3.53.

3.6.4. HU 3.4, 3.5: Clausulazos y aumentos de cláusula

Los clausulazos consisten en la transacción automática y sin necesidad de aceptación para un jugador de un usuario a otro. Estos pueden activarse o desactivarse además en la configuración de la liga. En caso de estar desactivados, simplemente se retirará todas las opciones pertinentes tanto de pago como de aumento de cláusula.

Cuando una persona realiza un clausulazo, internamente se procede a ejecutar un proceso de transacción explicado en el apartado anterior pero saltándose el paso de confirmación del otro usuario y en lugar de utilizando el valor de mercado, utilizando el valor de cláusula. Además cuando se ejecuta un clausulazo, se realiza un aumento de cláusula automáticamente (lo cual hace que si al persona a la que le has quitado el jugador, quiere recuperarlo, tenga que pagar más de lo que has pagado tú).

Con respecto al aumento de cláusula se ofrecen cuatro subidas posibles, 25 %, 33 %, 50 %, y 100 % del valor de cláusula. De manera cuanto mayor sea el aumento, menor será el coste proporcional. Es decir, a mayor dinero invertido, menor coste por unidad de cláusula aumentada. Puede observarlo de una manera más clara en figura 3.54 creada a partir de los datos de la tabla 3.4, en la que podemos observar como a medida que aumenta el coste total invertido, se reduce el espacio entre el coste proporcional y el incremento de cláusula.

Caso	Incremento	Cláusula Final	Coste Total	Coste Proporcional
1	25 %	125	25	5.00
2	33.33 %	133.33	37.33	3.57
3	50 %	150	63	2.38
4	100 %	200	150	1.33

Tabla 3.4: Tabla cláusula-coste para un ejemplo de cláusula de 100.

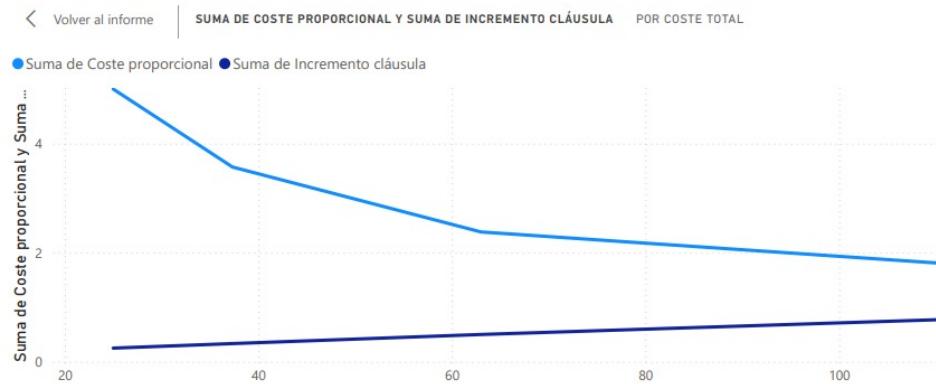


Figura 3.54: Gráfico de líneas relación coste proporcional / incremento de cláusula con respecto a coste total invertido.

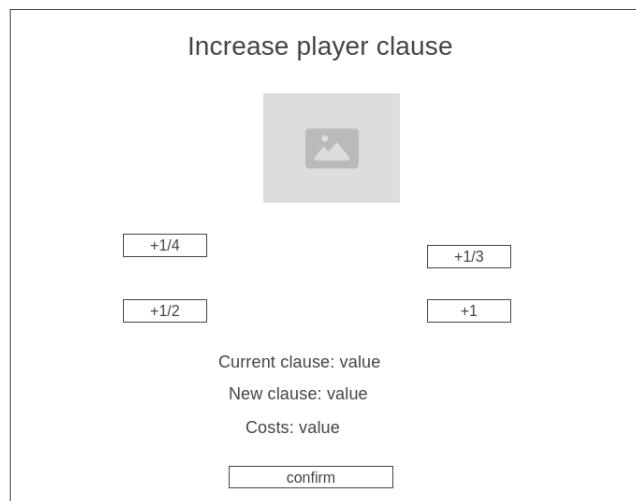


Figura 3.55: Wireframe de incremento de cláusula.

Respecto al front, podemos observar en las figuras 3.55 y 3.56, estos incrementos se han representado mediante un modal en el que se le ofrecen al usuario las opciones junto a un desglose de la información, lo que le permite visualizar los diferentes costes y proporciones de subida respecto a coste.

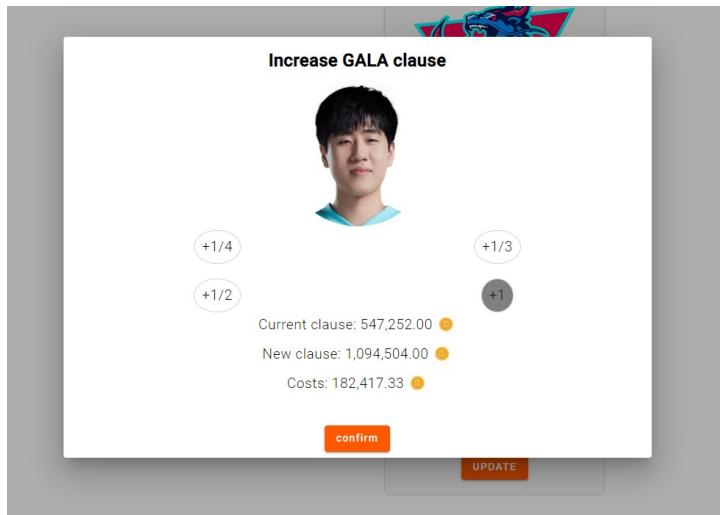


Figura 3.56: Modal de incremento de cláusula.

3.6.5. HU 3.6: Alineamiento de jugadores en equipos

Para el alineamiento de jugadores en los diferentes equipos de los usuarios, toma importancia el modelo de datos «UserXLeagueXPlayer»³⁸. En este modelo, concretamente lo que se almacena es si un jugador, de un determinado usuario que está asociado a una liga, está alineado en el equipo en cierta jornada.

Con esta información lo que conseguimos es, en definitiva, un historial de propiedad y posicionamiento sobre los jugadores. Permitiéndonos saber que jugador era propiedad de que usuario en cada momento. Además, añadiendo el atributo de alineación (el cual es un número representado por cada una de las posiciones posibles), podemos saber en qué rol decidió colocarlo el usuario, lo que nos permite que cada vez que entre a la pantalla de equipo, el usuario lo vea exactamente en la posición en la que lo dejó y además por saber que lo dejó alineado y no en el banquillo, atribuirle en el ranking los puntos correspondientes a la suma del historial de puntuaciones en los que tuvo determinados jugadores alineados.

Dado que la alineación de jugadores en un equipo es algo que realiza el usuario de manera manual (excepto en el caso de cambio de jornada, en este caso se actualiza la continuidad de la propiedad del usuario sobre el jugador y se le mantiene en la misma posición alineado para que el usuario no tenga que estar alineándolo de manera manual cada jornada), simplemente se asocia el jugador al usuario, liga y jornada en una determinada posición

³⁸Código del modelo UserXLeagueXPlayer.

mediante el modelo de datos «UserXLeagueXPlayer».

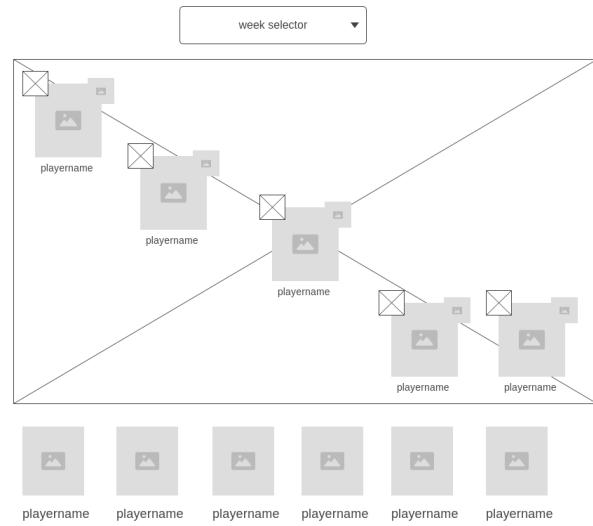


Figura 3.57: Wireframe de alineación de equipo de usuario.

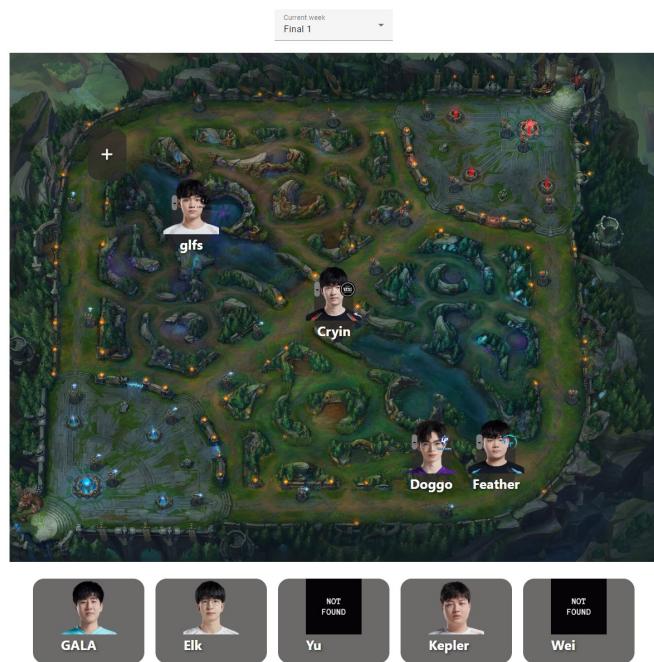


Figura 3.58: Alineación designada.

Con respecto al frontend, podemos ver en la figura 3.58, cómo se aporta dinamismo a la alineación de jugadores dibujando detrás un mapa que corresponda al videojuego de la liga, lo que aporta al usuario familiaridad a la hora de elegir la distribución de su equipo. El usuario podrá modificar estos jugadores siempre y cuando la jornada que quiere modificar no haya comenzado aún. En caso contrario, el usuario podrá ver la distribución de jugadores que había designado para jornadas anteriores seleccionadas, pero no modificarla para evitar así que afecte al cómputo de puntuación global.

3.6.6. HU 3.7: Consulta de historial

Para el apartado de consulta de historial, se introduce el modelo de datos «TransferPost»³⁹. Este modelo será un historial de transacciones, con campos como la fecha en la que tuvo lugar la transacción, quien era el anterior dueño, de qué jugador se trata e incluso la lista de usuarios que pujaron por el jugador.

El rellenado de esta entidad tiene lugar en tres momentos principales, cuando se resuelven las pujas en el «UpdateMarket», cuando se acepta una transacción o cuando tienen lugar un clausulazo.



Figura 3.59: Wireframe de pantalla de home de una liga.

Finalmente, este historial se muestra de forma conjunta para todos los usuarios de la liga, en orden de mayor a menor tiempo transcurrido desde el suceso, a modo de resumen de lo que ha sucedido en la liga. He decidido colocar este apartado en el menú home, el cual puede ver en las figuras 3.59 y 3.60, con la idea de mantener la familiaridad que tienen los usuarios que ya han sido jugadores de juegos fantasy anteriormente, pues esta es la posición en la que suele estar. Además considero que es importante, pues muchos

³⁹Código del modelo de datos TransferPost.

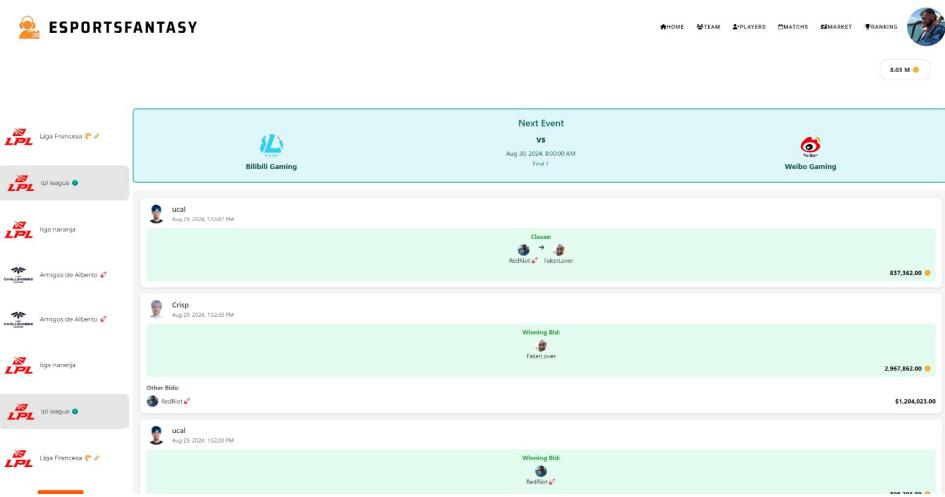


Figura 3.60: Home de una liga.

jugadores experimentados suelen utilizar este historial para llevar en mayor o menor medida un control del estado económico de los rivales.

3.6.7. Tests unitarios y de integración

Para este sprint se han realizado un total de 8 tests unitarios⁴⁰ y 5 tests de integración.

Desglose tests unitarios

Test	HU/HT	Descripción	Código de error	Resultado
testGeneralCascadeUpdate	5	Comprueba la recolección de datos en cascada.	-	✓
testUpdateMarket	8	Comprueba la gestión de aparición de jugadores en el mercado.	-	✓
testBidUpNoMarketEntry	3.1	Comprueba cuando no hay entradas de mercado para una puja.	1021	✓
testBidUp	3.1 - 3.2	Comprueba la gestión de pujas.	-	✓
testBidUpNoMarketInSell	3.1 - 3.2	Comprueba cuando no hay entradas de mercado en venta para una puja	1023	✓
testSetAligned	3.6	Comprueba la correcta alineación de un jugador en un equipo.	-	✓
testClausePlayer	3.4	Comprueba que el jugador es ahora propiedad del nuevo usuario.	-	✓
testUpdateClausePlayer	3.5	Comprueba que la cláusula de un jugador ha aumentado.	-	✓

Tabla 3.5: Listado test unitarios sprint 4

- **testGeneralCascadeUpdate:** Comprueba que la recolección de datos en cascada se realice correctamente.
- **testUpdateMarket:** Verifica que los jugadores aparezcan correctamente en el mercado de fichajes.

⁴⁰Código del test unitarios.

- **testBidUpNoMarketEntry:** Comprueba que el sistema gestione correctamente los casos en los que no hay entradas de mercado disponibles para una puja. Devuelve el error 1021 en caso de fallo.
- **testBidUp:** Valida la correcta gestión y actualización de una puja en el mercado.
- **testBidUpNoMarketInSell:** Verifica que el sistema maneje correctamente la falta de entradas de mercado en venta para una puja. Devuelve el error 1023 en caso de fallo.
- **testSetAligned:** Comprueba que un jugador esté correctamente aliñado en un equipo.
- **testClausePlayer:** Verifica que, después de una transacción, el jugador sea propiedad del nuevo usuario.
- **testUpdateClausePlayer:** Comprueba que la cláusula de un jugador se haya actualizado y aumentado correctamente.

Desglose tests integración

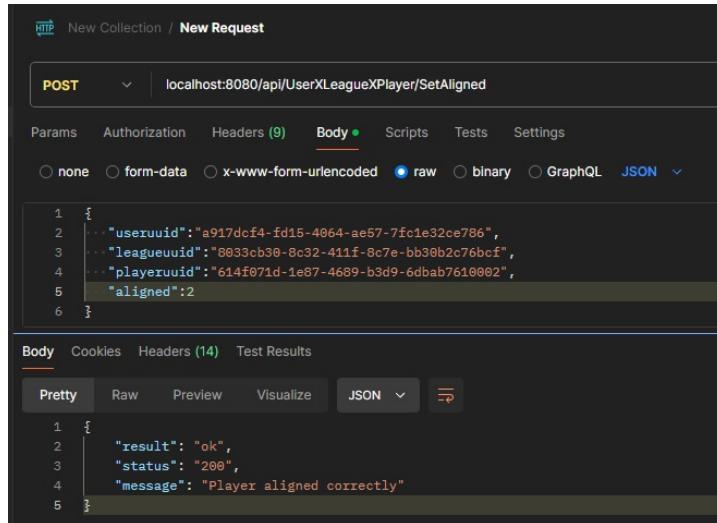


Figura 3.61: Petición a endpoint /setAligned con credenciales válidos.

The screenshot shows a Postman collection named "New Collection" with a "New Request". The request is a POST to "localhost:8080/api/UserXLeagueXPlayer/SetAligned". The "Body" tab is selected, showing raw JSON input:

```
1 {
2   ... "useruuid": "a917dcf4-fd15-4064-ae57-7fc1e32ce786",
3   ... "leagueuuid": "8033cb30-8c32-411f-8c7e-bb30b2c76bcf",
4   ... "playeruuid": "214f071d-1e87-4689-b3d9-6dbab7610002",
5   ... "aligned": 3
6 }
```

The response body is:

```
1 {
2   "result": "ko",
3   "appStatus": "1008",
4   "description": "Couldn't find a player."
5 }
```

Figura 3.62: Petición a endpoint /setAligned con credenciales inválidos.

The screenshot shows a Postman collection named "New Collection" with a "New Request". The request is a POST to "localhost:8080/api/Market/bidup". The "Body" tab is selected, showing raw JSON input:

```
1 {
2   ... "playeruuid": "5d8c2751-c4d7-4af3-ad10-68dfcd1b14a0",
3   ... "leagueuuid": "8033cb30-8c32-411f-8c7e-bb30b2c76bcf",
4   ... "useruuid": "a917dcf4-fd15-4064-ae57-7fc1e32ce786",
5   ... "value": 300000
6 }
```

The response body is:

```
1 {
2   "result": "ok",
3   "status": "201",
4   "message": "bidup created correctly"
5 }
```

Figura 3.63: Petición a endpoint /bidup con credenciales válidos.

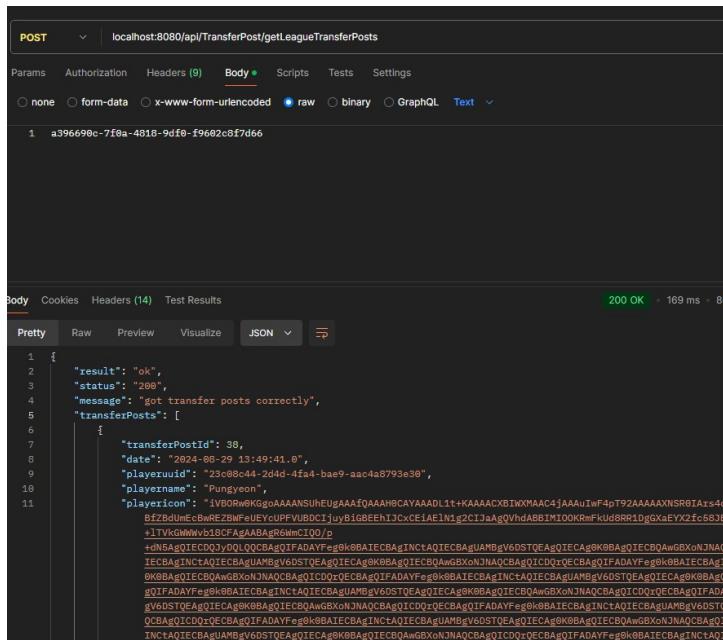


Figura 3.65: Petición a endpoint /getLeagueTransferPosts

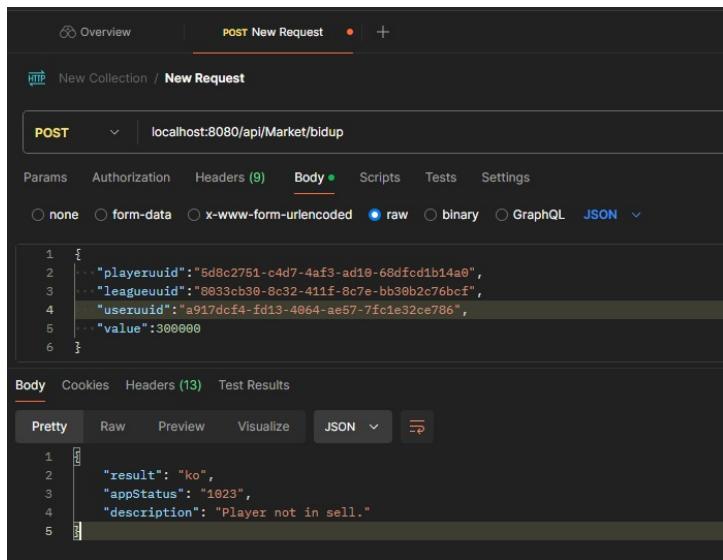


Figura 3.64: Petición a endpoint /bidup con credenciales inválidos.

Equivalencias HU/HT a pruebas de aceptación

- **HT 5** Se debe verificar el correcto llenado de la base de datos en el proceso de recolección de datos.
 - **TU:** testGeneralCascadeUpdate.

- **HT 8:** Comprueba que se ofrecen jugadores para pujar y estos no son propiedad de ningún jugador.
 - **TU:** testUpdateMarket.
- **HU 3.1 y 3.2:** Se debe verificar que el jugador es ahora propiedad del usuario comprador.
 - **TU:** testBidUpNoMarketEntry.
 - **TU:** testBidup.
 - **TU:** testBidupNoMarketInSell.
- **HU 3.4 y 3.5:** Se debe comprobar que un jugador ha sido clausulado / aumentado su cláusula.
 - **TU:** testClausePlayer.
 - **TU:** testUpdateClausePlayer.
- **HU 3.6:** Se debe verificar que un jugador ha sido correctamente aliñeadado.
 - **TU:** testSetAligned.
 - **TI:** Petición a /setAligned con credenciales válidos. Ver figura 3.61.
 - **TI:** Petición a /setAligned con credenciales inválidos. Ver figura 3.62.
- **HU 3.7:** Se debe verificar que se devuelve correctamente un historial de transacciones.
 - **TI:** Petición a /getLeagueTransferPosts. Ver figura 3.65.

3.6.8. Retrospectiva y retrasos

La principal problemática durante este sprint fue la recolección de datos. El videojuego del que trato de obtener datos (League of Legends), no cuenta con fuentes oficiales de obtención de datos para las partidas competitivas profesionales. Las APIs que encontraba por la red o no tenían información suficiente como para poder desarrollar la aplicación (principalmente porque les faltaban datos del desempeño de los jugadores con los que poder crear una heurística de puntuación realista), o en su defecto, tenían precios muy abusivos por encima de los 500€ mensuales.

Tras una larga investigación de semanas, encontré la extensión «Cargo» [Ext24] de «MediaWiki» [Med24], la cual me permitía obtener información

de sus páginas derivadas, por lo que de esta manera pude acceder a la información de «Leaguepedia» [Lea24] . Sin embargo la recolección de datos, siguió siendo una tarea complicada, no solo por tener que aprender a realizar las «cargoQueries», si no porque el diseño de las tablas es confuso, con tablas repetidas y redundantes, poca cohesión y alto acoplamiento entre ellas y con una documentación oficial bastante vaga llegando a encontrarme descripciones como “no se que es este campo, creo que es la clave primaria, yo no lo uso” o incluso preguntas que se lanzan entre los propios administradores sobre los campos como “¿El estatus resuelve el estado del rumor?”; textos, en general, completamente descontextualizados e incomprensibles para los desarrolladores.

Estas dificultades supusieron un retraso de casi dos meses en el desarrollo de este sprint.

3.7. Sprint 5

3.7.1. Introducción al Sprint 5

A lo largo del quinto sprint se desarrollará el sistema de puntuación de los jugadores. Normalmente, los juegos del entorno «fantasy» no suelen llevar implementados sus propios sistemas de puntuación, sin embargo, encontrar un lugar que nos otorgue el permiso legal para obtener sus puntuaciones sobre el desempeño de los jugadores ha sido imposible, por consiguiente, ha sido necesario desarrollar una heurística de puntuación propia. Lo mismo ha sucedido con las valoraciones de mercado de los jugadores, para la que también hemos creado una heurística propia.

Una heurística es, en definitiva, una manera rápida y eficiente de resolver un problema tomando como base experiencias, experimentación o reglas generales, sin un análisis completo ni profundo del tema pero tratando de acercarnos a la mejor solución posible [Pel85].

En la figura 3.66 podemos hacernos una idea general de la estructura del sprint, la cual iremos desglosando a lo largo del mismo.

3.7.2. HT 6: sistema de ponderación de puntuaciones de desempeño

El principal modelo de datos introducido a lo largo de este sprint es «PlayerPoits»⁴¹ el cual utilizaré para asociar puntos a jugadores en cada

⁴¹Código del modelo de puntuación de jugador.



Figura 3.66: Diagrama UML del sprint 5

uno de los partidos que disputa.

Aprovechando la obtención programada de resultados de partidos, se obtendrán también las estadísticas de cada uno de los jugadores en los diferentes partidos para los sistemas sin sistema de puntuación externo. Para cada juego que se implemente como posibilidad en la aplicación, será necesario o realizar una nueva heurística de puntuaciones basada específicamente en las estadísticas que aporte cada juego, u obtener los datos de puntuación mediante otra fuente de datos.

Para crear esta heurística⁴² me he apoyado en algunos de mis compañeros de universidad con nivel semi-profesional en el juego, para poder ponderar bien la importancia que tienen las diferentes estadísticas que obtengo para cada uno de los roles en el videojuego. A continuación comentaré brevemente la heurística de puntuación que se ha diseñado para el primer juego implementado en la aplicación (League of Legends):

Dado que dependiendo de la posición que se está evaluando se deben priorizar unos u otros objetivos, comentaremos las principales diferencias en las puntuaciones:

- **Asesinatos:** Valores superiores para las posiciones de jungla, mid y bot, dada la importancia de que estas posiciones asesinen enemigos. Inferior en el caso del top y el support, los cuales tienen otro tipo de prioridades como dar soporte o destruir objetivos.
- **Asistencias:** Valores superiores para el support, el cual debe asistir lo máximo posible para apoyar las acciones de asesinato de su equipo.
- **Minions (CS):** Mayores ratios de puntuación para el top, cuya prioridad son los objetivos y el asesinato de minions, ratios inferiores para los demás, que deben priorizar la búsqueda de asesinatos.
- **Puntos de visión:** Mayor ratio de puntuación para el support, el cual debe mantener una buena visibilidad de mapa para sus aliados a lo largo de la partida.
- **Oro obtenido:** Menores ratios de oro obtenido requeridos para soportes, los cuales obtienen de media mucho menos dinero, con el objetivo de no des-balancear entre posiciones.
- **Daño realizado a campeones:** Igual de escalable para cada posición, pues dependerá de la composición de equipo de ese partido.

⁴²Código de la heurística.

- **Proporciones con respecto a equipo:** Cálculo de la proporción de asesinatos y oro total con respecto a tu equipo. Esto premiará a los jugadores que han destacado en participación dentro de su propio equipo.

3.7.3. HT 7: Heurística de valores de mercado

Para crear esta heurística⁴³ he decidido que el punto angular debería ser las puntuaciones de los jugadores. De este modo, se calcula una mediana (con el objetivo de no desbalancear los precios por culpa de jugadores con muy grandes o muy bajas puntuaciones fuera de la norma) de puntuación total para los jugadores de una liga real, siempre ignorando los jugadores que no han puntuado por no haber jugado. Partiendo de un precio de valor por defecto para todos los jugadores, se calculará un índice de recompensa como la división de los puntos totales de un jugador entre la mediana de puntos totales en la liga a la que pertenece. Calculamos un valor de jugador multiplicando el precio por defecto por esa puntuación, haciendo así el valor del jugador directamente proporcional a su desempeño.

Sin embargo, también se ha llegado a la conclusión de que depender de las puntuaciones de los jugadores como única fuente de información puede resultar escaso, mientras que un buen punto de referencia podría ser los jugadores por los que pujan los usuarios, por lo que también se ha obtenido de nuevo una mediana, pero esta vez de pujas realizadas a los jugadores. De este modo, ponderará también la cantidad de pujas que un jugador recibe.

3.7.4. Tests unitarios y de integración

desglose test unitarios

Para este sprint se ha realizado un total de 6 tests unitarios⁴⁴.

Test	HU	Descripción	Código de error	Resultado
testCalculateLOLPoints_EmptyGames	6	Comprueba el correcto cálculo de puntuación para el juego vacío.	-	✓
testCalculateLOLPoints_TopRole	6	Comprueba el correcto cálculo de puntuación para el rol Top.	-	✓
testCalculateLOLPoints_JungleRole	6	Comprueba el correcto cálculo de puntuación para el rol Jungla.	-	✓
testCalculateLOLPoints_MidRole	6	Comprueba el correcto cálculo de puntuación para el rol Mid.	-	✓
testCalculateLOLPoints_BotRole	6	Comprueba el correcto cálculo de puntuación para el rol Bot.	-	✓
testCalculateLOLPoints_SupportRole	6	Comprueba el correcto cálculo de puntuación para el rol Support.	-	✓
testPlayerValMarket	7	Comprueba el correcto cálculo de puntuación para el rol Support.	-	✓

Tabla 3.6: Listado test unitarios sprint 5.

- **testCalculateLOLPoints_EmptyGames:** Verifica el correcto cálculo de puntuación para el caso de un juego vacío.

⁴³Código con la heurística de valor de mercado.

⁴⁴Código de los tests para los servicios de puntuación de jugadores.

- **testCalculateLOLPoints_TopRole:** Comprueba que el cálculo de puntuación sea correcto para jugadores que desempeñan el rol de Top.
- **testCalculateLOLPoints_JungleRole:** Valida el cálculo de puntuación para el rol de Jungla.
- **testCalculateLOLPoints_MidRole:** Verifica el cálculo de puntuación correcto para jugadores en el rol de Mid.
- **testCalculateLOLPoints_BotRole:** Asegura que el cálculo de puntuación sea adecuado para el rol de Bot.
- **testCalculateLOLPoints_SupportRole:** Comprueba el cálculo de puntuación para el rol de Support.
- **testPlayerValMarket:** Verifica el correcto cálculo de puntuación para el rol de Support dentro del mercado.

Equivalencias HU/HT a pruebas de aceptación

- **HT 6:** Se debe verificar que los jugadores obtienen puntuaciones acordes a la heurística.
 - **TU:** testCalculateLOLPointsEmptyGames
 - **TU:** testCalculateLOLPointsTopRole
 - **TU:** testCalculateLOLPointJungleRole
 - **TU:** testCalculateLOLPointMidRole
 - **TU:** testCalculateLOLPointBotRole
 - **TU:** testCalculateLOLPointSupportRole
- **HT 7:** Se debe verificar que los jugadores obtienen valoraciones de mercado acordes a la heurística.
 - **TU:** testPlayerValMarket.

3.7.5. Retrospectiva y retrasos

No hubo apenas dificultades a lo largo de este sprint, por lo que se cumplió dentro del tiempo previsto e incluso se completó varios días antes de lo esperado.

3.8. Sprint 6

3.8.1. Introducción al sprint 6

El sprint 6 será el último en realizarse. En este se realizarán múltiples tareas, principalmente de frontend en el que se mapearán y dará estilo a los componentes de la aplicación que no son interactuables si no simplemente visibles ya que otorgan información ajena a los usuarios.

3.8.2. HU 4.1 y 4.2 : Mapeo y muestreo de datos en el frontend.

Pantalla de partidos

Para la pantalla de partidos, comprobaremos antes si se trata de un usuario que ha iniciado sesión. En caso de que así sea, se obtendrá la lista de partidos correspondiente a la liga que el usuario tenga seleccionada y se mostrarán los resultados junto a algunos datos del partido si es que los tiene. En caso de que se trate de un usuario no registrado, se pasará antes por una pantalla de selección de juego y liga, para evitar así sobrecargar la aplicación del cliente con peticiones y demasiada información por parte del servidor. Puede ver el escaparate de partidos en las figuras 3.67 y 3.68



Figura 3.67: Wireframe de escaparate de partidos.

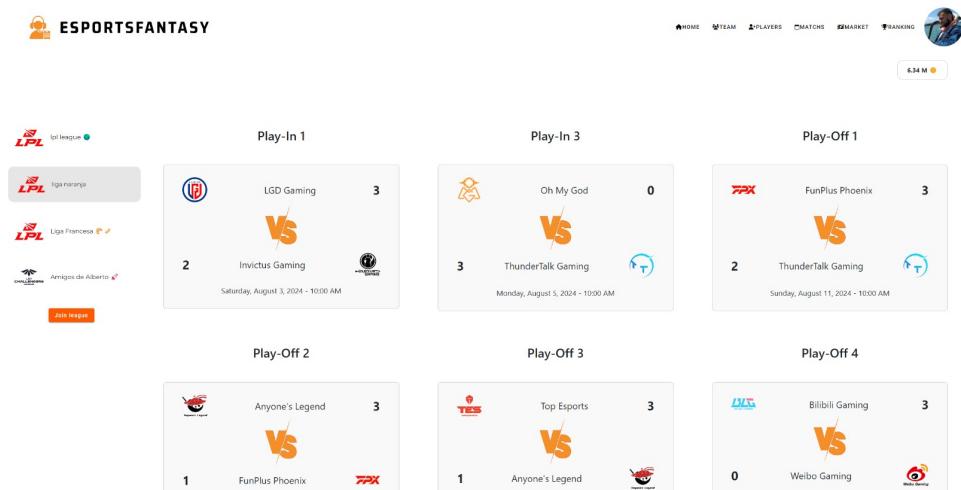


Figura 3.68: Escaparate de partidos.

Pantalla de jugadores y equipos

Para la pantalla de jugadores, procederemos de la misma forma que en la pantalla de partidos con el objetivo de mantener coherencia entre las diferentes pantallas.

Para los usuarios con la sesión iniciada, se mostrará directamente una lista de los jugadores pertenecientes a los equipos de la liga seleccionada por el usuario, (ver 3.71 y 3.72). Mientras que para los usuarios sin sesión iniciada (usuarios espectadores), se mostrará la misma pantalla de selección de juego, liga y adicionalmente para esta pantalla, equipo y jugador, que se utiliza en la creación de una liga, de nuevo con el objetivo de no sobrecargar al cliente con peticiones e información. Una vez el usuario espectador haya seleccionado el recurso que quiere ver, se utilizarán las pantallas de información de equipo o información incrustadas bajo el acordeón de selección (ver 3.69 y 3.70)

Además, en la pantalla de lista de jugadores se han incluido algunos filtros, que te permiten diferenciar los datos por cantidad de puntos, precio de mercado, equipos y posiciones.

Para cada uno de los jugadores, se guarda también información a la que se puede acceder haciendo click en el mismo, dejándonos ver puntuaciones, desempeño en los últimos partidos, etc (ver 3.73 y 3.74).

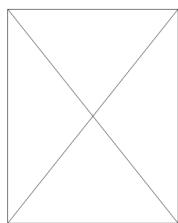
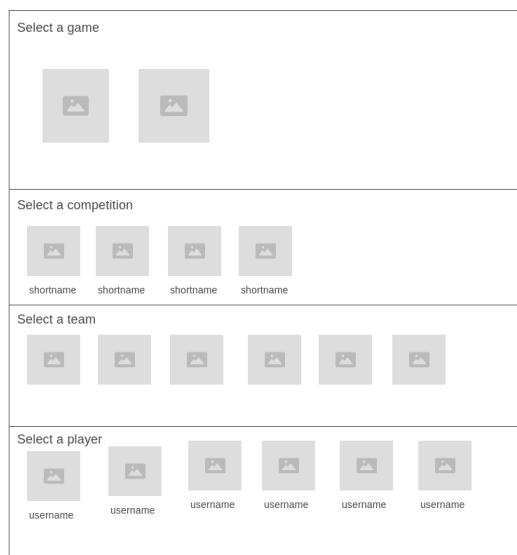


Figura 3.69: Wireframe de visualización de jugador para usuario espectador.

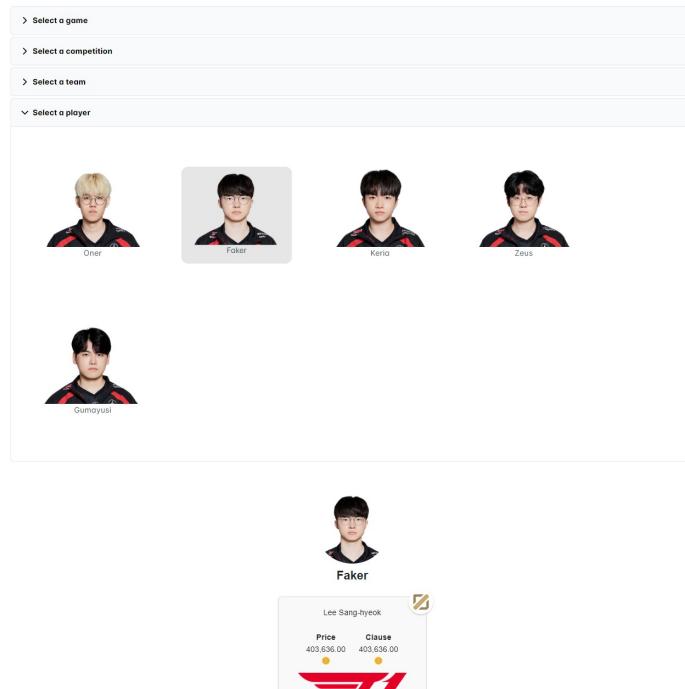


Figura 3.70: Pantalla de visualización de jugador para usuario espectador.

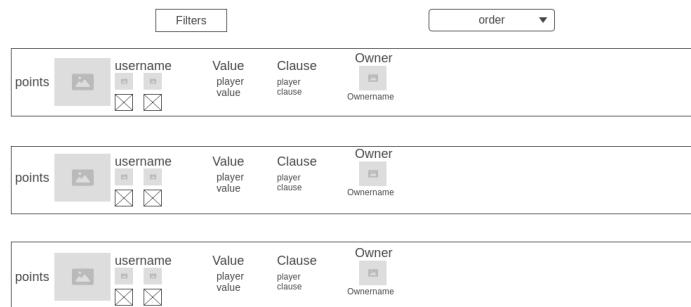


Figura 3.71: Wireframe de pantalla con lista de jugadores.

	Player Name	Price	value	clause	Owner
127	ucal	418681.00	837.362.00	●	No Owner
124	Feather	408791.00	817.582.00	●	RedSot
119	Elk	392307.00	784.514.00	●	RedSot
114	Breathe	375524.00	751.488.00	●	No Owner
106	Shanks	340450.00	698.390.00	●	No Owner

Figura 3.72: Listado de jugadores.

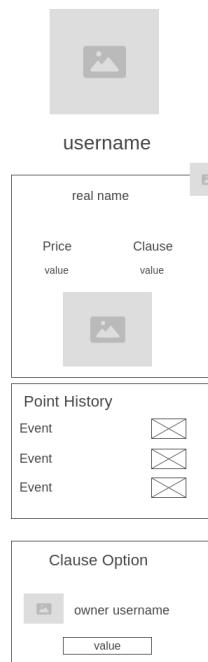
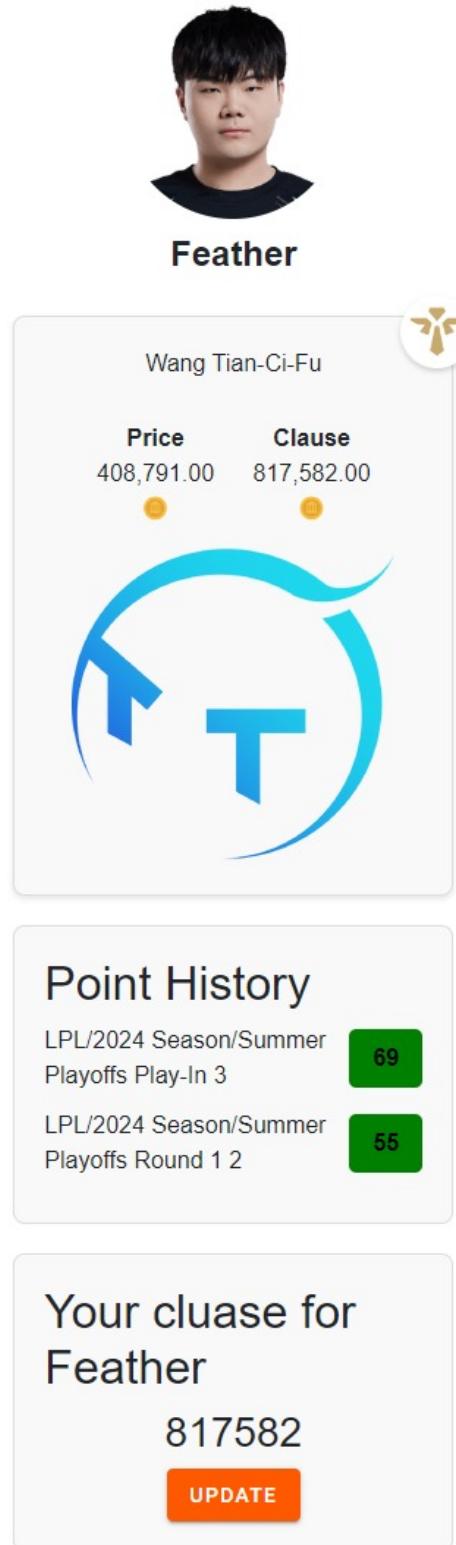


Figura 3.73: Wireframe pantalla de información de jugador.



A detailed player profile card for Wang Tian-Ci-Fu, also known as Feather. The card features a portrait of the player at the top, followed by his name "Feather". Below the name is a circular icon containing a stylized golden feather logo. The card is divided into several sections: "Price" (408,791.00) and "Clause" (817,582.00), both accompanied by small orange circular icons. A large blue circular arrow graphic with a central "T" is prominently displayed. The "Point History" section shows two entries: "LPL/2024 Season/Summer Playoffs Play-In 3" with a value of 69, and "LPL/2024 Season/Summer Playoffs Round 1 2" with a value of 55. The final section, "Your clause for Feather", displays the clause value of 817582 and includes an "UPDATE" button.

Wang Tian-Ci-Fu

Price Clause

408,791.00 817,582.00

Point History

LPL/2024 Season/Summer Playoffs Play-In 3 69

LPL/2024 Season/Summer Playoffs Round 1 2 55

Your clause for Feather

817582

UPDATE

Figura 3.74: Pantalla de información de un jugador en propiedad.

3.8.3. Tests unitarios y de integración

Dado que en este sprint no se han creado servicios y solo se han reutilizado funcionalidades anteriores para crear nuevas pantallas en el lado del cliente, consideraremos como las pruebas de aceptación el resultado de las pantallas que podemos ver en las figuras que se han mostrado a lo largo de todo este sprint.

3.8.4. Retrospectiva y retrasos

Dado que en este sprint ya se contaba con toda la funcionalidad backend necesaria para obtener los datos que se requería, no ha habido que crear ningún servicio, por lo que ha resultado en un trabajo bastante ágil de maquetación. Dado que no ha habido ningún retraso y se terminó antes de la fecha límite, se aprovechó para resolver errores arrastrados o descubiertos a posteriori correspondientes a los sprints anteriores y mejorar algunas secciones de usabilidad y estética general de la web.

Capítulo 4

Conclusiones y trabajos futuros

4.1. Emprendimiento y visión comercial

eSportsFantasy nació como consecuencia de un vacío en la explotación comercial de los videojuegos del género fantasy aplicados a los deportes electrónicos. Esta nueva modalidad competitiva se trata sin lugar a dudas de un sector en constante crecimiento y expansión, destacando especialmente entre los públicos más jóvenes. La estructura altamente escalable a múltiples disciplinas de eSports que tiene la aplicación, le aporta un factor adaptativo muy importante en un medio tan veloz y cambiante, lo que podría servir no solo para colocarse como uno de los pioneros en explotar el medio, si no para mantenerse en un puesto de gran importancia a lo largo del tiempo.

Este será un videojuego de descarga gratuita, sin embargo esto no lo convierte en una mala perspectiva de negocio, pues hoy en día, los videojuegos que más dinero facturan, son videojuegos gratuitos [ads24]. Incluir micro-transacciones en los videojuegos gratuitos está a la orden del día. Una posible micro-transacción a incluir en el videojuego podría ser, por ejemplo, monedas de pago con las que poder aumentar la cláusula de sus jugadores, y obtener así ventaja con respecto a sus enemigos.

Esta implementación de micro-transacción se podría complementar con múltiples estrategias de marketing, como realizar promociones de estas monedas, en momentos clave de las diferentes competiciones como fichajes de grandes estrellas o grandes rachas de puntos. Estos momentos clave provocan inseguridad a los usuarios con respecto a posibles clausulazos de sus compañeros de liga, lo que sumado a una promoción o descuento en la aplicación, podría conllevar a una compra con el objetivo de evitar que sus comunes le roben a su jugador máspreciado.

4.2. Conclusiones

Como conclusión final del trabajo de Fin de Grado, se comenzará por destacar como se han cumplido todos los objetivos básicos del proyecto expuestos en el apartado 1.2.1. Se ha creado una aplicación web de escritorio, la cual cuenta con una alta escalabilidad a los diferentes eSports o competiciones que estén de moda en cada momento. Que además cuenta con un sistema dinámico de mercado ajeno e independiente de la disciplina de deporte electrónico y que en adición cuenta con un buen acoplamiento de sistemas de puntuación, ya sean externos o creados por el desarrollador de la propia aplicación.

Se ha obtenido un producto innovador que no existía antes en el mercado como es un juego fantasy multidisciplinar, pero manteniendo la esencia clásica de la gestión de recursos que tanto engancha a los jugadores de este género.

4.3. Posibles trabajos futuros

Si bien la aplicación es considerablemente grande, queda abierta a un sin fin de mejoras o ampliaciones, entre las que podemos destacar:

- **Inclusión de nuevos videojuegos:** Destacados en deportes electrónicos como CSGO, Overwatch o Valorant.
- **Adaptación responsive:** Modelado responsive del maquetado de la aplicación que se adapte a múltiples resoluciones de pantalla como tablets o móviles.
- **Generación APK:** Si bien la aplicación se ha construido a modo de PWA, una vez se haya realizado la adaptación responsive, generar APK para poder contar con la web en formato de APK es trabajo fácil gracias a herramientas como Cordova.
- **Nuevas modalidades:** Si bien el juego está pensado para deportes electrónicos, la normalización de fuentes de datos podría ayudar a incluir nuevas modalidades, como por ejemplo un modo “competitivas”, en el cual se obtengan en lugar de los datos de equipos de las ligas profesionales, los de los jugadores que juegan a los modos competitivos y están en las ligas más alta de estos.

4.4. Reflexión personal

Como reflexión personal, he de destacar la magnitud del proyecto, la cual personalmente menosprecié en un primer lugar cuando tuve la idea de

llevar a cabo este Trabajo de Fin de Grado. Resultando en un total de +99 clases propias, +50 archivos de maquetación entre html y css, 14 tablas en bases de datos y más de 40.000 lineas de código. Este proyecto ha supuesto para mi un auténtico golpe de realidad, en el que me he dado cuenta de las magnitudes que alcanzan los proyectos reales, además de todos los conceptos que creía afianzados y he tenido que volver a estudiar y revisar para poder llevarlos acabo. Los conocimientos adquiridos a lo largo de la carrera han sido fundamentales, desde principios de diseño de arquitectura, planificación software y diseño de bases de datos hasta llegar a rescatar conocimientos de asignaturas de más bajo nivel, como aplicación del uso de máscaras a nivel de bit o diseño de redes locales. También ha sido especialmente relevante mi propio aprendizaje fuera de la carrera, el cual sustentado en los pilares que se me han otorgada, ha sido más fácil de llevar, estudiante nuevas tecnologías que están a la orden del día como AngularJS, Java Springboot y docker.

Bibliografía

- [02] «Comunio». Último acceso: 2 de junio de 2024. 2002. URL: <https://www.comunio.es>.
- [16] «Marca Fantasy». Último acceso: 2 de junio de 2024. 2016. URL: <https://fantasy.marca.com/>.
- [19] «Mister mundo deportivo». Último acceso: 2 de junio de 2024. 2019. URL: <https://mister.mundodeportivo.com/>.
- [20a] «E-go Fantasy». Último acceso: 2 de junio de 2024. 2020. URL: <https://play.egoapp.gg>.
- [20b] «Super fantasy LOL». Último acceso: 2 de junio de 2024. 2020. URL: <https://superfantasylol.com>.
- [ads24] adslzone. «Juegos que más facturan». Último acceso: 27 de agosto de 2024. 2024. URL: <https://www.adslzone.net/reportajes/videojuegos/juegos-mas-dinero-generan/>.
- [Ang23] Angular. «Introducción a servicios e inyección de dependencias». Último acceso: 27 de agosto de 2024. 2023. URL: <https://docs.angular.lat/guide/architecture-services>.
- [Ang24] Angular. «Angular official documentation». Último acceso: 27 de agosto de 2024. 2024. URL: <https://angular.io/docs>.
- [Atl24] Atlassian. «Velocity Scrum». Último acceso: 26 de agosto de 2024. 2024. URL: <https://www.atlassian.com/es/agile/project-management/velocity-scrum>.
- [bae24] baeldung. «Clean Architecture with Spring Boot». Último acceso: 27 de agosto de 2024. 2024. URL: <https://www.baeldung.com/spring-boot-clean-architecture>.
- [Bel23] Gbadebo Bello. «What is JWT?». Último acceso: 27 de agosto de 2024. 2023. URL: <https://blog.postman.com/what-is-jwt>.
- [Doc24] Docker. «What is a container?». Último acceso: 27 de agosto de 2024. 2024. URL: <https://www.docker.com/resources/what-container>.

- [est24a] estalent. «Salario medio de un Administrador base de datos». Último acceso: 27 de agosto de 2024. 2024. URL: <https://es.talent.com/salary?job=administrador+base+de+datos>.
- [est24b] estalent. «Salario medio de un desarrollador Angular». Último acceso: 27 de agosto de 2024. 2024. URL: <https://es.talent.com/salary?job=programador+angular>.
- [est24c] estalent. «Salario medio de un desarrollador java». Último acceso: 27 de agosto de 2024. 2024. URL: <https://es.talent.com/salary?job=programador+java>.
- [est24d] estalent. «Salario medio de un devops». Último acceso: 27 de agosto de 2024. 2024. URL: <https://es.talent.com/salary?job=devops>.
- [est24e] estalent. «Salario medio de un maquetador web». Último acceso: 27 de agosto de 2024. 2024. URL: <https://es.talent.com/salary?job=maquetador>.
- [Ext24] MediaWiki Cargo Extension. «Cargo extension official page». Último acceso: 23 de julio de 2024. 2024. URL: <https://www.mediawiki.org/wiki/Extension:Cargo>.
- [Flu24] Flutter. «Flutter official documentation». Último acceso: 27 de agosto de 2024. 2024. URL: <https://docs.flutter.dev>.
- [Ger12] Joana Geraldí. «Gantt charts revisited: A critical analysis of its roots and implications to the management of projects today». En: *International Journal of Managing Projects in Business* (2012). URL: https://www.researchgate.net/publication/263571905_Gantt_charts_revisited_A_critical_analysis_of_its_roots_and_implications_to_the_management_of_projects_today.
- [Goo24a] Google. «Google token verifier». Último acceso: 21 de julio de 2024. 2024. URL: <https://cloud.google.com/java/docs/reference/google-api-client/latest/com.google.api.client.googleapis.auth.oauth2.GoogleIdTokenVerifier>.
- [Goo24b] Google. «Precios Compute engine». Último acceso: 27 de agosto de 2024. 2024. URL: <https://cloud.google.com/compute/all-pricing?hl=es-419>.
- [Hak05] Maurizio Morisio Hakan Erdogmus Marco Torchiano. «On the effectiveness of the test-first approach to programming». En: *IEEE Transactions on Software Engineering* (2005). Último acceso: 27 de agosto de 2024. URL: https://www.researchgate.net/publication/3188484_On_the_effectiveness_of_the_test-first_approach_to_programming/link/0046351aef3d1d6b0d000000/download?tp=eyJjb250ZXh0Ijp7ImZpcnN0UGFnZSI6InB1YmzpY2F0aW9uLiwcicGFnZ

- [Hos24] Hostinger. «Presupuesto hostinger». Último acceso: 27 de agosto de 2024. 2024. URL: https://www.hostinger.es/?utm_campaign=Brand-Exact%7CNT:Se%7CLO:ES&utm_medium=ppc&gad_source=1&gclid=Cj0KCQjw28W2BhC7ARIsAPerrcKkjamj-QQbW3z9WjDiWe7rz-6k_vo2C1dzJCGB6kE1F7LXbP_q1-oaAh0VEALw-wcB.
- [Jul20] Stephanny Aguirre Barrera Julieth Aguirre Barrera. «Metodologías para el desarrollo de Proyectos». En: (2020). Último acceso: 8 de abril de 2024.
- [Kin24] Kinsta. «Cuota de mercado de PHP». Último acceso: 27 de agosto de 2024. 2024. URL: <https://kinsta.com/es/cuota-mercado-php/#:~:text=PHP%20es%20el%20lenguaje%20de,debido%20a%20su%20uso%20generalizado..>
- [Lea24] Leaguepedia. «Leaguepedia official page». Último acceso: 25 de julio de 2024. 2024. URL: https://lol.fandom.com/wiki/League_of_Legends_Esports_Wiki.
- [Med24] MediaWiki. «MediaWiki official page». Último acceso: 23 de julio de 2024. 2024. URL: <https://www.mediawiki.org/wiki/MediaWiki/es>.
- [Mon24] MongoDB. «MongoDB official documentation». Último acceso: 27 de agosto de 2024. 2024. URL: <https://www.mongodb.com/docs/>.
- [Nod24] NodeJS. «NodeJS official documentation». Último acceso: 27 de agosto de 2024. 2024. URL: <https://nodejs.org/docs/latest/api/>.
- [Ora24] Oracle. «MySQL official documentation». Último acceso: 27 de agosto de 2024. 2024. URL: <https://dev.mysql.com/doc/>.
- [Pel85] Francis Jeffry Pelletier. «What is an heuristic?» En: *Computational Intelligence* (1985). URL: https://www.researchgate.net/publication/227733871_What_is_a_heuristic/link/5a0a0fe0aca272d40f4122ae/download?tp=eyJjb250ZXh0Ijp7ImZpcnN0UGFnZSI6InB1Ymxp
- [Php24] Php. «Php official documentation». Último acceso: 27 de agosto de 2024. 2024. URL: <https://www.php.net/docs.php>.
- [Pit24] Sten Pittet. «Types of software testing». Último acceso: 27 de agosto de 2024. 2024. URL: <https://www.atlassian.com/es/continuous-delivery/software-testing/types-of-software-testing>.
- [Pos24] Postman. «Test API integrations and data flow in Postman». Último acceso: 27 de agosto de 2024. 2024. URL: <https://learning.postman.com/docs/tests-and-scripts/test-apis/integration-testing/>.

- [rsa20] janpier rsanchez. «Nx Monorepo múltiples proyectos con Angular». Último acceso: 27 de agosto de 2024. 2020. URL: <https://janpierrsanchez.medium.com/nx-monorepo-angular-e6bd090b7e1a#:~:text=Nx%20es%20el%20kit%20de,stack%20con%20su%20marco%20preferido..>
- [Spr24a] Spring. «Spring data JPA». Último acceso: 27 de agosto de 2024. 2024. URL: <https://spring.io/projects/spring-data-jpa>.
- [Spr24b] SpringBoot. «SpringBoot official documentation». Último acceso: 27 de agosto de 2024. 2024. URL: <https://docs.spring.io/spring-boot/index.html>.
- [Vue24] VueJS. «VueJS official documentation». Último acceso: 27 de agosto de 2024. 2024. URL: <https://vuejs.org/guide/introduction.html>.