Memoria del proyecto

Dream NBA: aplicación para la creación de ligas virtuales y gestión de un equipo propio

Trabajo de Fin de Grado

INGENIERÍA INFORMÁTICA

VNiVERSiDAD D SALAMANCA

CAMPUS DE EXCELENCIA INTERNACIONAL

Julio 2022

Autor

Jairo Sánchez García

Tutores

Gabriel Villarrubia González

Juan Francisco de Paz Santana

Héctor Sánchez San Blas

CERTIFICADO DE LOS TUTORES

D. Héctor Sánchez San Blas, D. Juan Francisco de Paz Santana y D. Gabriel Villarrubia González, profesores del Departamento de Informática y Automática de la Universidad de Salamanca.

HACEN CONSTAR:

Que el trabajo titulado “DreamNBA: aplicación para la creación de ligas virtuales y gestión de un equipo propio” ha sido realizado por D. Jairo Sánchez García, con número de documento 70917967C y constituye la memoria del trabajo realizado para la superación de la asignatura Trabajo de Fin de Grado de la Titulación Grado en Ingeniería Informática de la Universidad de Salamanca.

Y para que así conste a todos los efectos oportunos.

En Salamanca, a 4 de julio de 2022.

SANCHEZ SAN BLAS HECTOR - 70901148Z
Digitally signed by SANCHEZ SAN BLAS HECTOR - 70901148Z
Date: 2022.07.04 17:51:46 +02'00'

D. Héctor Sánchez San Blas

DE PAZ SANTANA JUAN FRANCISCO - 76125754D
Firmado digitalmente por DE PAZ SANTANA JUAN FRANCISCO - 76125754D
Nombre de reconocimiento (DN): c=ES, serialNumber=IDCES-76125754D, givenName=JUAN FRANCISCO, sn=DE PAZ SANTANA, cn=DE PAZ SANTANA JUAN FRANCISCO - 76125754D
Fecha: 2022.07.05 10:23:54 +02'00'

D. Juan Francisco de Paz Santana

VILLARRUBIA GONZALEZ GABRIEL - 71932845T
Firmado digitalmente por VILLARRUBIA GONZALEZ GABRIEL - 71932845T
Fecha: 2022.07.07 12:02:55 +02'00'

D. Gabriel Villarrubia González

RESUMEN

En la actualidad, cuando llega agosto, septiembre u octubre, con el comienzo de las competiciones deportivas, es normal entre los seguidores de estas que se inicien ligas virtuales para concursar entre sí y ver quién es el que consigue más puntos a final de temporada. Estos puntos son obtenidos a través del desempeño real de los jugadores en el campo. La existencia de estas aplicaciones o sistemas hace que aumente la atención hacia las competiciones, o incluso que nazcan nuevos seguidores, ya que es una forma de sumergirse en ellas, mirando partidos, estadísticas, rumores, noticias, etc.

Este trabajo se centra en el desarrollo de una aplicación web que posibilita a los usuarios crear y participar en dichas ligas virtuales de la competición americana de baloncesto, NBA. Se les dotará de un equipo inicial, el cual tienen que gestionar a través de la compraventa de jugadores en el mercado, pudiendo así amoldar el roster a su gusto. Los puntos obtenidos en cada jornada, proveerán la posición en la clasificación de cada usuario, vendrán dados a través de un sistema de puntaje. Así mismo, cabrá la posibilidad de observar qué partidos se disputarán en esa jornada para que los participantes se decanten por unos jugadores u otros. Además de ver qué resultados hubo en partidos anteriores con las estadísticas individuales.

Para el desarrollo de las funciones de backend se ha utilizado TypeScript con Node.js y con la infraestructura para aplicaciones web de Node.js, Express. Para la autenticación de los usuarios, gestión de estos y almacenamiento de datos se realiza a través de Firebase, plataforma en la nube para el desarrollo de aplicaciones web de Google basada en servicios. La parte de frontend se lleva a cabo con el framework Ionic para los componentes de la interfaz gráfica, además de Vue.js para el resto de las funcionalidades. Por último, a través de la API sportdataverse, instalada a modo de paquete de Node.js, se recogen ciertos datos necesarios para el correcto funcionamiento de la aplicación.

Palabras clave: aplicación web, NBA, ligas virtuales, backend, TypeScript, Node.js, Express, Ionic, Vue.js.

ABSTRACT

Nowadays, when the sports competitions start usually in August, September and October the virtual leagues are initiated by the fans to race with each others and see who is the one with more points at the end of the season, these points are based on the players performance in real life. The existence of these applicatios allow users who take part on them, increase the attention to the competitions and evenmore the birth of new supporters due to is a good way to dive into them looking games, statistics, rumours, news, ...

This project is focused on the development of a web application which provides users with several options like create and take part on the virtual leagues of the american basketball competition called NBA. Will be provided with an initial team that the users have to manage through the player sales and the purchase of them from the market, being able to change their roster whatever they want. Moreover, the points scored each matchday which will give the position in the table, will be given by a scoring system. In addition, will be the possibility to see what games will be played that journey so that the participants change their starters, moreover see the results of the previous matches with the individual statistic of each.

Finally, for the development of backend functions TypeScript with Node.js and the infraestructure for Node.js web applications called Express has used. Whereas for user authentication, user management and data storage Firebase is used which is Google's service-based cloud platform for the development of web applications. Otherwise, the frontend part is carried out with the Ionic framework for user interface components and Vue.js for the rest of funcionalities. At last, such data is colleted through the API called sportdataverse which is installed like a Node.js package, for the correct functioning of the application.

Key words: web application, NBA, virtual leagues, backend, TypeScript, Node.js, Express, Ionic, Vue.js.

TABLA DE CONTENIDO

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

Desde la creación de Internet, las aplicaciones web han sufrido una evolución destacable, pasando de simples páginas en donde se mostraba un texto a páginas con una enorme capacidad de interacción, inteligencia artificial, bases de datos, etc. Es decir, se han convertido en aplicaciones software ejecutadas sobre el navegador web.

Con la aparición de los dispositivos móviles y el aumento de ordenadores personales, las aplicaciones web son cada vez más usadas y demandadas, disponiendo de una gran variedad de ellas, desde servicios para ver el tiempo hasta videojuegos, pasando por redes sociales. En adición del aumento de los dispositivos, según (Koeze & Popper, 2020) con la aparición de la pandemia el uso de aplicaciones de escritorio se ha visto reducido a favor de las aplicaciones web, en la *Ilustración 1* sacada de la fuente (Koeze & Popper, 2020) se puede observar el ejemplo de Facebook y Netflix.

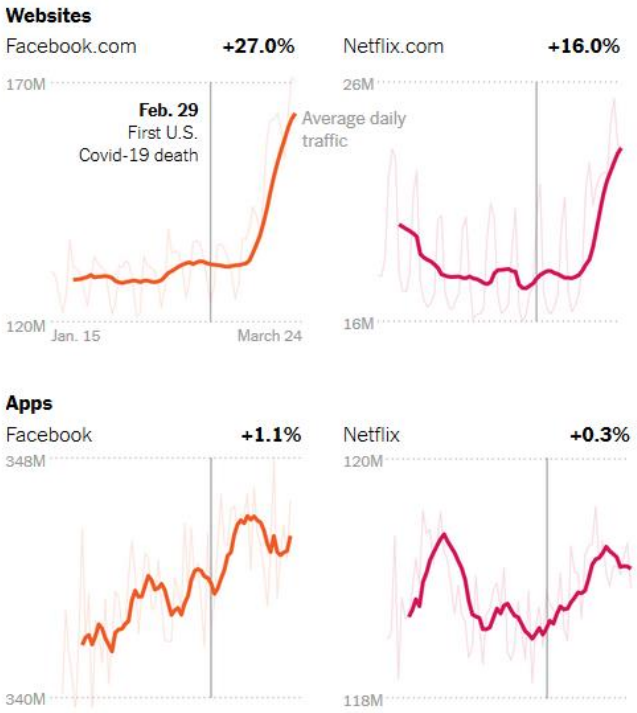


Ilustración 1: Gráfica de uso Facebook y Netflix

Uno de los tipos de aplicaciones más famosas entre los seguidores de las competiciones deportivas, son las aplicaciones que permiten simular dicha competición con las actuaciones reales de los jugadores participantes en ella. Cada vez son más los ámbitos del deporte y deporte electrónico que se están sumando a este tipo de sistemas, esto permite el aumento de interés por la competición, además de la adición de nuevos forofos.

El objetivo principal de este proyecto es el desarrollo de una aplicación web que permita crear ligas virtuales de la NBA (National Basketball Association), donde los participantes compiten entre ellos mediante un sistema de puntaje basado en el rendimiento real de los jugadores. Inicialmente, a estos se les asignará un equipo y presupuesto de partida de manera igualitaria entre todos. Cada liga contará además con un mercado para vender y comprar jugadores y con un calendario para visualizar los partidos y estadísticas individuales de los mismos.

1.1 Organización de la memoria

La finalidad principal de este documento es la de relatar los pasos realizados para el desarrollo del proyecto y la exposición de técnicas y herramientas utilizados.

Para ello esta memoria está estructurada de la siguiente forma:

- **Estado del arte:** Se realiza un estudio de mercado sobre las aplicaciones similares a la aplicación que se pretende desarrollar existentes en él.
- **Objetivos:** Se detallan tanto los objetivos personales como los del sistema que se tienen que satisfacer una vez finalizado el proyecto.
- **Conceptos teóricos:** Se definirán varios conceptos que se utilizarán a lo largo del proyecto para comprender su utilidad.
- **Técnicas y herramientas:** Se presentarán diversas técnicas y herramientas que se usarán para la resolución del proyecto.
- **Aspectos relevantes del desarrollo:** Se detallarán las tareas con más peso dentro del proyecto, así como un breve resumen de cada fase del mismo.
- **Funcionalidad del sistema:** Se realizará una pequeña guía para explicar el funcionamiento de la aplicación.
- **Conclusiones y líneas de trabajo futuras:** Se recogerán las ideas resultantes tras la finalización del proyecto, además de una exposición de posibles mejoras e implementaciones en un futuro.

1.2 Documentación

Adicionalmente a este documento se presenta el código fuente resultante del desarrollo del sistema y la documentación técnica comprendida en un conjunto de seis anexos, donde se despliegan las fases del *Proceso Unificado* de la siguiente manera:

- **Anexo I - Plan del Proyecto:** Realización de estimación de costes iniciales y planificación temporal del proyecto.
- **Anexo II - Especificación de Requisitos Software:** Recoge la documentación asociada a la especificación de requisitos software del sistema a desarrollar siguiendo la *metodología de Durán y Bernárdez*.
- **Anexo III - Análisis de Requisitos:** Contiene las tareas asociadas a la fase de análisis, en donde se definen de una manera más profunda los requisitos detallados en el *Anexo II - Especificación de Requisitos Software*. Conjuntamente se obtienen las clases de análisis y una primera vista de la arquitectura del sistema.
- **Anexo IV - Diseño del Sistema Software:** Desarrolla las tareas correspondientes a la fase de diseño software. Se produce un refinamiento de las especificaciones llevadas a cabo en el *Anexo III - Análisis de Requisitos*. Además, se define la arquitectura del sistema, el modelo de despliegue y el diseño de la base de datos.
- **Anexo V - Documentación técnica:** Desglosa la documentación asociada al código fuente y directorios que conforman el sistema. Sirviendo de ayuda a personas externas al proyecto para comprender el código fuente y estructura.
- **Anexo VI - Manual de usuario:** Detalla el proceso a seguir en cada una de las pantallas y funciones que presenta el sistema, de modo que ilustre a los usuarios sobre el funcionamiento del sistema y la correcta interacción con sus elementos.

2. ESTADO DEL ARTE

En este apartado se va a realizar un pequeño análisis de algunas aplicaciones análogas que se encuentran en el mercado.

Este tipo de sistemas tienen algunas características similares como el sorteo de jugadores iniciales, se intenta hacer de manera equilibrada para no adulterar la competición. Otra de las propiedades comunes es la existencia de ligas globales y de ligas personales, donde en las globales se compite contra muchos participantes del sistema por premios semanales, y en las personales son ligas creadas por los usuarios para competir entre amigos. También muestran los calendarios y resultados de todos los partidos en las jornadas.

2.1 SuperFantasy LOL

Se trata de una aplicación en donde se le proporciona al usuario una colección de jugadores de LOL (League Of Legends) base. A raíz de esta colección el usuario tendrá que guardar el roster para competir en la jornada, *Ilustración 2*.

Ilustración 2: SuperFantasy LOL, Pantalla Inicial

Se compite en una liga general, donde están todos los usuarios registrados, como se visualiza en la *Ilustración 3*, y a parte de la clasificación de la temporada global hay una clasificación por jornada con distintos premios. Además de esta liga general existe la posibilidad de unirse a una liga de club, con fans y premios del club en cuestión, o unirse a una liga pública. También se puede crear una liga propia cerrada.

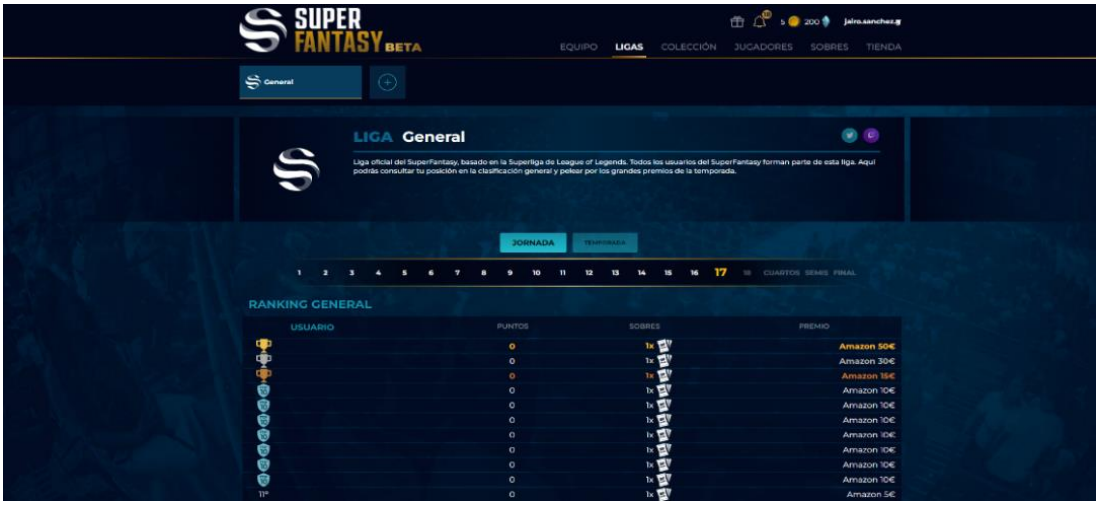


Ilustración 3: SuperFantasy LOL, Pantalla Ranking

La manera de obtener jugadores para la colección es mediante la apertura de sobres con créditos, situada en la sección *Sobres* mostrada en la *Ilustración 4*.

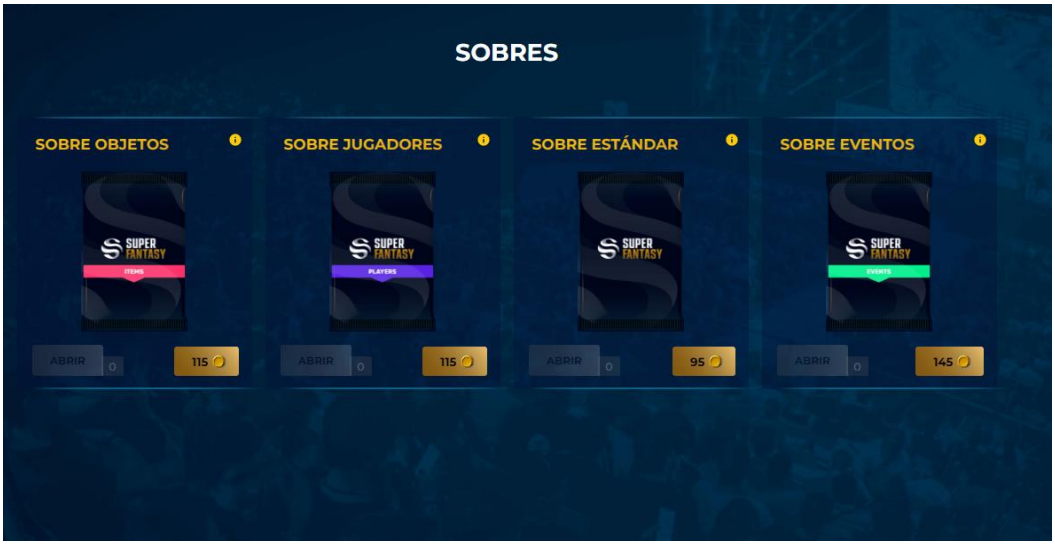


Ilustración 4: SuperFantasy LOL, Pantalla Sobres

Además de estas opciones comentadas, se tienen otras como la sección *Estadísticas*, en donde aparecen los jugadores con más puntos obtenidos en total y los puntos obtenidos jornada a jornada, *Ilustración 5*. La pantalla *Calendario* muestra los partidos y resultados por jornada, *Ilustración 6*.

Ilustración 5: SuperFantasy LOL, Pantalla Estadísticas

Ilustración 6: SuperFantasy LOL, Pantalla Calendario

Y por último, la opción *Colección* expone las cartas que posee el usuario, mostrada en la *Ilustración 7*.

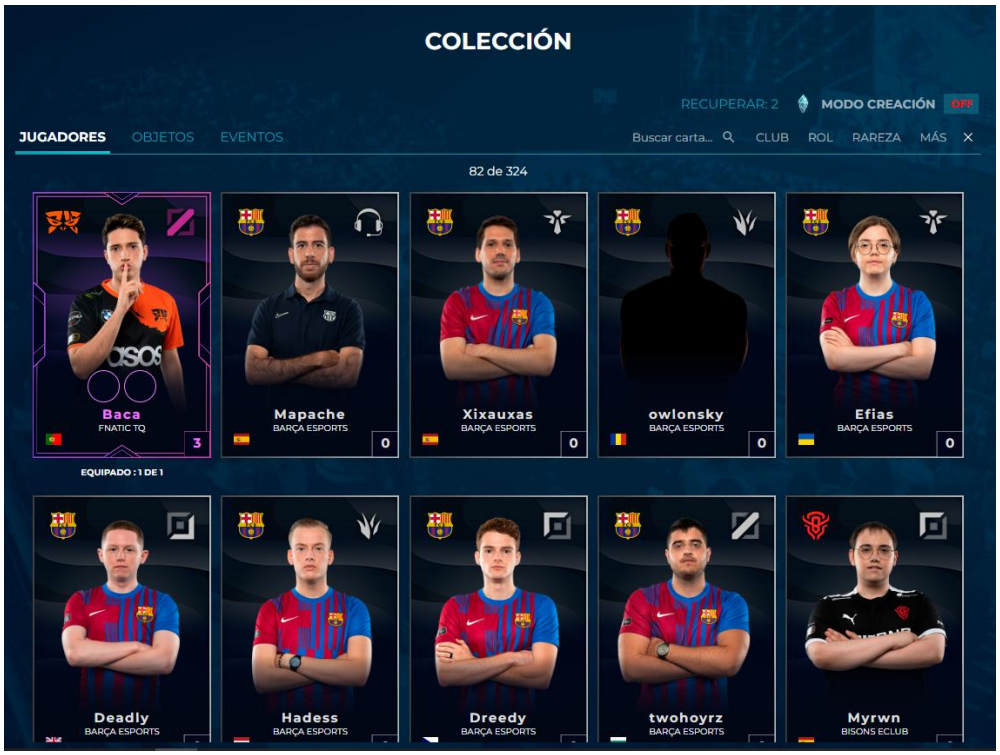


Ilustración 7: SuperFantasy LOL, Pantalla Colección

2.2 La Liga Fantasy Marca

Es la aplicación de fútbol más jugada. En ella una vez registrado el usuario, este se tendrá que meter a alguna liga, bien de las existentes, a través de un código sin son privadas o directamente si son públicas, o bien creando una liga, estas opciones se pueden observar en la pantalla de inicio, así como un calendario de partidos y las últimas noticias relacionadas con la competición, *Ilustración 8*. Posteriormente, cuando el usuario entre en una liga, se le proporcionará un equipo de un valor similar a los demás cuando entraron, además de un presupuesto de unos 100 millones.

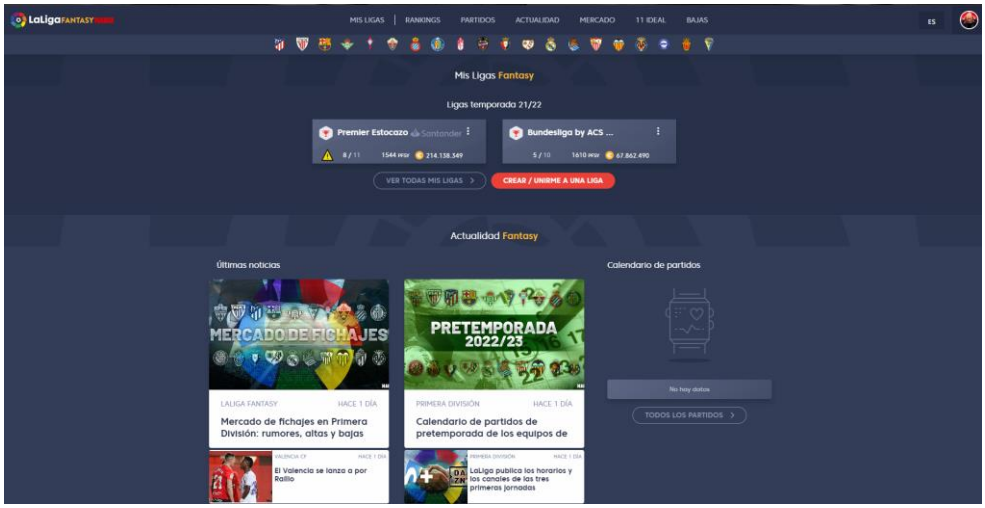


Ilustración 8: Fantasy Marca, Pantalla Inicio

A parte de unas secciones generales que posee, se explicarán las específicas que se encuentran dentro de cada liga ya que contienen la funcionalidad principal de la aplicación. La primera que se observa es la pantalla *Clasificación*, donde se muestra el ranking de los usuarios según los puntos que han ido obteniendo jornada a jornada, permite visualizar la clasificación de dicha jornada así como ver los equipos y alineaciones que utilizaron los otros usuarios en ella, *Ilustración 9*.

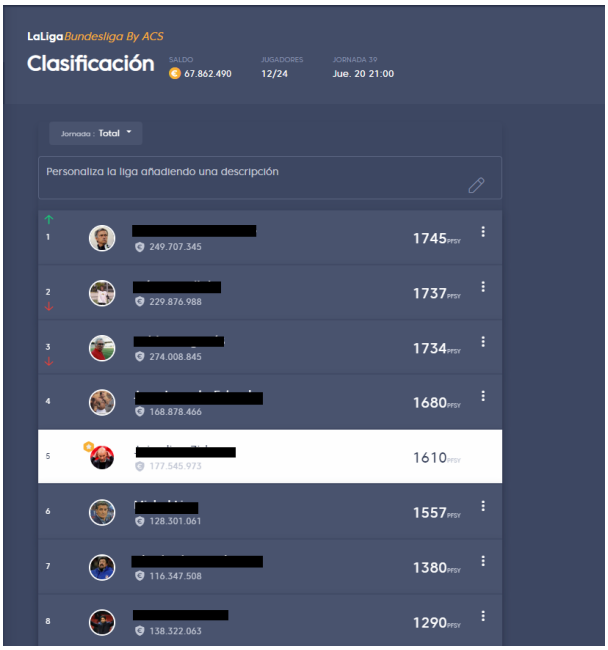


Ilustración 9: Fantasy Marca, Pantalla Clasificación

A continuación, en la pantalla *Mi equipo*, *Ilustración 10*, aparece la lista de jugadores que tiene el usuario y la alineación, la cual se puede modificar poniendo y quitando jugadores. Sobre cada uno se puede hacer varias operaciones como subirles la cláusula o venderlos en el mercado.

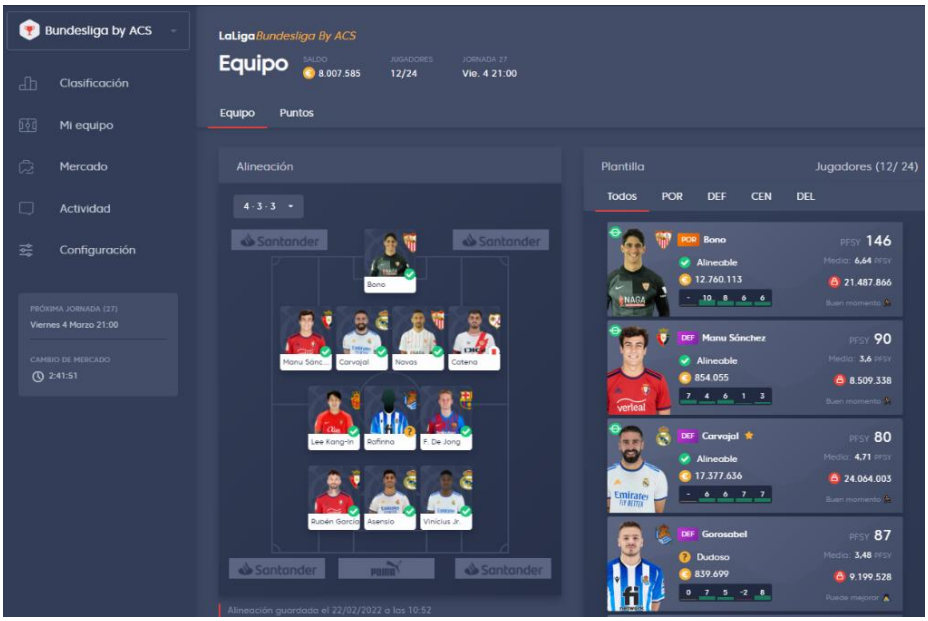


Ilustración 10: Fantasy Marca, Pantalla Mi Equipo

Por último, se tienen las pantallas *Mercado* y *Actividad*. En la primera se tiene la lista de jugadores que los usuarios pueden comprar mediante un sistema de pujas. En ella aparecerán también los jugadores que los usuarios hayan mandado al mercado, *Ilustración 11*. Para terminar, en la segunda pantalla mencionada al inicio del párrafo, se muestran las noticias relacionadas con la liga, como el dinero ganado cada jornada o los fichajes realizados.

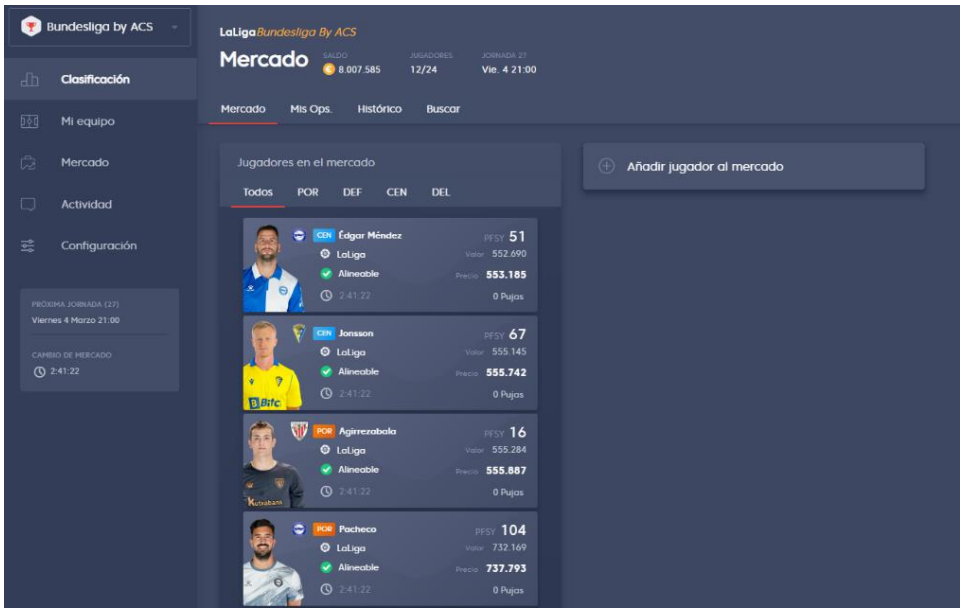


Ilustración 11: Fantasy Marca, Pantalla Mercado

2.3 Biwenger NBA As

Es la aplicación de mánager de baloncesto más usada en España. Como pasa en la aplicación anterior se necesita de un registro y de un ingreso en una liga para que se le proporcione el equipo y el saldo al usuario. Se encuentran varias secciones como la de *Inicio*, en donde aparecerán noticias relacionadas con la NBA y con la propia liga, como se observa en la *Ilustración 12*.

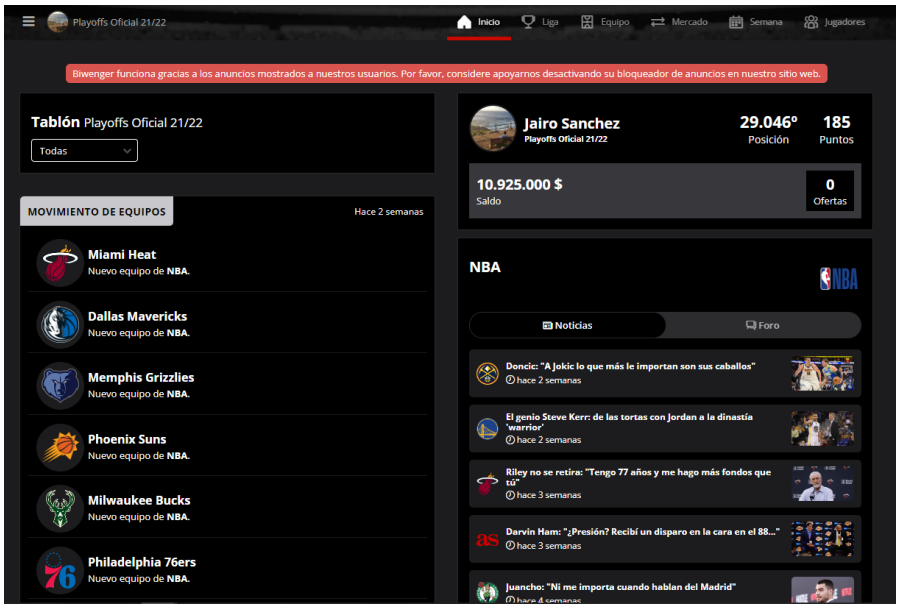


Ilustración 12: Biwenger NBA, Pantalla Inicio

Por otro lado, en la sección de *Liga* se muestra la clasificación de la liga con los puntos que han ido obteniendo los usuarios a lo largo de las jornadas, se permite aplicar una serie de filtros para ordenar esta clasificación por valor de equipo o número de jugadores, aparte de por posición. El apartado *Equipo*, *Ilustración 13*, donde cada usuario podrá gestionar sus jugadores y quinteto, dando la posibilidad de realizar varias acciones sobre cada jugador como añadirlo o quitarlo del quinteto o venderlo.

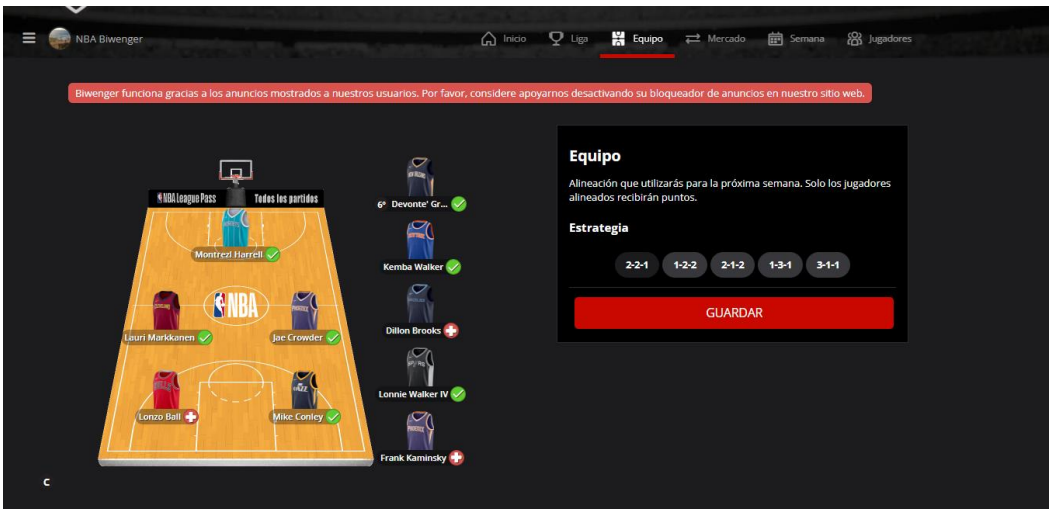


Ilustración 13: Biwenger NBA, Pantalla Equipo

Por último, se tienen la opción de *Mercado* y la opción *Semana*, *Ilustración 14* e *Ilustración 15* respectivamente. En la primera sección se pueden comprar jugadores pujando por ellos, así como vender los propios del usuario. Para finalizar con la sección *Semana*, se pueden observar los puntos que los jugadores del usuario están obteniendo en la jornada, así como la clasificación semanal de la liga y un calendario de partidos para conocer qué equipos juegan.

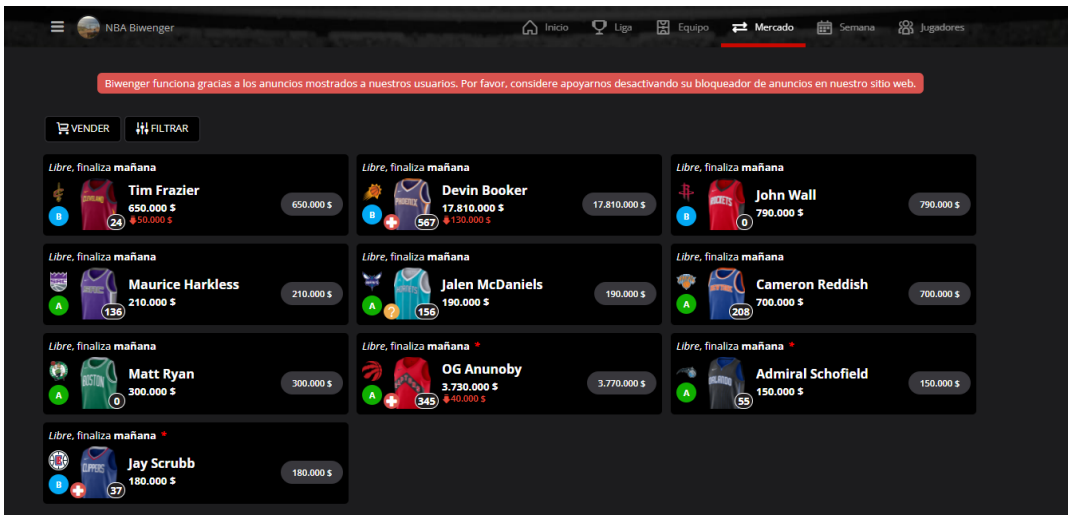


Ilustración 14: Biwenger NBA, Pantalla Mercado

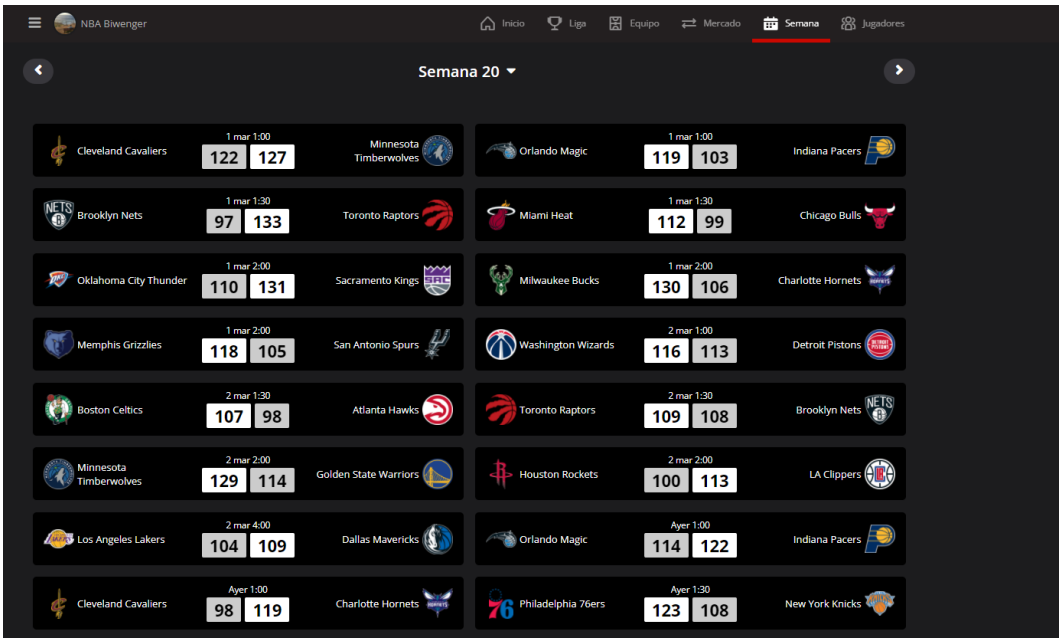


Ilustración 15: Biwenger NBA, Pantalla Semana

2.4 Comunio

Se trata de una de las primeras aplicaciones de este tipo. Sus funcionalidades no distan mucho de las comentadas anteriormente. Para entrar se necesita de un registro, en donde se tienen que rellenar los campos con los datos del usuario y seleccionar el tipo de promoción que quiere (básica o premium mediante un pago mensual), además de seleccionar la liga a donde el usuario quiere pertenecer, sino lo mete a una que sea pública.

Una vez dentro en la liga se puede encontrar una interfaz sencilla con distintas opciones: noticias, clasificación, alineación, mercado, copa y predicciones. En la opción de noticias, *Ilustración 16*, se encuentran todas las informaciones correspondientes a la liga como fichajes, ventas, mejores alineaciones de la jornada, entre otras. En el apartado de clasificación, *Ilustración 17*, se puede ver el ranking que ocupan los miembros de la liga en función de sus puntos obtenidos, cabe la posibilidad de filtrar por jornada para visualizar el puntaje y resultados de esa fecha. También desde esta opción se permite revisar los equipos que tienen los demás usuarios en una jornada en concreto o de manera general.

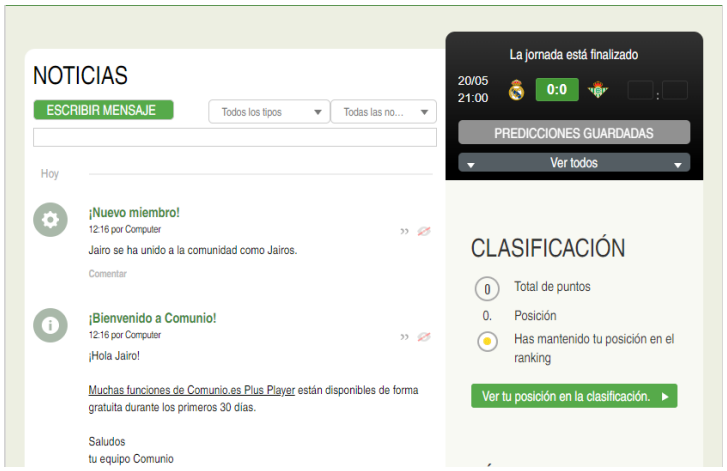


Ilustración 16: Comunio, Pantalla Noticias

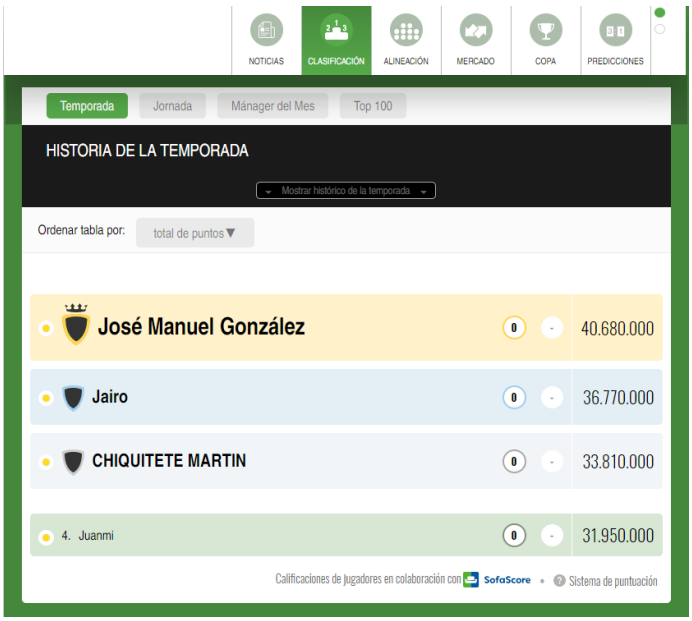


Ilustración 17:Comunio, Pantalla Clasificación

Pasando a la ventana de alineación, *Ilustración 18*, se aprecia una opción más interactiva donde poder modificar el tipo de alineación, los jugadores a utilizar por posición e incluso partidos y resultados de la jornada activa. Desde este calendario, *Ilustración 19*, se brinda la posibilidad de ver en detalle los jugadores de cada equipo y de cada partido. Por último, se tiene la opción de mercado, en la cual el usuario puede pujar por los jugadores que aparecen cada día y los que ponen otros usuarios a la venta, así como poner a la venta a los suyos propios.

Ilustración 18: Comunio, Pantalla Alineación

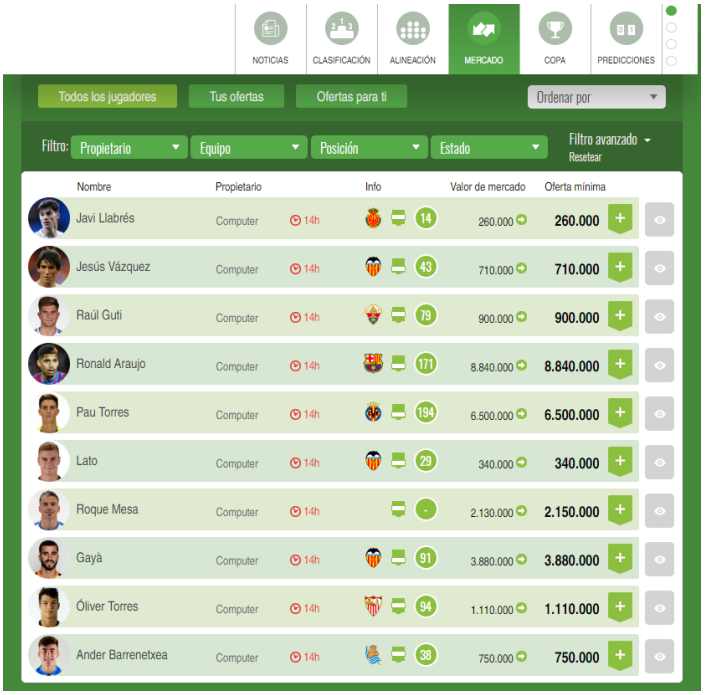


Ilustración 19: Comunio, Pantalla Mercado

3. OBJETIVOS

En este apartado se detallarán los diferentes objetivos tanto personales como del sistema que el Trabajo de Fin de Grado debe satisfacer.

3.1 Objetivos personales

Seguidamente, se describen los objetivos personales que me han hecho proponer y desarrollar este tema.

La principal meta cuando se decidió el tema del Trabajo de Fin de Grado era conocer en profundidad como se lleva a cabo un proyecto real, pasando por todas las fases ya que a lo largo de la carrera no se realiza ninguno como tal, sino que se hacen pequeños trozos en diferentes asignaturas. Por ello, ponerlo en práctica de manera conjunta y ver las dificultades que supone es un gran reto personal.

Otro de los objetivos es la curiosidad que siempre me ha despertado cómo funcionarían las aplicaciones y sistemas, es decir, cómo serían internamente, de qué manera conseguirían los datos o la manera de construir la interfaz. Gracias a este proyecto he podido comprender la manera de desarrollar este tipo de aplicaciones y las arquitecturas que siguen.

Por último, debido a la poca experiencia laboral, ha sido una buena oportunidad para aprender nuevos lenguajes como TypeScript e instruirse en trabajar con diferentes marcos de trabajo como Ionic y Vue. Además de ganar experiencia en las comunicaciones Cliente-Servidor a través del desarrollo de una API REST.

A parte del apartado técnico, es gratificante poder desarrollar una aplicación de estas características ya que he sido usuario de ellas desde que se empezaron a comercializar.

3.2 Objetivos del sistema

En este apartado se van a exponer los objetivos que el sistema tiene que cumplir para llevar a cabo las funcionalidades de este.

El objetivo principal es el desarrollo de un sistema que permita a los usuarios crear ligas virtuales para competir entre ellos a través de un sistema de puntaje basado en las actuaciones reales de los jugadores. A parte, se tiene que garantizar una correcta gestión de usuarios y del mercado entre otros objetivos explicados a continuación:

- **Gestión de los usuarios:** El sistema debe permitir a los usuarios darse de alta en el sistema, entrar y salir de él. Además debe ser capaz de consentir a estos la modificación de sus datos.
- **Gestión de las ligas:** El sistema debe permitir la creación de ligas, la adhesión y salida de los usuarios a estas. Así como el correcto funcionamiento dentro de cada liga, como el control de la clasificación, el mercado o los equipos.
- **Gestión de las jornadas:** El sistema debe generar correctamente la puntuación de cada jornada en función de las estadísticas. Además de la actualización del presupuesto y de la puntuación total.
- **Gestión del roster:** El sistema debe generar un roster para cada usuario de manera equilibrada, además de permitir al propietario poner y quitar a los jugadores del quinteto inicial o mandarlos al mercado para su venta.

- **Gestión del mercado:** El sistema debe permitir pujar por jugadores, modificar o eliminar las pujas realizadas, vender y quitar jugadores propios.
- **Gestión del calendario:** El sistema debe consentir visualizar los partidos y resultados por fecha, así como las estadísticas por parte de los jugadores.
- **Gestión de las noticias:** El sistema debe permitir visualizar las noticias relacionadas con las ligas existentes.

4. CONCEPTOS TEÓRICOS

En este apartado se explicarán diferentes conceptos que se mencionan y/o utilizan para llevar a cabo el proyecto.

4.1 Firebase

Se trata de una plataforma para el desarrollo de aplicaciones web, móvil y juegos gestionada por *Google*. Se encuentra en la nube utilizando un conjunto de herramientas para la creación y monitorización de proyectos. Ofrece una serie de ventajas como el uso de herramientas plataforma, la utilización de la infraestructura de *Google* lo que beneficia al escalado y no es necesario el uso de un servidor a través de las herramientas SDK que incluye. (Google, s.f.)

A parte, brinda unos servicios que se pueden incluir de manera sencilla en los proyectos a través de la instalación de módulos, dentro de estos servicios se encuentran *Firebase Analytics*, para llevar a cabo una monitorización de la aplicación. *Firebase Cloud Messaging*, plataforma para mensajes y notificaciones. *Firebase Auth*, servicio de autenticación de usuarios. *Realtime Database*, como su nombre indica es una base de datos en tiempo real organizada en forma de árbol JSON. *Firebase Storage*, almacenamiento de archivos en la nube. Y por último*, Firebase Cloud Firestore* proporciona un servicio de almacenamiento de datos en la nube.

4.2 API REST

Se define como un conjunto de reglas que definen cómo se comunican y relacionan los sistemas entre sí, cumpliendo los principios de diseño de la arquitectura REST. Algunas de estos principios de diseño más importantes son el desacoplamiento cliente-servidor, el no mantenimiento del estado y la interfaz uniforme. (Santamaría, 2022)

La comunicación se realiza a través de peticiones HTTP mediante las operaciones que estas ofrecen: GET, POST, PUT y DELETE.

4.3 Arquitectura Cliente-Servidor

Es un modelo arquitectónico compuesto por uno o varios clientes que realizan peticiones y un servidor que resuelve dichas peticiones. Presenta una serie de ventajas en inconvenientes que se van a mostrar a continuación.

Dentro de las ventajas se encuentra la centralización que permite una consistencia en los datos utilizados por todos los clientes. Facilidad de instalación debido a las pocas dependencias, desacoplamiento con el cliente de manera que se separan la lógica de negocio con la de presentación.

Por la parte de los inconvenientes, se tiene a su vez la centralización ya que un número elevado de clientes puede llevar al servidor a que funcionara como un cuello de botella. La tolerancia a fallos, se produciría un fallo de proceso en caso de caída del servidor.

4.4 Caché web

Es un recurso utilizado por los navegadores, proveedores de Internet y servidores con el propósito de reducir el ancho de banda al cargar páginas web. Sirviendo también de almacenamiento de datos temporales.

Este tipo de caché se encuentra implementada en varios niveles en función de su uso. Por un lado se tienen las caché web privadas propias de los navegadores web que almacenan en memoria principal las páginas visitadas asiduamente. Por otro lado, existen las caché compartida que es la que utilizan los proveedores de Internet y por último, las caché web pasarela que son implementadas a nivel de servidor y son iguales a todos los usuarios. (Parsec Media S.L, 2020)

4.5 Framework

Es un marco de trabajo utilizado para el desarrollo de software que permite agilizar los procesos y evita la redundancia de código. Contiene un conjunto estandarizado de objetos y prácticas para resolver problemas. Puede incluir soporte de programas, bibliotecas y un lenguaje interpretado.

4.6 Responsive

Se refiere al diseño web adaptable, en donde el propio diseño y desarrollo tiene como objetivo acomodar la interfaz a cada dispositivo y resolución de pantalla desde donde se accede a la aplicación, ya sea un teléfono móvil, un ordenador o una tablet.

4.7 Herramientas CASE

Conjunto de aplicaciones informáticas que proporciona ayuda automatizada a las actividades del proceso de desarrollo de software. Dan soporte a los desarrolladores a lo largo del ciclo de vida en tareas como el diseño de proyectos, cálculo del esfuerzo o documentación entre otras, esta ayuda normalmente suele ser automatizada por estas herramientas.

El uso de las mismas contribuye a una mejora de la calidad y productividad en el proceso de realización de ciertas tareas permitiendo aplicar metodologías estructuradas, aumentar la eficacia de estandarización de la documentación o facilitar la reutilización de componentes. (EcuRed, s.f.)

5. TÉCNICAS Y HERRAMIENTAS

A continuación se expondrán el conjunto de técnicas, herramientas, módulos, lenguajes de programación y frameworks usados para lograr el desarrollo del proyecto.

5.1 Técnicas y herramientas utilizadas en aplicación web

5.1.1 Vue.js

Se define como un framework progresivo de JavaScript para construir interfaces de usuario. Se compone de una biblioteca central accesible que se centra en la capa de vista solamente, y un ecosistema de bibliotecas de apoyo que le ayuda a abordar la complejidad en grandes aplicaciones de una sola página. (Vue.js, s.f.)

En su ecosistema incluye *Vue Router* que proporciona un modo de enrutado basado en la URL, donde se permite cambiar la interfaz en función de esta. Así mismo, admite el mapeo de rutas anidadas y ofrece un control de transición uniforme.

Por otro lado, *Vue* a través de *Vue Class Component* ofrece la posibilidad de utilizar un lenguaje basado en clases como TypeScript.

5.1.2 Ionic

Ionic es un SDK de frontend para el desarrollo de aplicaciones híbridas basado en HTML, CSS y JavaScript. Permite crear aplicaciones para iOS nativo, Android y web desde una sola base de código mediante el plugin Capacitor. Se integra con otros marcos de trabajo como *Vue* o *Angular*.

Proporciona componentes, bloques de construcción de alto nivel, para implementar interfaces de usuario con un diseño sencillo y limpio. (Ionic Framework, s.f.)

5.1.3 CSS y HTML

CSS es un lenguaje de diseño gráfico utilizado para describir la presentación de documentos escritos en un lenguaje marcado. Describe la manera en la que debe ser renderizado un elemento en la página.

Mientras que HTML, es un lenguaje de marcas que define el significado y la estructura del contenido web. A parte de CSS para la apariencia, utiliza JavaScript para describir su funcionalidad.

5.2 Técnicas y herramientas utilizadas en el servidor

5.2.1 Node.js

Se trata de un entorno en tiempo de ejecución multiplataforma para la capa del servidor basado en JavaScript y en el motor V8 de Google. Presenta una arquitectura de programación asíncrona donde permite gestionar un elevado número de eventos ejecutados de manera independiente unos de otros. Esto se ha conseguido cambiando la manera en la que se realizan las conexiones con el servidor, pasando de un hilo de ejecución para cada cliente a la generación de un evento por petición. (OpenJS, s.f.)

5.2.2 Express

Express es una infraestructura de aplicaciones web *Node.js* mínima y flexible que proporciona un conjunto sólido de características para las aplicaciones web y móviles. Ofrece una gran cantidad de métodos de programa HTTP y middleware para crear una API sólida. (Fundación Openjs, s.f.)

5.2.3 NPM

Es el gestor de paquetes utilizado por *Node.js*, mediante el cual se puede incluir cualquier módulo a través su comando en consola.

5.2.4 TypeScript

Es un lenguaje de programación que se trata de un superconjunto de *JavaScript*. Con él a diferencia de JavaScript se puede realizar una programación orientada a objetos con una sintaxis parecida a Java o C#. Se trata de un lenguaje fuertemente tipado y que da la posibilidad de utilizar interfaces y decoradores.

5.2.5 JSON

JSON(JavaScript Object Notation) es un formato de texto ligero para el intercambio de datos. Es independiente del lenguaje de programación y está constituido por dos estructuras, una colección de pares nombre-valor y una lista ordenada de valores.

En la *Ilustración 20*, se puede observar un ejemplo de la estructura de este formato en una petición de datos hacia un servidor.


```
[
    {
        "name": "Aleksej Pokusevski",
        "photo": "https://a.espncdn.com/i/headshots/nba/players/full/4683018.png",
        "position": "Center",
        "salary": 3113160,
        "points": 12
    },
    {
        "name": "Duncan Robinson",
        "photo": "https://a.espncdn.com/i/headshots/nba/players/full/3157465.png",
        "position": "Shooting Guard",
        "salary": 15650000,
        "points": 17
    },
    {
        "name": "Jimmy Butler",
        "photo": "https://a.espncdn.com/i/headshots/nba/players/full/6430.png",
        "position": "Small Forward",
        "salary": 36016200,
        "points": 31
```


Ilustración 20: Ejemplo estructura JSON

5.2.6 Postman

Postman se trata de una plataforma de API donde se puede diseñar, construir y probar otras API. Permite crear peticiones HTTP hacia los servicios de una manera sencilla añadiendo los parámetros o el cuerpo de la petición cómodamente , así como visualizar la respuesta de estos en varios formatos como JSON, XML o HTML. (Postman, s.f.)

En la *Ilustración 21* se puede ver la manera de probar el servicio creado a través de esta herramienta.

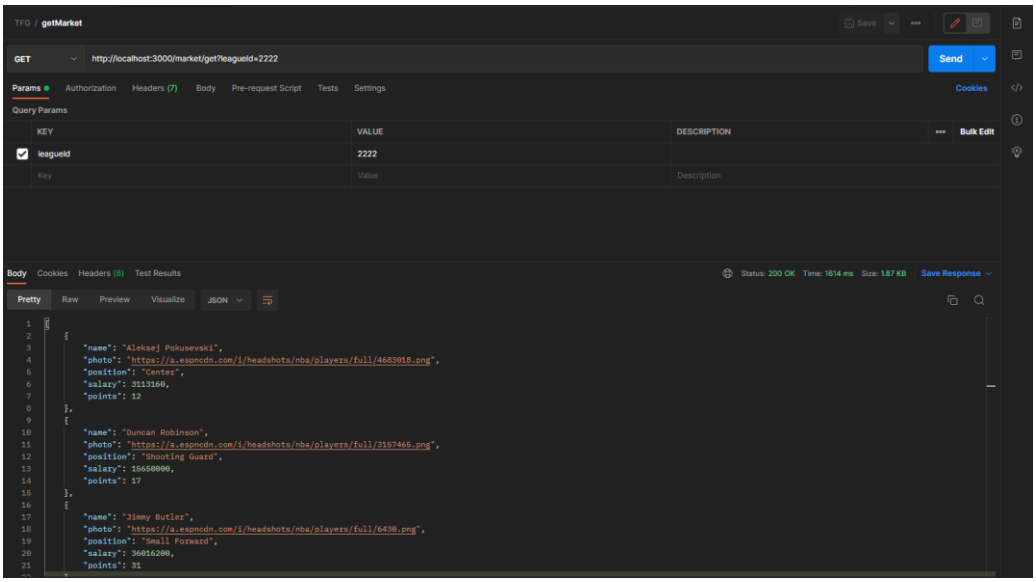


Ilustración 21: Postman

5.2.7 Firebase Cloud Firestore

Es uno de los servicios proporcionados por *Firebase*. Se trata de una base de datos NoSQL en la nube basada en colecciones y documentos. Crea jerarquías para almacenar datos relacionados y recuperar los datos mediante consultas expresivas, donde cada consulta se escala con el conjunto de resultados. (Google, s.f.)

En la *Ilustración 22* se observa un ejemplo de colecciones y documentos de la base de datos.

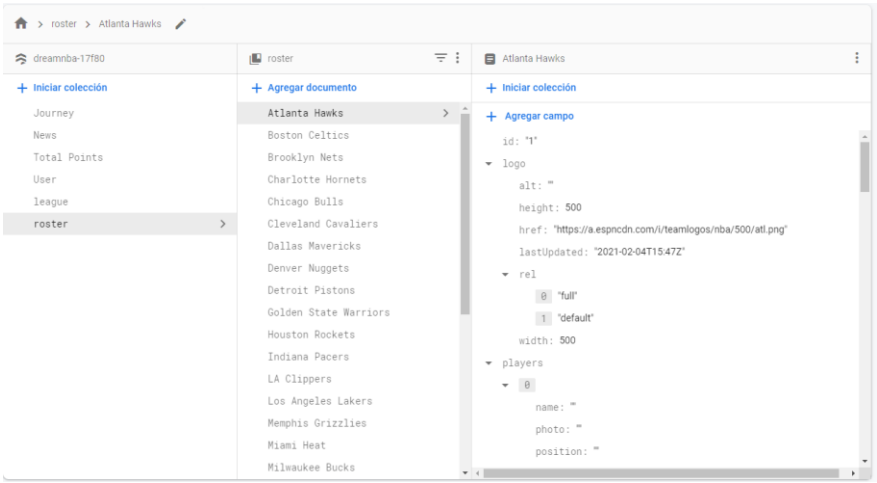


Ilustración 22: Firestore

5.2.8 Firebase Authentication

Servicio proporcionado por *Firebase* para gestionar de manera sencilla la autorización de los usuarios de la aplicación, ofrece una interfaz con diferentes métodos para realizar tareas de login, logout, recuperación de contraseñas, registro, verificación, entre otras. (Google, s.f.)

5.2.9 API Sportdataverse

Módulo instalado con npm que ofrece datos relacionados con competiciones deportivas de Estados Unidos. Se ha utilizado para obtener cierta información actualizada sobre la NBA. (Gilani, s.f.)

5.3 Herramientas CASE

5.3.1 Visual Paradigm for UML

Para realizar el modelado visual de los procesos de ingeniería del software a lo largo del ciclo de vida del proyecto se ha utilizado *Visual Paradigm*.

Se define como una herramienta donde modelar las distintas fases del Proceso Unificado a través de la realización de los diferentes diagramas que las componen. Facilita este proceso proporcionando lo necesario para llevar a cabo las tareas. (Visual Paradigm, s.f.)

En la *Ilustración 23* se puede observar la interfaz de *Visual Paradigm for UML*.

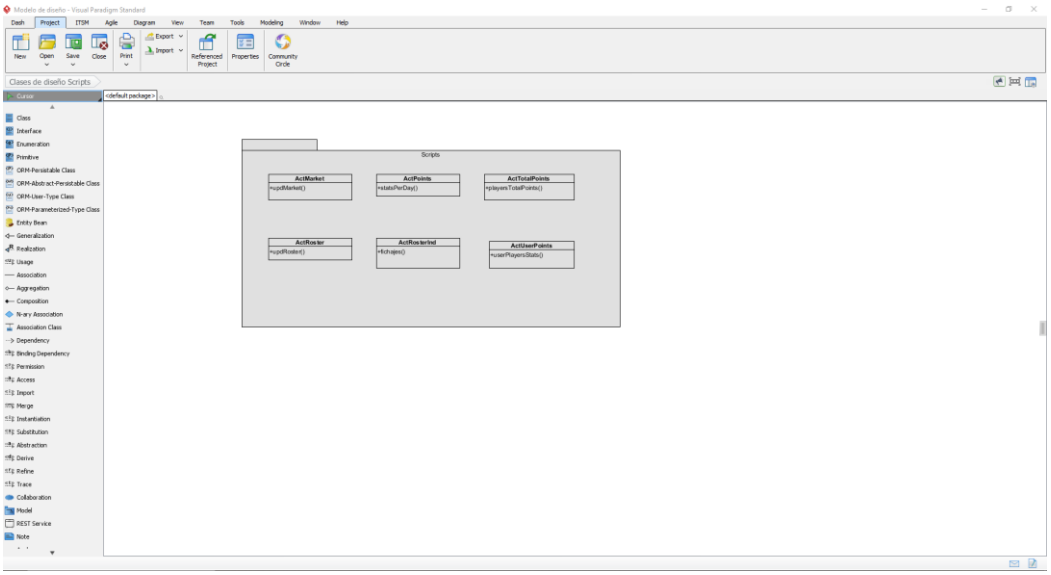


Ilustración 23: Visual Paradigm for UML

5.3.2 Microsoft Project

Como herramienta CASE para la planificación temporal del proyecto se ha hecho uso de *Microsoft Project* en una fase inicial de este.

Se define como un software de administración de proyectos y programas desarrollado por Microsoft para dar soporte a los líderes de proyecto en la asignación de recursos y tareas, en la estimación de costes, en el análisis de las cargas de trabajo, entre otras. (Microsoft, s.f.)

En la *Ilustración 24* se puede ver un ejemplo de la interfaz de esta herramienta.

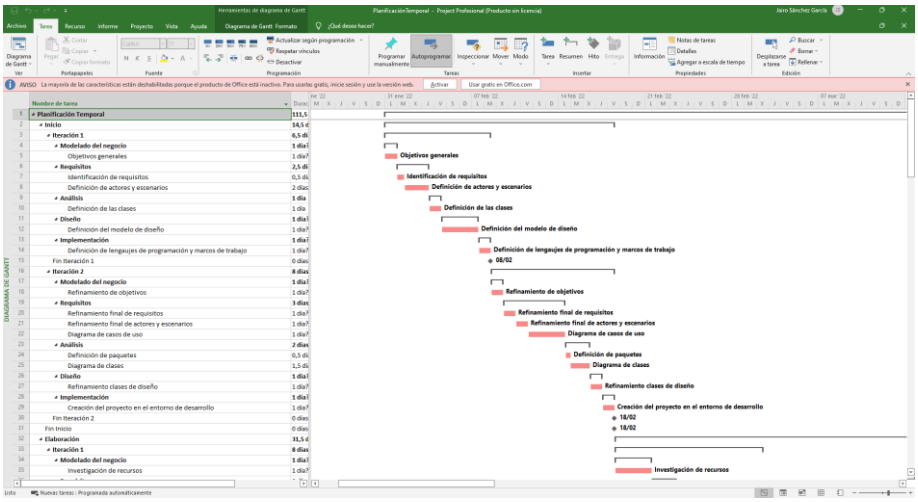


Ilustración 24: Microsoft Project

5.3.3 EZEstimate

Para llevar a cabo la estimación de los costes y esfuerzo que supone realizar el proyecto se ha utilizado *EZEstimate*.

EZEstimate es una herramienta que a partir de la definición de complejidad de los casos de uso y actores, y la introducción de los factores técnicos y del entorno calcula una estimación en horas de lo que se tardaría en desarrollar el proyecto. Para producir esa evaluación de los costes computa las tres variables de la métrica UCP(Use Case Points).

En la siguiente figura, *Ilustración 25*, se muestra un ejemplo de *EZEstimate*.

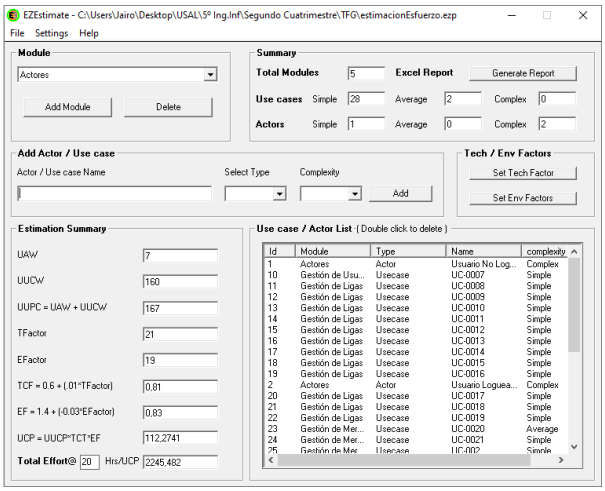


Ilustración 25: EZEstimate

5.4 Entornos de desarrollo

Para llevar a cabo el proceso de programación se ha hecho uso del entorno de desarrollo explicado a continuación.

5.4.1 Visual Studio Code

Visual Studio Code se trata de un editor de código fuente independiente que se ejecuta en distintos sistemas operativos. Admite un gran número de lenguajes de programación a través de su instalación por parte de extensiones. Así mismo, posee una consola propia que no es más que una interpretación de la consola del sistema en donde se pueden compilar y ejecutar programas. (Microsoft, s.f.)

Se ha elegido como entorno de desarrollo para el proyecto debido a su capacidad de interacción y sencillez que ayuda a la realización del código fuente.

A continuación, en la *Ilustración 26* se muestra un ejemplo de su interfaz.

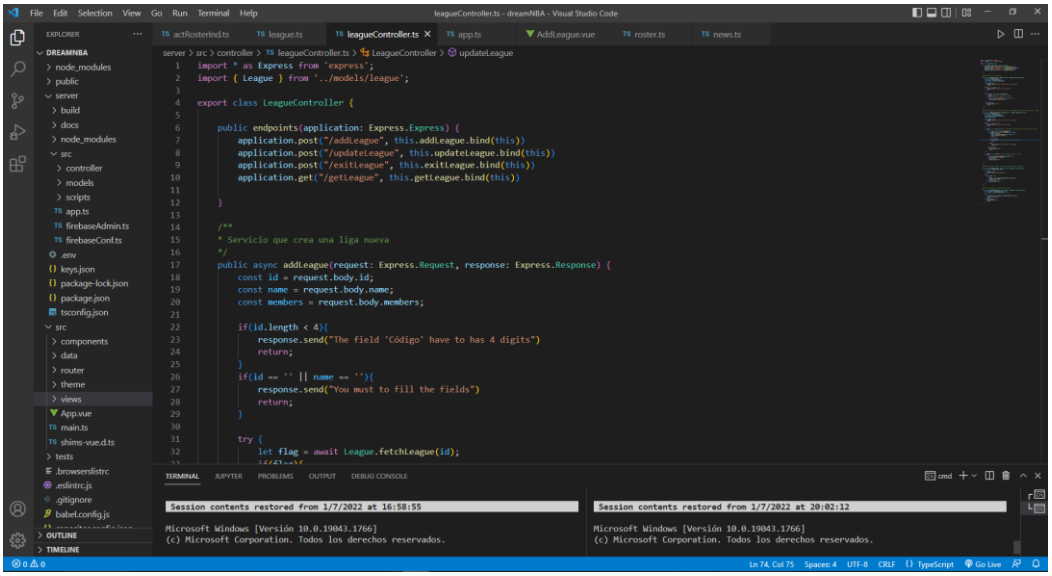


Ilustración 26: Visual Studio Code

5.5 Herramientas de generación de documentación

Para la generación automática de documentación del código se ha utilizado *TypeDoc* (Typedoc, s.f.), debido a que una gran parte del código está escrito en TypeScript. Se ha procedido a su instalación con npm y posteriormente mediante la ejecución de un comando se ha generado dicho informe en un archivo HTML.

6. ASPECTOS RELEVANTES DEL DESARROLLO

En esta sección se detallarán las fases con mayor relevancia por las que se ha pasado para llegar a la resolución total del proyecto bajo un marco de trabajo. Se va a recorrer todo el ciclo de vida del desarrollo software desde la estimación de costes hasta la fase de implementación.

6.1 Marco de trabajo

Como se definió en el apartado *4.5 - Framework* de este mismo documento, un marco de trabajo es un conjunto estandarizado de objetos y prácticas que ayudan al desarrollo de un sistema. La utilización de un correcto marco de trabajo garantiza un aumento de la calidad del sistema, dotándolo de robustez y consistencia, y una reducción de costes y esfuerzo. También permite realizar una esquematización detalla del proyecto limitando así el número de cambios que se pueden producir en un futuro, además de una reutilización de software.

Se han valorado varios marcos de trabajo para su uso en este proyecto. Debido a la poca experiencia en estas técnicas y a las necesidades del proyecto, se ha decidido usar el Proceso Unificado. Proporciona a este proyecto un desarrollo completo de cada fase permitiendo comprender la manera completa de cómo se lleva a cabo un proyecto real, mientras se va definiendo a su vez el sistema.

El *Proceso Unificado* tiene una serie de características principales que lo hacen ser un proceso software genérico utilizado en distintas áreas y sobre diferentes tamaños de proyecto. Estas características son:

- **Dirigido por casos de uso:** Emplea los casos de uso para satisfacer los requisitos del usuario. Se recogen los requisitos funcionales y a través de estos se guía el proceso de desarrollo mediante la realización de los casos de uso, es decir, se erige el sistema software sobre la funcionalidad final.
- **Centrado en la arquitectura:** Una arquitectura bien definida y consensuada entre los participantes del proyecto sienta las bases de un sistema robusto y consistente. La arquitectura normalmente está compuesta por varios modelos que cubren las necesidades del sistema.
- **Iterativo e incremental:** Divide al proyecto en diversas fases con diferentes versiones del producto. Esta característica es importante ya que reduce considerable los riesgos.

Este proceso software consta de un esquema de fases y disciplinas, en donde cada fase se va a descomponer en un número de iteraciones. A continuación se procede a definir las distintas fases y disciplinas del *Proceso Unificado*:

- **Inicio:** En esta fase se define el alcance del proyecto y se desarrollan los casos de negocio.
- **Elaboración:** Se realiza la planificación y detalle de los casos de uso, así como el diseño de la arquitectura.
- **Construcción:** Se procede a hacer la construcción del proyecto.
- **Transición:** refinamiento de problemas y mejoras.

Cada una de estas fases se descomponen a su vez en una serie de disciplinas:

- **Modelado del negocio:** se centra en reuniones con los clientes o internas entre los propios participantes en el proyecto. Se definen los riesgos, costes, detalles del sistema, entre otras cosas. Se pretende modelar el proyecto implicando a todas las partes que participan en él. La mayor carga de trabajo de esta disciplina reside en la fase de inicio.
- **Requisitos:** se produce la especificación de requisitos software, incluyendo la descripción y desarrollo de los casos de uso y sus diagramas correspondientes. Tiene más peso en las fases iniciales del proyecto aunque la carga de trabajo prosigue a lo largo del ciclo de vida debido a los cambios que se pueden producir.
- **Análisis:** como su propio nombre indica se produce en esta fase se va a realizar el análisis de requisitos. Se identifican las clases de análisis y se realiza la arquitectura del modelo de análisis, aparte de definir el comportamiento dinámico de los casos de uso a través de los diagramas de secuencia y el modelado de datos con el diagrama de clases. Su carga de trabajo se concentra especialmente en la fase de elaboración, pero como pasa con la disciplina anterior se va a extender a lo largo del ciclo de vida a causa de las revisiones y refinamientos.
- **Diseño:** se produce el modelo de diseño, conformando el sistema en un nivel más bajo. Una de las tareas que se producen en esta disciplina es la realización casos de uso diseño y la elaboración del diagrama de clases de diseño. Se definen los patrones arquitectónicos y de diseño que se van a utilizar en el desarrollo del sistema. En las fases de elaboración y construcción reside la mayor carga de trabajo de esta disciplina.
- **Implementación:** se hacen las tareas relacionadas con la programación del sistema. En las primeras fases su carga de trabajo es casi indiferente debido a la poca información y modelado que se tiene del sistema, por ello aumenta de manera considerable en la fase de construcción donde implica el mayor tiempo de trabajo de esta.
- **Pruebas:** en esta disciplina se evalúan los aspectos relevantes del sistema y el correcto funcionamiento de este evitando arrastrar fallos a lo largo del ciclo de vida y permitiendo una corrección temprana de los mismos, de esta manera se remedian realizar cambios muy costosos que impliquen un retraso en la entrega. Por lo que a medida que se va teniendo un sistema más completo la carga de trabajo de pruebas aumentará, siendo relevante en la fase de transición donde se dispone de una versión casi completa del sistema.

En la *Ilustración 27* se muestra el flujo de trabajo que tienen las fases y disciplinas explicadas.

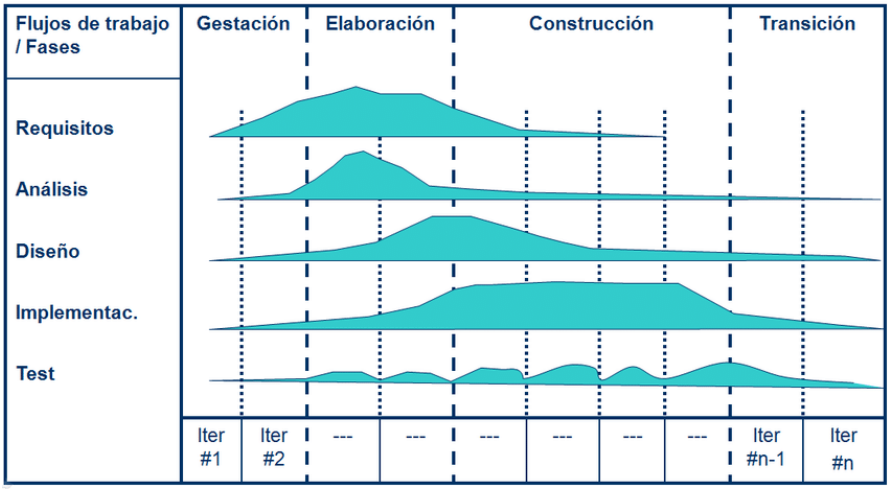


Ilustración 27: Fases y disciplinas del Proceso Unificado

6.2 Estimación del esfuerzo

Antes del comienzo de cualquier proyecto se tiene que realizar un estudio para estimar si su realización es viable y en caso de serlo cuál sería su coste.

Para ello en este proyecto en sus fases iniciales se hizo una estimación del esfuerzo en horas, mediante la métrica UCP(Use Case Points), donde se deben considerar actores, casos de uso y factores técnicos y del entorno para llevar a cabo el cálculo de las variables UUCP(Unadjusted Use Case Points), TCF(Technical Complexity Factor) y ECF(Environment Complexity Factor).

Para proceder al cálculo de estas variables expuestas y por ende de la estimación del esfuerzo, se ha hecho uso de la herramienta *EZEstimate*, explicada en el apartado *5-Técnicas y herramientas* de este mismo documento.

Si se desea conocer el proceso completo y la definición de cada factor de complejidad se puede consultar el *Anexo I – Plan del proyecto*.

En las siguientes ilustraciones se muestran los factores de complejidad introducidos y el resultado final.

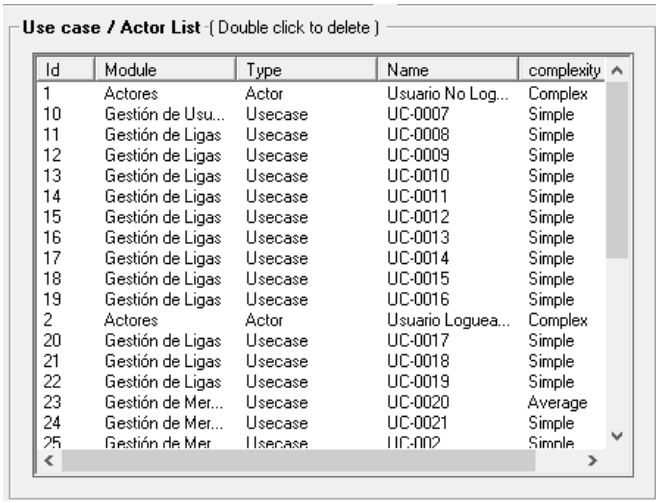


Ilustración 28: Complejidad casos de uso/actores

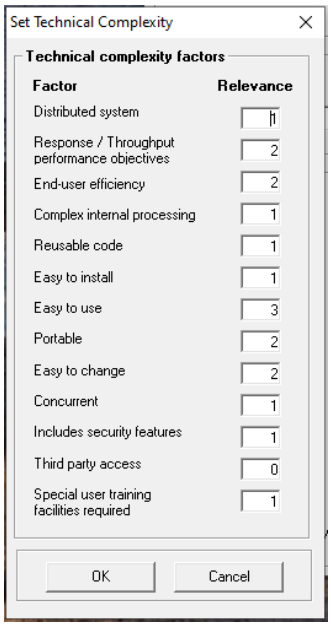


Ilustración 30: Factores de complejidad técnica

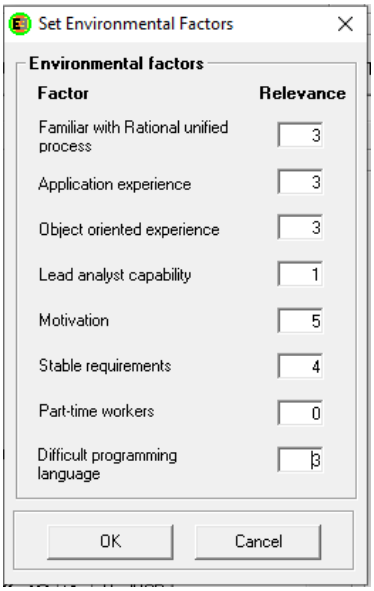


Ilustración 29: Factores del entorno

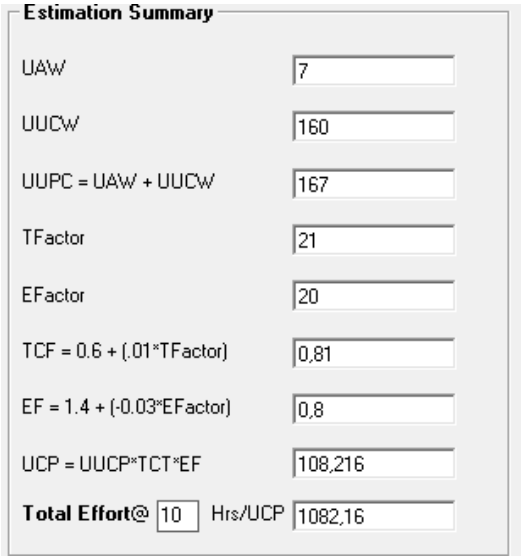


Ilustración 31: Variables y resultado final

La estimación resultante pasada de horas a días, sería de unos 135 días aproximadamente, teniendo en cuenta que el desarrollo lo produce un solo desarrollador dedicándole 8 horas diarias.

6.3 Planificación temporal

Una vez realizada la estimación de duración del proyecto se procede a realizar su planificación temporal. Es una labor de vital importancia ya que define la duración, organización y asignación de recursos a cada tarea a realizar en cada fase y disciplina del Proceso Unificado explicado anteriormente.

Para ello se ha utilizado la herramienta *Microsoft Project,* definida en la sección *5- Técnicas y herramientas.*

Para conocer más información y visualizar completamente la planificación y diagrama de Gantt, visitar el *Anexo I – Plan del proyecto*.

Como se observa en la siguiente imagen, la fecha de comienzo del proyecto fue el 31 de enero del año 2021 y la fecha en la que se tiene previsto acabar es la de martes 5 de julio de 2022, lo que lleva a un total de 111, 5 días, *Ilustración 33*, laborales para la realización de este. Comparando con el apartado anterior se rebaja la cifra estimada en 23 días.

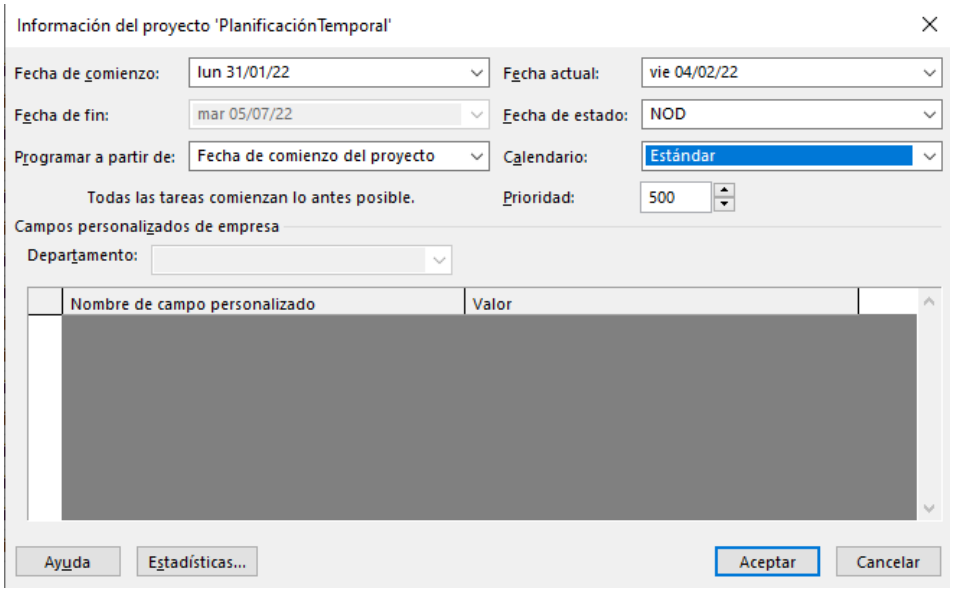


Ilustración 32: Información del proyecto

1	**Planificación Temporal**	**111,5 días?**	**lun 31/01/22**	**mar 05/07/22**	
2	**Inicio**	**14,5 días?**	**lun 31/01/22**	**vie 18/02/22**	
31	Fin Inicio	0 días	vie 18/02/22	vie 18/02/22	2
32	**Elaboración**	**31,5 días?**	**vie 18/02/22**	**lun 04/04/22**	**31**
79	Fin Elaboración	0 días	lun 04/04/22	lun 04/04/22	32
80	**Construcción**	**52,5 días?**	**mar 05/04/2**	**jue 16/06/22**	**79**
141	Fin Construcción	0 días	jue 16/06/22	jue 16/06/22	80
142	**Transición**	**13 días?**	**jue 16/06/22**	**mar 05/07/22**	**141**
163	Fin Transición	0 días	mar 05/07/2	mar 05/07/22	142
164	Fin Proyecto	0 días	mar 05/07/2	mar 05/07/22	1

Ilustración 33: Tiempo total y de cada fase

En las posteriores ilustraciones se van a mostrar ejemplos de la organización de las tareas y del diagrama de Gantt resultante.

80	**Construcción**	**52,5 días?**	**mar 05/04/2**	**jue 16/06/22**	**79**	
81	**Iteración 1**	**11 días?**	**mar 05/04/2**	**mar 19/04/22**		
82	**Modelado del negocio**	**0,5 días?**	**mar 05/04/2**	**mar 05/04/22**		
83	Investigación de recursos y herramientas	0,5 días?	mar 05/04/2	mar 05/04/22		Jairo Sánchez
84	**Requisitos**	**0,5 días?**	**mar 05/04/2**	**mar 05/04/22**		
85	Revisión de requisitos funcionales	0,5 días?	mar 05/04/2	mar 05/04/22	83	Jairo Sánchez
86	**Análisis**	**1 día?**	**mié 06/04/2**	**mié 06/04/22**		
87	Diagramas de comunicación	1 día?	mié 06/04/2	mié 06/04/22	85	Jairo Sánchez
88	**Diseño**	**1,5 días?**	**jue 07/04/22**	**vie 08/04/22**		
89	Patrón de diseño y patrón arquitectónico	1,5 días?	jue 07/04/22	vie 08/04/22	87	Jairo Sánchez
90	**Implementación**	**7 días**	**vie 08/04/22**	**mar 19/04/22**		
91	Elaboración subsistema "Gestión de Usuarios"	7 días	vie 08/04/22	mar 19/04/22	89	Jairo Sánchez
92	**Pruebas**	**0,5 días?**	**mar 19/04/2**	**mar 19/04/22**		
93	Pruebas atómicas de cada componente/vista	0,5 días?	mar 19/04/2	mar 19/04/22	91	Jairo Sánchez
94	Pruebas atómicas de cada servicio	0,5 días	mar 19/04/2	mar 19/04/22	91	Jairo Sánchez
95	Fin Iteración 1	0 días	mar 19/04/2	mar 19/04/22	81	
96	**Iteración 2**	**16 días?**	**mié 20/04/2**	**mié 11/05/22**	**95**	
97	**Modelado del negocio**	**0,5 días**	**mié 20/04/2**	**mié 20/04/22**		
98	Investigación sobre recursos y herramientas	0,5 días	mié 20/04/2	mié 20/04/22		Jairo Sánchez
99	**Requisitos**	**0,5 días**	**mié 20/04/2**	**mié 20/04/22**		
100	Revisión requisitos no funcionales	0,5 días	mié 20/04/2	mié 20/04/22		Jairo Sánchez
101	**Análisis**	**0,5 días?**	**mié 20/04/2**	**mié 20/04/22**		
102	Vista de arquitectura	0,5 días?	mié 20/04/2	mié 20/04/22		Jairo Sánchez
103	**Diseño**	**1 día?**	**mié 20/04/2**	**jue 21/04/22**		
104	Refinamiento casos de uso diseño	1 día?	mié 20/04/2	jue 21/04/22	102	Jairo Sánchez
105	**Implementación**	**14 días**	**jue 21/04/22**	**mié 11/05/22**		
106	Elaboración subsistema "Gestión de Ligas"	14 días	jue 21/04/22	mié 11/05/22	104	Jairo Sánchez
107	**Pruebas**	**0,5 días**	**mié 11/05/2**	**mié 11/05/22**		
108	Pruebas atómicas de cada componente/vista	0,5 días	mié 11/05/2	mié 11/05/22	106	Jairo Sánchez
109	Pruebas atómicas de cada servicio	0,5 días	mié 11/05/2	mié 11/05/22	106	Jairo Sánchez
110	Fin Iteración 2	0 días	mié 11/05/2	mié 11/05/22	96	

Ilustración 34: Ejemplo de organización y planificación de tareas

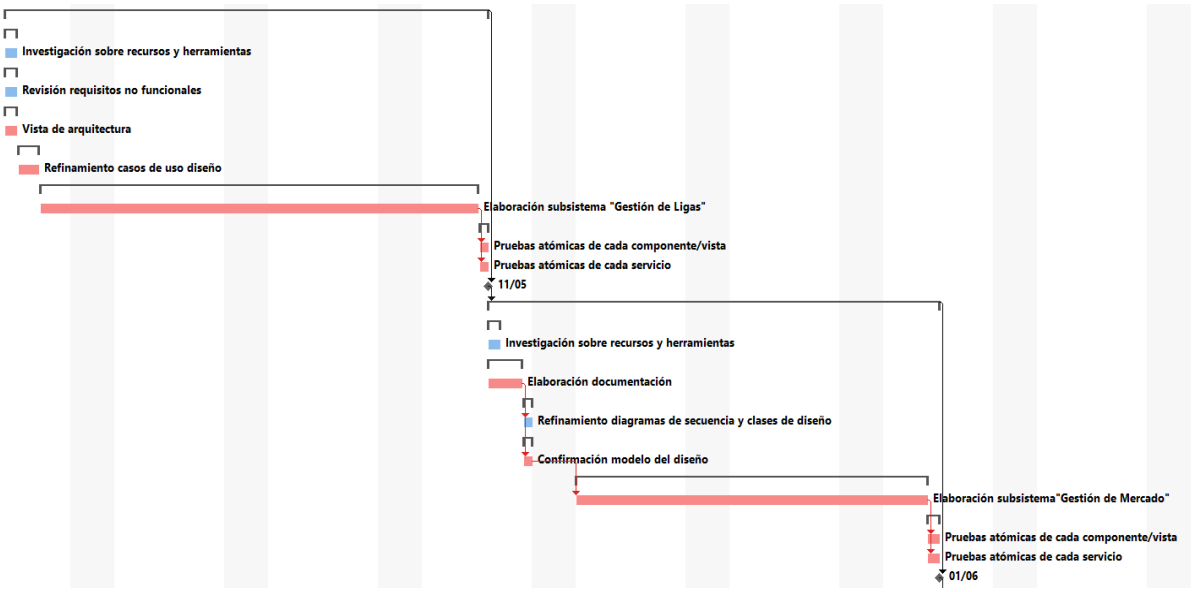


Ilustración 35: Ejemplo Diagrama de Gantt

6.4 Especificación de requisitos

En este apartado se va a cometer un resumen de la fase de especificación de requisitos del sistema software. Para ello se ha seguido la estructura proporcionada por la *metodología de Durán y Bernárdez* para la obtención y documentación de requisitos, siendo la siguiente:

- Participantes en el proyecto.
- Objetivos del sistema.
- Catálogo de requisitos del sistema.
    - Requisitos de información.
    - Requisitos funcionales.
        - Diagramas de casos de uso.
        - Definición de actores.
        - Casos de uso del sistema.
    - Requisitos no funcionales.
- Matriz de rastreabilidad objetivos/requisitos.

Para una completa información y visualización de los diagramas resultantes en esta fase y el desarrollo de requisitos consultar el *Anexo II – Especificación de Requisitos Software*.

6.4.1 Participantes en el proyecto

La primera tarea que abordar es la definición de los participantes en el proyecto y los roles que estos tomarán en él.

Para llevar a cabo el desarrollo se cuenta con cuatro participantes, siendo uno de ellos el alumno encargado de la realización del Trabajo de Fin de Grado y los otros tres resultantes son los tutores encargados de la supervisión del trabajo.

6.4.2 Objetivos del sistema

Seguidamente, se definen los objetivos que el sistema tiene que cumplir para proporcionar un correcto funcionamiento.

Lo objetivos que se han determinado son :

- Gestión de los usuarios.
- Gestión de las ligas.
- Gestión de las jornadas.
- Gestión del roster.
- Gestión del mercado.
- Gestión del calendario.
- Gestión de las noticias.

Estos objetivos están recogidos en tablas como la mostrada en la *Tabla 1*, especificando cada una de sus características, en esta se muestra el *OBJ-0001 Gestión de los usuarios*.

OBJ-0001	Gestión de los usuarios
Versión	1.0
Autores	Jairo Sánchez García
Fuentes	
Descripción	El sistema deberá permitir a los usuarios darse de alta en el sistema, entrar y salir de él. Además debe permitir la modificación de sus datos.
Subobjetivos	Ninguno
Importancia	Vital
Urgencia	Inmediata
Estado	Validado
Estabilidad	Alta
Comentarios	Ninguno

Tabla 1: Ejemplo tabla Objetivos del sistema OBJ-0001 Gestión de los usuarios

6.4.3 Catálogo de requisitos del sistema

A continuación, se realiza la elicitación de los diversos requisitos que existen y son necesarios para satisfacer las necesidades finales.

6.4.3.1 Requisitos de información

Este tipo de requisitos indican los datos que el sistema debe almacenar.

Los requisitos de información definidos en el proyecto son los siguientes:

- Información sobre los usuarios.
- Información sobre los jugadores.
- Información sobre las ligas.
- Información sobre los roster.
- Información sobre las jornadas.
- Información sobre las noticias.
- Información sobre el mercado.

Cada uno de estos requisitos de información ha sido especificado en tablas como la mostrada en la *Tabla 2*, donde se detalla las características que tiene el requisito de información *IRQ-0006 Información sobre las noticias*.

IRQ-0006	Información sobre las noticias	
Versión	1.0	
Autores	Jairo Sánchez García	
Fuentes		
Dependencias	Ninguno	
Descripción	El sistema deberá almacenar información correspondiente a las noticias generadas en el sistema. En concreto:	
Datos específicos	• Id • Id de liga • Día • Mes • Año • Información • Tipo • Nombre usuario	
Tiempo de vida	**Medio**	**Máximo**
Ocurrencias simultáneas	**Medio**	**Máximo**
Importancia	Importante	
Urgencia	Inmediata	
Estado	Validado	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla 2: Ejemplo Requisito de Información IRQ-0006 Información sobre las noticias

6.4.3.2 Requisitos funcionales

En esta sección se definen los requisitos funcionales que el sistema deberá proporcionar y la manera en que actúan ante determinados escenarios.

Diagrama de paquetes

Primeramente se deben estructurar los paquetes que conforman el sistema software. Para ello se realiza un diagrama de paquetes donde se muestra la jerarquía de estos. Los paquetes que se tienen en el sistema son: Gestión de Usuarios, Gestión de Ligas, Gestión de Mercado y Gestión de Roster. El diagrama se puede ver en la *Ilustración 36*.

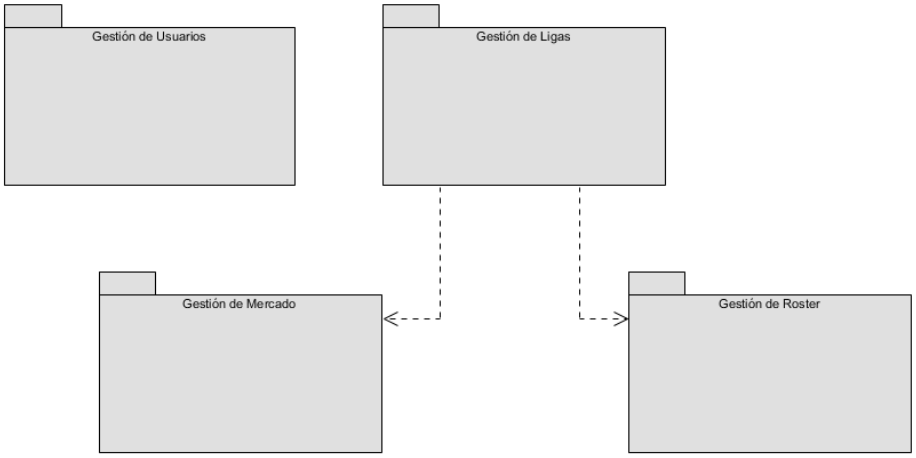


Ilustración 36: Diagrama de Paquetes

Definición de actores

Seguidamente, se detallan los actores que participarán en el sistema, siendo estos una representaciones de los supuestos usuarios que van a interaccionar con el sistema.

Los actores especificados para este sistema son:

- Usuario No Logueado
- Usuario Logueado
- Timer

La representación de estos en UML se puede observar en la *Ilustración 37*.

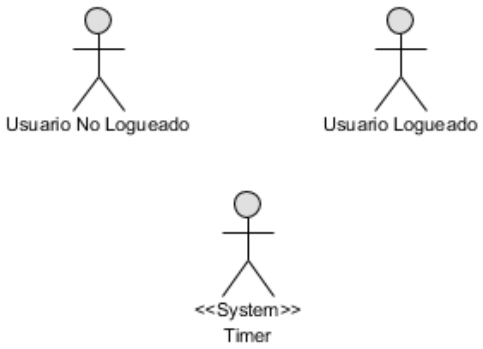


Ilustración 37: Actores del sistema

La especificación de estos actores se lleva a cabo en tablas como Tabla 3, donde se explica el *ACT-0001 Usuario No Logueado*.

ACT-0001	Usuario No Logueado
Versión	1.0
Autores	Jairo Sánchez García
Fuentes	
Descripción	Actor que va a interactuar con el sistema pero sin estar dado de alta en él.
Comentarios	Ninguno

Tabla 3: Ejemplo Actor ACT-0001 Usuario No Logueado

Casos de uso del sistema

Se especifican los casos de uso del sistema, definiendo los escenarios sobre donde se desarrollarán e interaccionarán con los actores previamente definidos.

Para hacer más efectiva esta tarea se ha llevado a cabo la realización de un diagrama de casos de uso para cada paquete del sistema.

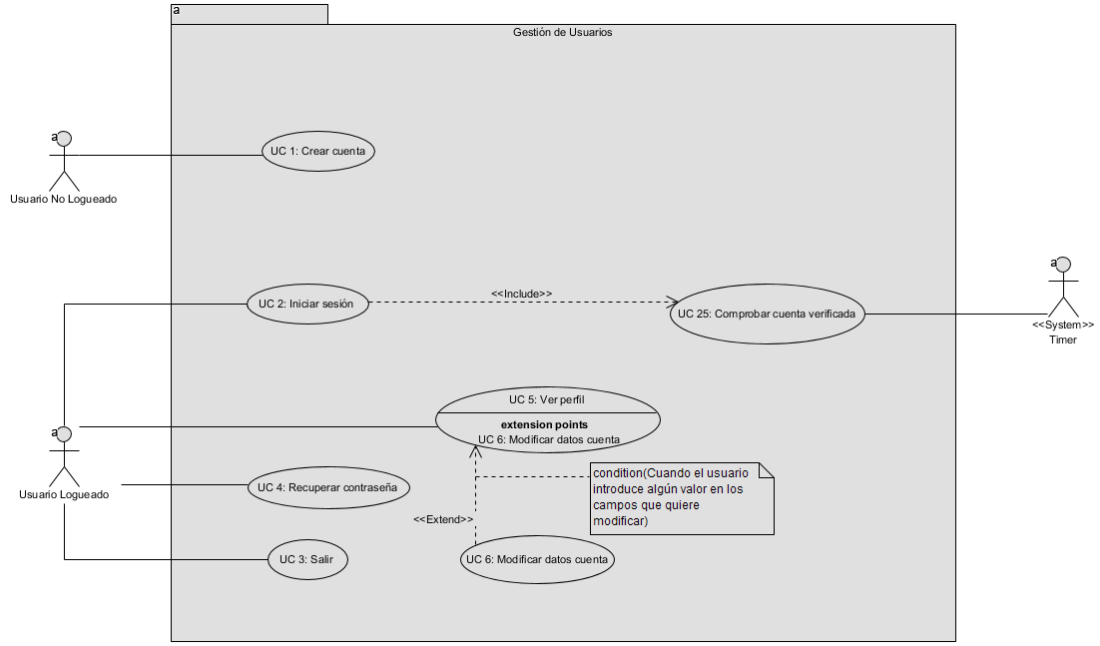


Ilustración 38: Diagrama de Casos de uso, Gestión de Usuarios

En la *Ilustración 38*, se expone el diagrama de casos de uso resultante del paquete *Gestión de Usuarios*.

Cada caso de uso se define en tablas donde se especifica la secuencia de pasos que debe seguir para su desarrollo, así como su descripción, dependencias y demás características.

Se muestra como ejemplo en la *Tabla 4*, la especificación del caso de uso *UC-0001-Crear cuenta.*

UC-0001	**Crear cuenta**	
Versión	1.0	
Autores	Jairo Sánchez García	
Fuentes		
Objetivos asociados	• OBJ-0001 Gestión de los usuarios	
Casos de uso asociados		
Descripción	El sistema deberá comportarse tal como se describe a continuación cuando un usuario no logueado seleccione la opción '*Registrarse*'.	
Precondición	Ninguna	
Secuencia normal	**Paso**	**Acción**
	1	El sistema mostrará al usuario un formulario con los campos a rellenar, como nickname, contraseña y logo.
	2	El actor '*Usuario No Logueado*' (ACT-0001), tendrá que rellenar dichos campos con los datos correctos y realizar la opción '*Registro*'.
	3	Si los campos han sido rellenados con datos correctos, seguidamente el sistema enviará un correo para la confirmación de la cuenta.
Postcondición	Ninguna	
Excepciones	**Paso**	**Acción**
	3	En caso de que algún campo tenga un valor incorrecto o el usuario ya posea una cuenta con ese email en el sistema, se le notificará el error. Este caso de uso se repetiría hasta que decida salir de la zona de registro, o se complete satisfactoriamente.
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Frecuencia esperada		
Importancia	Vital	
Urgencia	Inmediata	
Estado	Válido	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla 4: Ejemplo Caso de uso UC-0001 Crear cuenta

6.4.3.3 Requisitos no funcionales

Para finalizar la fase de especificación de requisitos se describen los requisitos no funcionales del sistema. Estos van a imponer al sistema unas restricciones en el diseño y en la implementación. El cumplimiento de estos le da al sistema un cierto grado de calidad, así mismo, el no cumplimiento de los requisitos no funcionales puede provocar que no se satisfagan las necesidades necesarias.

En la *Tabla 5* se muestra un ejemplo de definición de requisito no funcional, donde se detalla el *NFR-0001 Portabilidad*.

NFR-0001	**Portabilidad**
Versión	1.0
Autores	Jairo Sánchez García
Fuentes	
Objetivos asociados	Ninguno
Requisitos asociados	Ninguno
Descripción	El sistema debe funcionar correctamente tanto desde la web como desde Android e IOS.
Importancia	Vital
Urgencia	Inmediata
Estado	Válido
Estabilidad	Alta
Comentarios	Ninguno

Tabla 5: Ejemplo Requisito no funcional NFR-0001 Portabilidad

6.5 Análisis de requisitos

En este punto se realizará un resumen de los procedimientos llevados a cabo y de los modelos obtenidos en la fase de análisis de requisitos del sistema. El objetivo principal de esta fase es el refinamiento de los casos de uso especificados en el *Anexo II – Especificación de Requisitos del Software*. Si se quiere conocer toda la información acerca del proceso completo al análisis de requisitos definido se encuentra en el *Anexo III – Análisis de Requisitos*.

6.5.1 Modelo de dominio

Uno de los aspectos claves de la fase de análisis de requisitos es realizar una correcta conceptualización de los objetos que participan en el sistema de manera física y relaciones que existen entre sí. A partir de los requisitos de información especificados en el *Anexo II* anteriormente comentado, , se va a determinar las entidades del modelo de negocio.

Para representar el modelo del dominio se realiza el siguiente diagrama de clases:

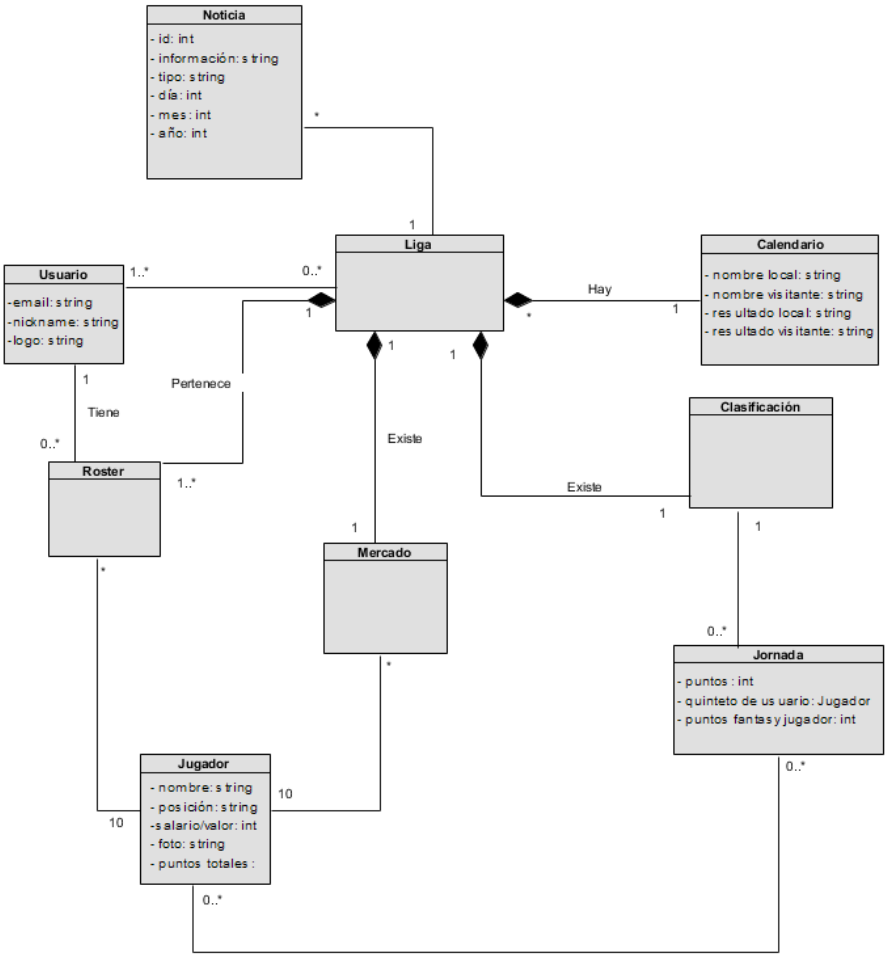


Ilustración 39: Diagrama de clases

6.5.2 Realización de casos de uso-análisis

Seguidamente de la conceptualización realizada en el apartado anterior se va a explicar el comportamiento dinámico de los casos de uso y la secuencia de mensajes intercambiado por los objetos de los diferentes paquetes que conforman el sistema mediante la utilización de diagramas de secuencia.

En la *Ilustración 40* se muestra un ejemplo de diagrama de secuencia correspondiente al caso de uso *UC-0001 Crear cuenta*.

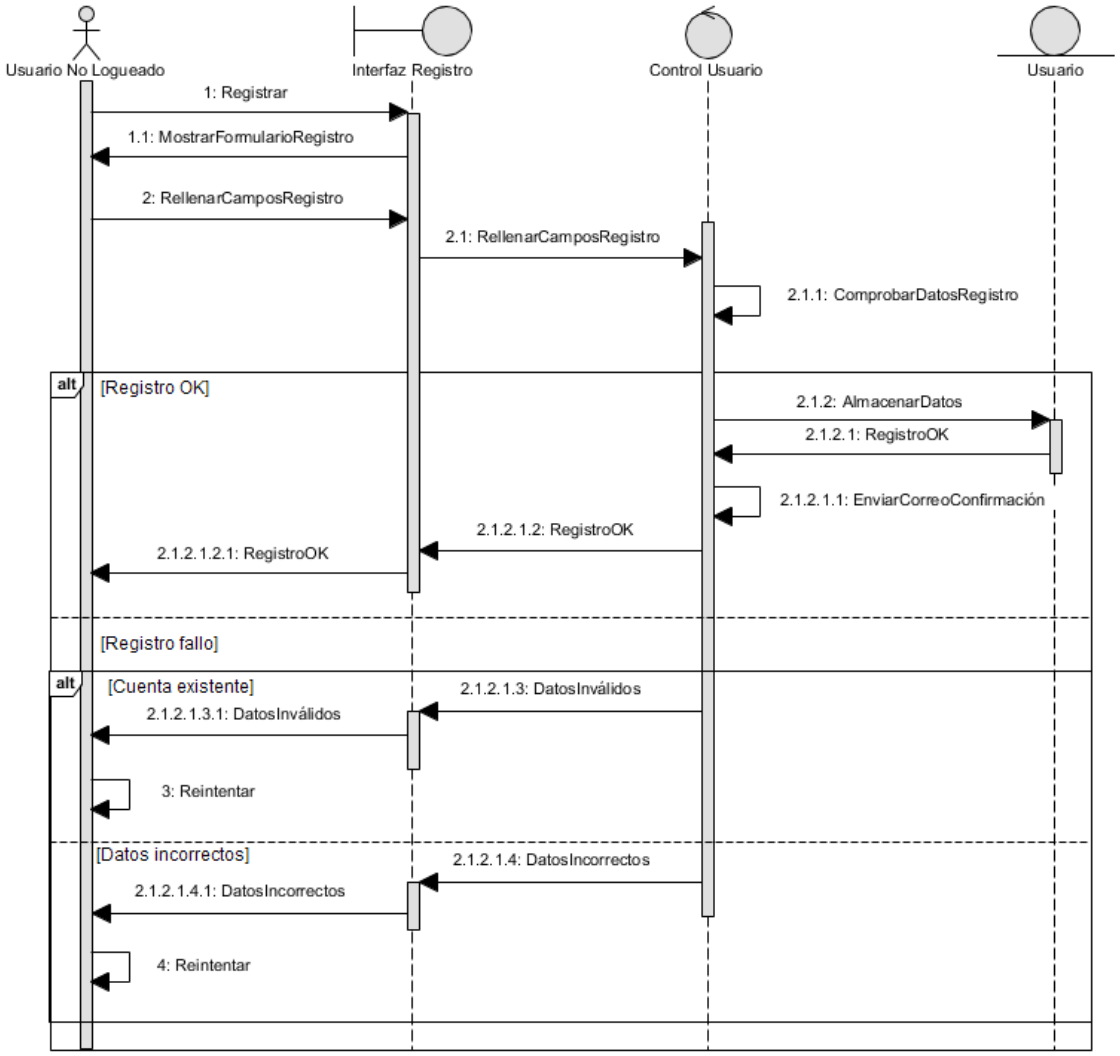


Ilustración 40: Diagrama de secuencia UC-0001 Crear cuenta

6.5.3 Clases de análisis

La siguiente tarea correspondiente a esta fase una vez finalizados los diagramas de secuencia es la definición de cómo se relacionan y comunican entre sí las clases de análisis que forman parte de los diagramas explicados en la sección anterior.

Mediante la *Ilustración 41* se puede ver un ejemplo de estos diagramas de comunicación relacionado con el paquete *Gestión de Usuarios*.

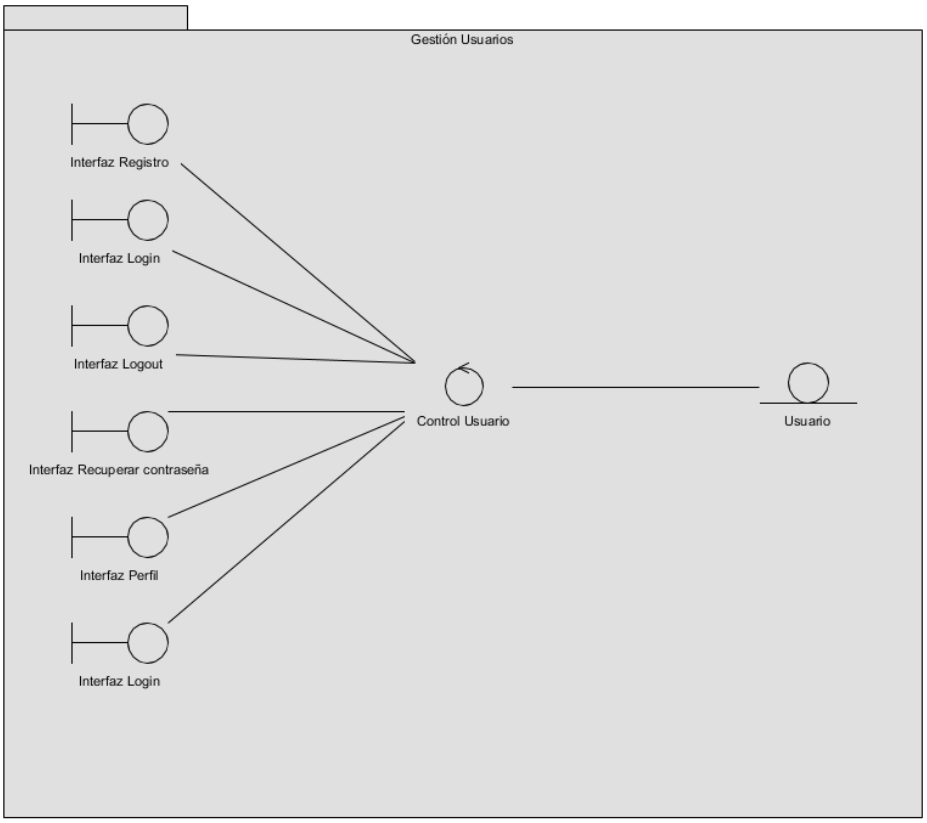


Ilustración 41: Diagrama de comunicación Paquete Gestión de Usuario

6.5.4 Vista arquitectónica del modelo de análisis

Para finalizar las tareas de análisis, se obtiene como resultado una primera vista de la arquitectura del sistema. Siguiendo el modelo arquitectónico *Modelo-Vista-Controlador*, se permite desacoplar la lógica de control de la interfaz de usuario y de los datos de la aplicación.

En la siguiente ilustración se van a mostrar las clases de análisis definidas en el punto anterior agrupadas en una vista de arquitectura del modelo de análisis.

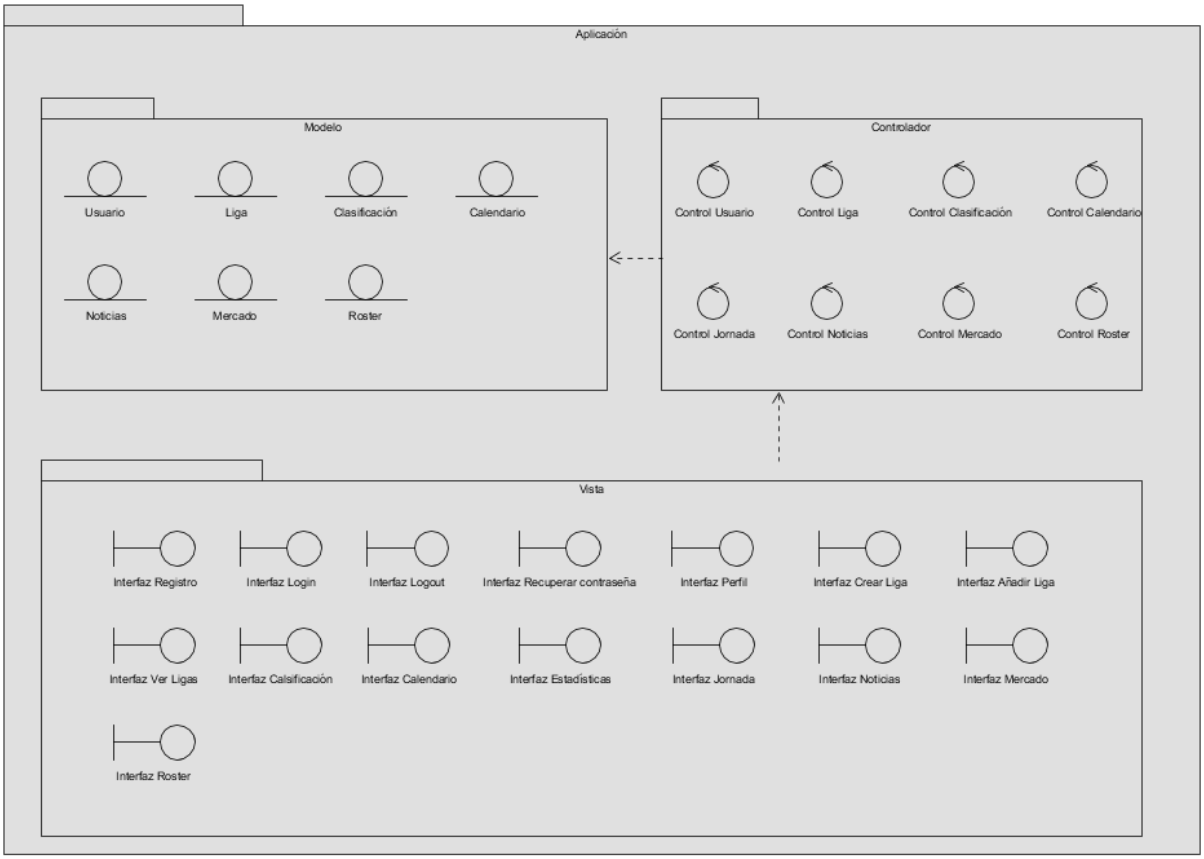


Ilustración 42: Vista de arquitectura modelo de análisis

6.6 Diseño del sistema

Se va a mostrar de manera concisa el proceso realizado durante la fase de diseño del sistema. Tras su finalización se debe conseguir una visión cercana al resultado de implementación del sistema. Para ello se procede a hacer un refinamiento de las especificaciones llevadas a cabo en la etapa previa desarrollada en el *Anexo III – Análisis de Requisitos*.

Para conocer el proceso completamente desarrollado de esta fase consultar el *Anexo IV – Diseño del Sistema Software*.

6.6.1 Modelo de diseño

El modelo de diseño permite modelar el sistema hacia un nivel más bajo cercano a la programación del sistema. A partir de esta fase se tendrá una especificación detallada de los paquetes, clases de diseño y casos de uso, así como una abstracción de la implementación del sistema.

En esta fase se deben fijar los patrones arquitectónicos y de diseño sobre los que se le dará forma al sistema.

6.6.1.1 Patrones arquitectónicos

La elección del patrón arquitectónico que se debe seguir para desarrollar el sistema es un punto relevante del diseño. Se deben de tener en cuenta varios aspectos para su elección, como el número y forma de los paquetes que componen el sistema y la relación existente entre ellos.

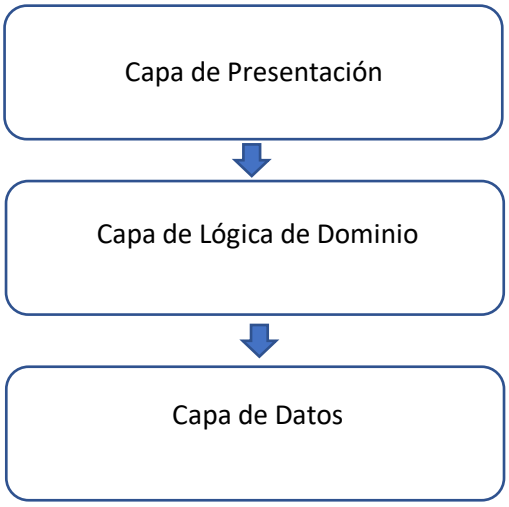
El patrón que según bajo mi criterio es el que mejor se adapta al sistema software a desarrollar es el patrón por capas. Así mismo debido a la composición que tiene el sistema una parte interna de él seguirá el patrón MVVM(Model-View-ViewModel).

Patrón por capas

El patrón por capas permite organizar los componentes de un sistema en diferentes niveles con responsabilidades distintas. Las capas superiores colaboran con las inferiores, en el caso contrario se necesita de interfaces bien definidas para llevar a cabo la comunicación, de modo que las capas superiores sólo conocen las interfaces de comunicación de las inferiores, lo cual resulta muy útil a la hora de realizar cambios en alguna capa.

El sistema estará formado por las siguientes capas:

- **Capa de Presentación:** Representa la interacción entre el usuario y el sistema a través de interfaces de usuario. No debe de tener mucha funcionalidad, solamente mostrar y recoger datos del usuario y validaciones simples.
- **Capa de Lógica de Dominio:** Funcionalidad de la aplicación. Contiene los cálculos basados en la información dada por el usuario y datos almacenados.
- **Capa de Datos:** Lógica de comunicación con otros sistemas, como una API externa, o con la base de datos.



Patrón MVVM

Este patrón arquitectónico es el que siguen los marcos de trabajo *Vue.js* e *Ionic.* La estructura y modo de funcionamiento es similar a la que se presenta en *Modelo-Vista-Controlador.*

Siguiendo este patrón se separa la lógica de negocios con la presentación de la aplicación. Se tienen tres componentes principales cada uno con responsabilidades distintas, estos componentes son:

- **Modelo:** Contiene clases no visuales en donde se encapsulan los datos que utiliza la aplicación. Se comunica con servicios para proporcionar el acceso a datos.
- **Vista:** Define la estructura y apariencia que ve el usuario, no debe contener código relacionado con la lógica de negocio.
- **Modelo de vista:** Implementa propiedades a las que la vista puede enlazar datos y notifica a esta los cambios de estado. Las acciones que ofrece el modelo de vista definen la funcionalidad que ofrece la interfaz de usuario. Separa el modelo de la vista.

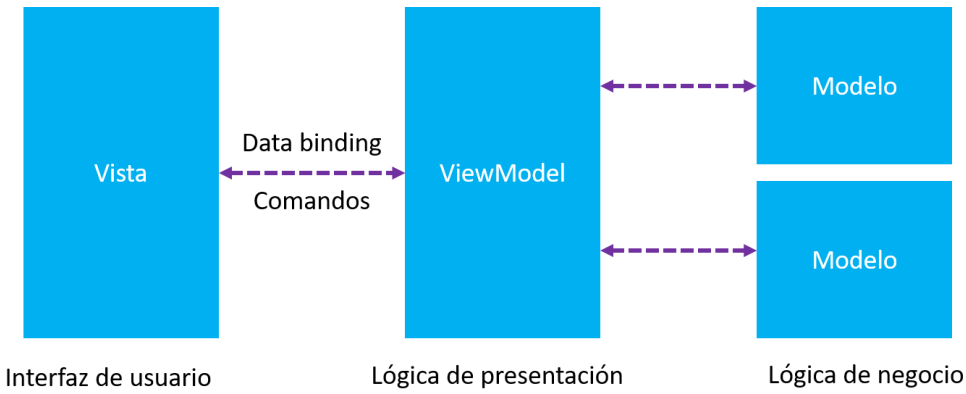


Ilustración 43: Esquema Patrón MVVM

Patrón Publisher-Subscriber

Es un patrón de intercambio de mensajes en el que participan emisores, publishers, y receptores, subscribers.

El funcionamiento del patrón consiste en que el publicador envía mensajes asíncronos hacia algún servidor, ya que desconoce el paradero y existencia de los subscriptores. Estos mensajes publicados se diferencias por clases, los subscriptores expresan interés en uno o varios tipos de clases, y sólo reciben los mensajes que corresponden a las clases con interés sin conocer al publicador. Esta relación anónima permite tener una alta escalabilidad.

El esquema de funcionalidad del patrón arquitectónico *Publisher-Subscriber* se encuentra en la *Ilustración 44*.

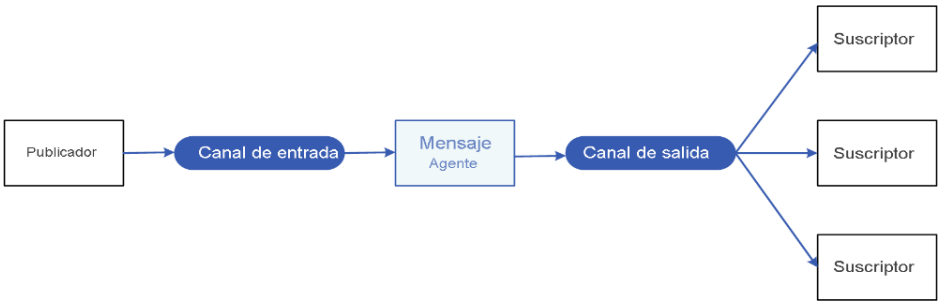


Ilustración 44: Esquema Patrón Publisher-Subscriber

6.6.1.2 Subsistemas de diseño

Se decide descomponer el sistema dos paquetes principales distintos dentro del modelo de diseño. Estos a su vez estarán compuestos por una serie de subpaquetes. En la *Ilustración 45* se observa su disposición.

Aplicación Web

Este paquete contiene la parte con la que el usuario interactuará. Se tiene un subpaquete llamado *Components* que tomará el papel de modelo de vista del patrón MVVM compuesto por varios archivos .vue, por otra parte se tiene otro subpaquete llamado *Views* el cual contiene los archivos .vue con las interfaces de usuario. El modelo en este paquete no existe como tal debido a la propiedad data que se dispone en Vue.

Servidor

Por otro lado el paquete servidor está formado por tres subpaquetes, *Controlador*, *Scripts* y *Modelo*. El subpaquete *Controlador* es el middleware encargado de construir los endpoints, a través de los cuales los clientes realizarán las llamadas al servidor. Este subpaquete además, recoge las peticiones por parte del cliente, las procesa y responde al cliente, por otra parte tendrá comunicación con el modelo. El subpaquete *Scripts* contendrá las clases que se deben ejecutar de manera automática cada cierto tiempo. El subpaquete *Modelo* hará la comunicación con la base de datos y API externas y mantendrá la información de los datos.

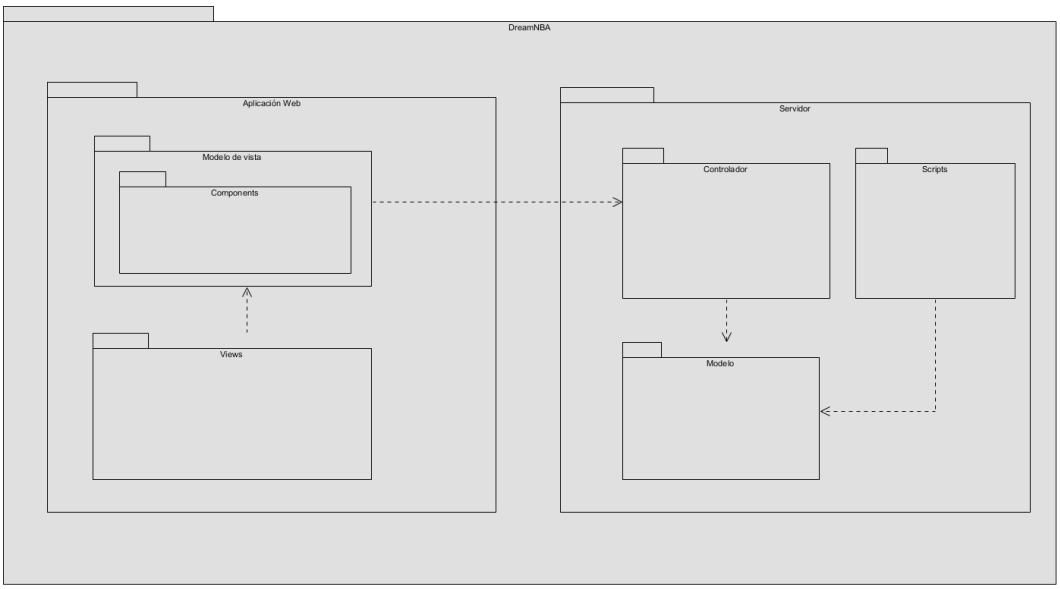


Ilustración 45: Modelo de diseño

6.6.1.3 Clases de diseño

Se define el contenido de cada capa de la arquitectura, además de las distintas clases de diseño que se encuentran dentro de cada paquete, indicando los métodos y atributos que tiene cada componente de este.

Siguiendo con las diferentes estructuras se va a presentar primero el paquete *Aplicación Web* que sigue el patrón *Modelo-Vista-Modelo de vista* pero con la modificación de la ausencia del modelo representado con atributos de la propiedad data en los componentes del modelo de vista. Por último, se exhibe el paquete *Servidor*, el cual ocupa las capas de negocio y datos de la arquitectura global del sistema, la capa de presentación la formará los subpaquetes de la *Aplicación Web.*

Mediante las siguientes ilustraciones se mostrará algunos ejemplos de las clases de diseño definidas en cada capa.

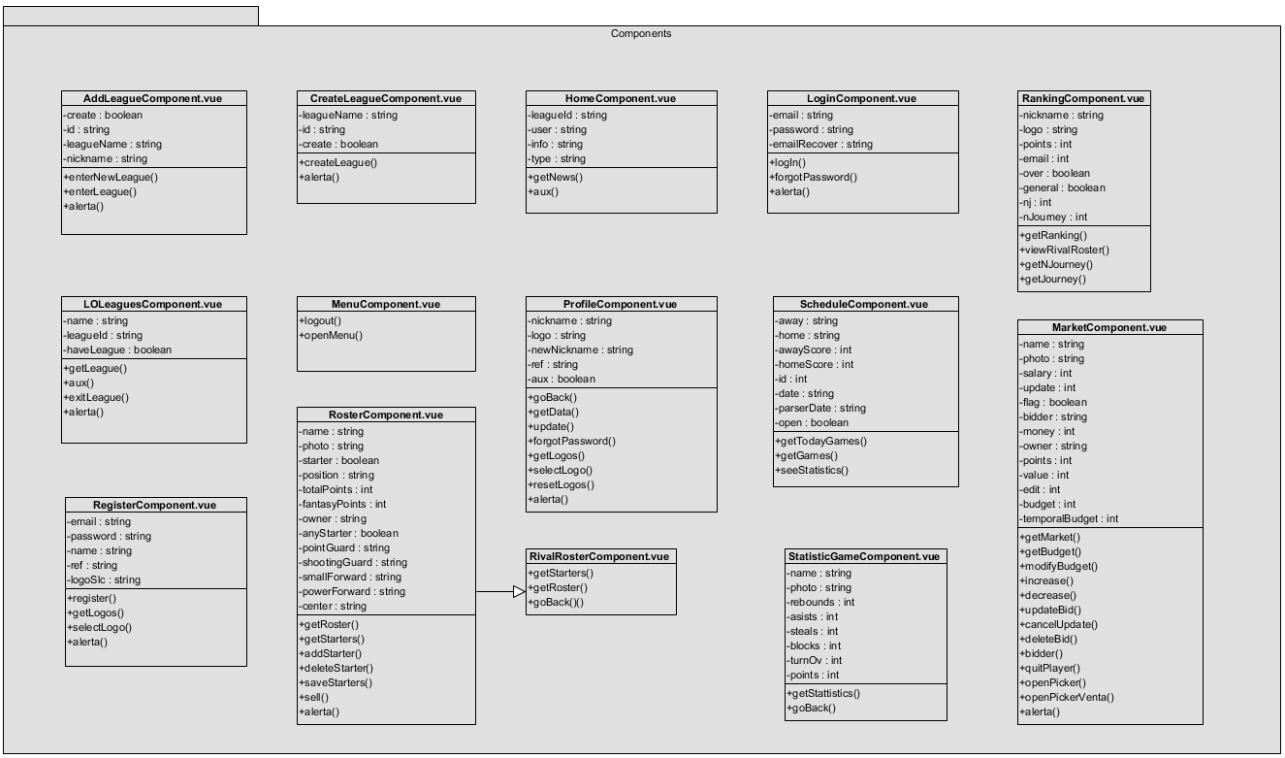


Ilustración 46: Clases de diseño Modelo de Vista

En la *Ilustración 46* se encuentran las clases de diseño definidas dentro del Modelo de Vista de la capa de presentación.

Mientras que en la siguiente, *Ilustración 47*, están situadas las clases de diseño del paquete *Controlador* de la capa lógica de dominio.

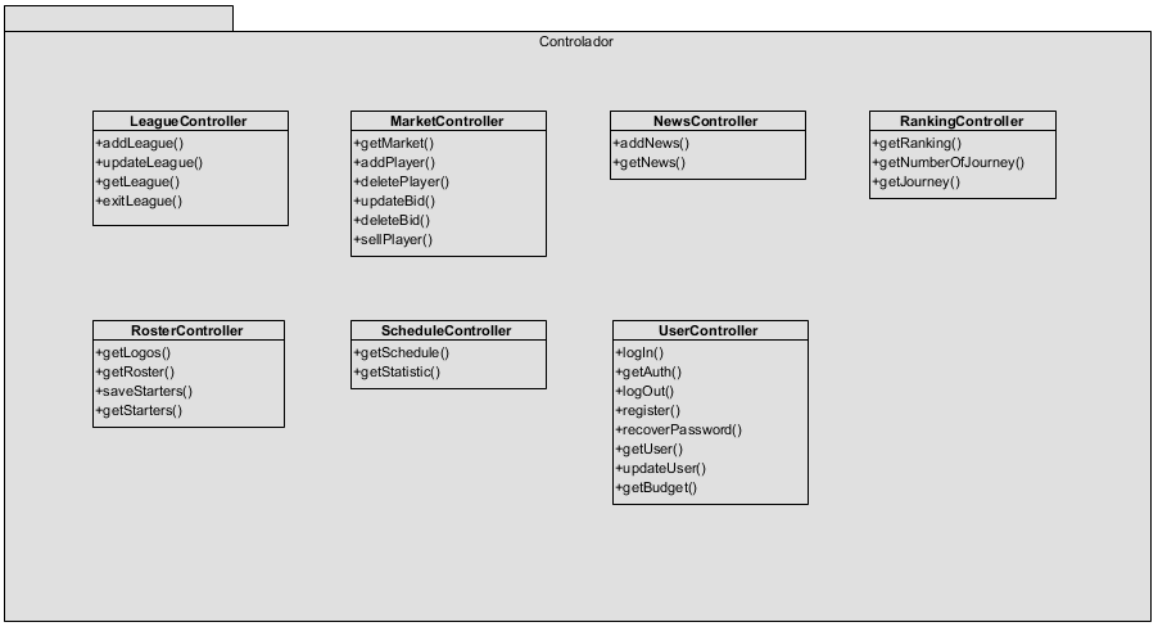


Ilustración 47: Clases de diseño Controlador

6.6..1.4 Vista arquitectónica del modelo de diseño

Mediante la realización de la vista arquitectónica se pretender definir una visión global del sistema. Para ello con los paquetes que se definieron en el modelo del diseño se van a ordenar en la capa correspondiente definiendo así la arquitectura.

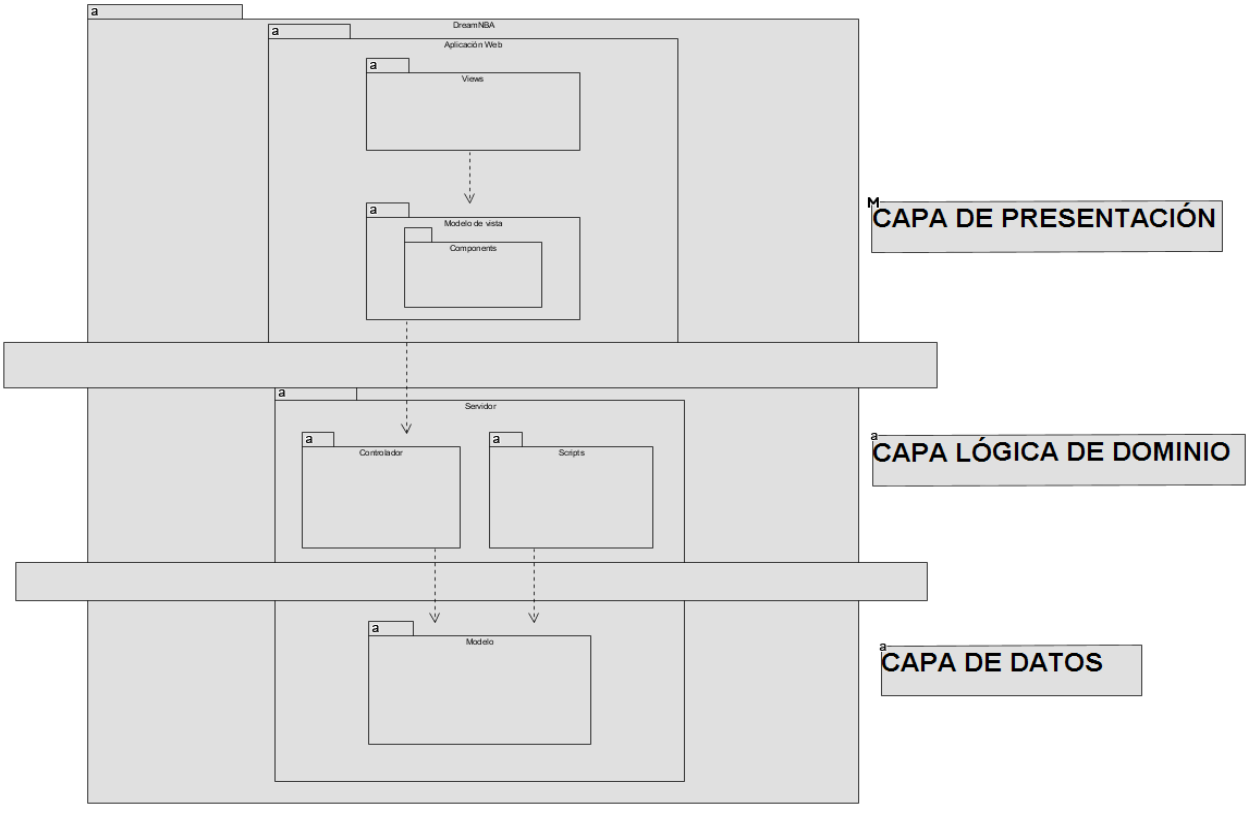


Ilustración 48: Vista arquitectónica del modelo de diseño

6.6.1.5 Realización de Casos de Uso

Para la realización de casos de uso en la etapa de diseño se produce un refinamiento de los diagramas de secuencia definidos en el *Anexo III* mencionado previamente.

En estos diagramas de secuencia se busca un nivel de detalle similar al que se llevaría a cabo en la implementación por ello los nombres de las clases, métodos y atributos que aparecen en ellos deben ser los del software resultante.

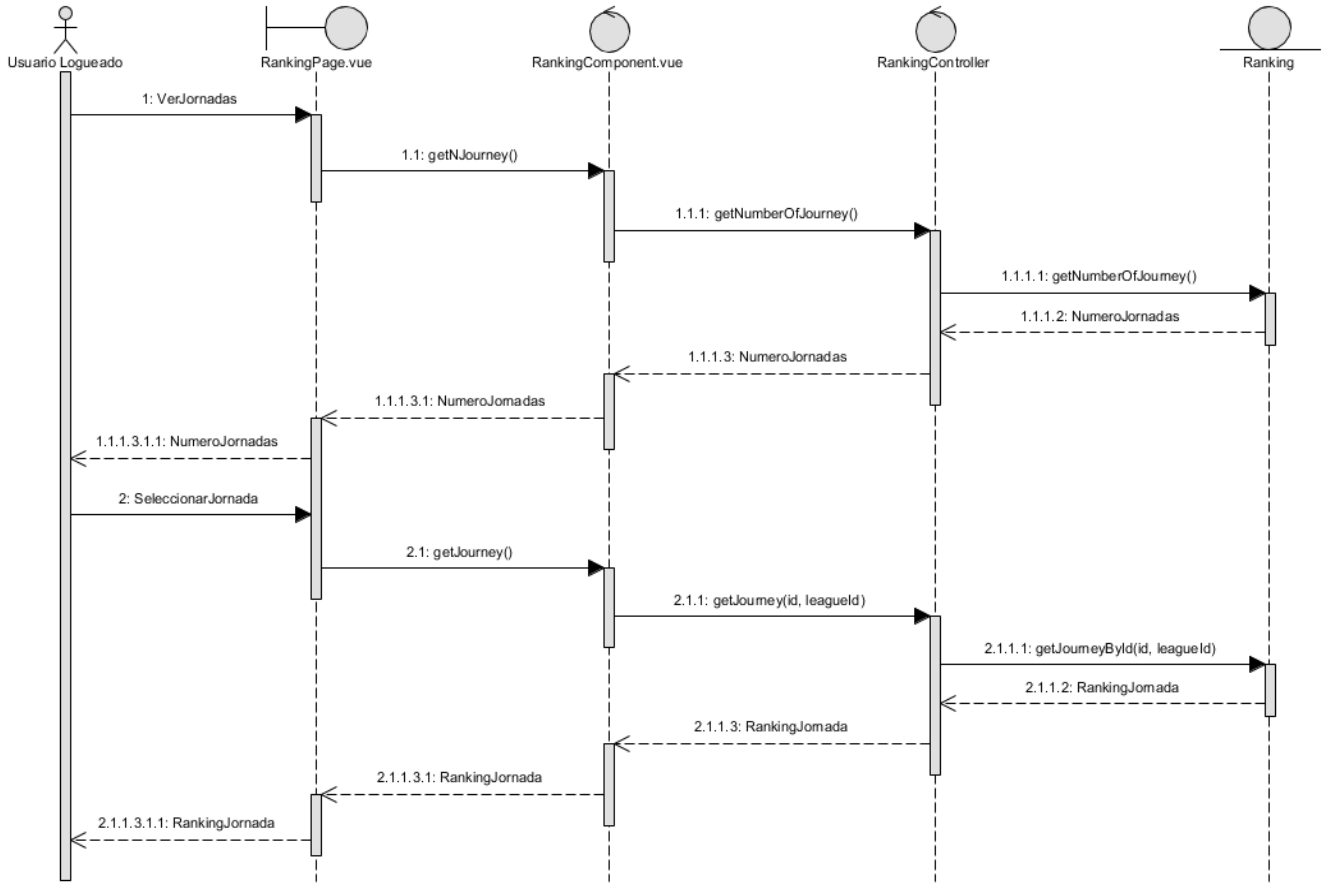


Ilustración 49: Diagrama de secuencia-diseño UC-0013 Ver clasificación por jornada

6.6.1.6 Diseño base de datos

Debido a la necesidad de recuperación y almacenamiento de datos que requiere la aplicación es importante la realización de un buen diseño de la base de datos.

El sistema va a utilizar una variación de base de datos NoSQL, una base de datos documental utilizada por *Firebase,* herramienta definida en el apartado *4 – Conceptos Teóricos*. Los datos se conceptualizarán como un árbol JSON dentro de las colecciones y documentos.

A continuación se presenta el diagrama de base de datos, como la leyenda indica en la *Ilustración 50*, los campos resaltados en rojo hacen referencia al identificador del documento dentro de la colección. Así mismo, los campos destacados en verde y blanco indican

los campos que se encuentran dentro de un mapa o un array, esto es debido a la estructura de árbol JSON comentada anteriormente.

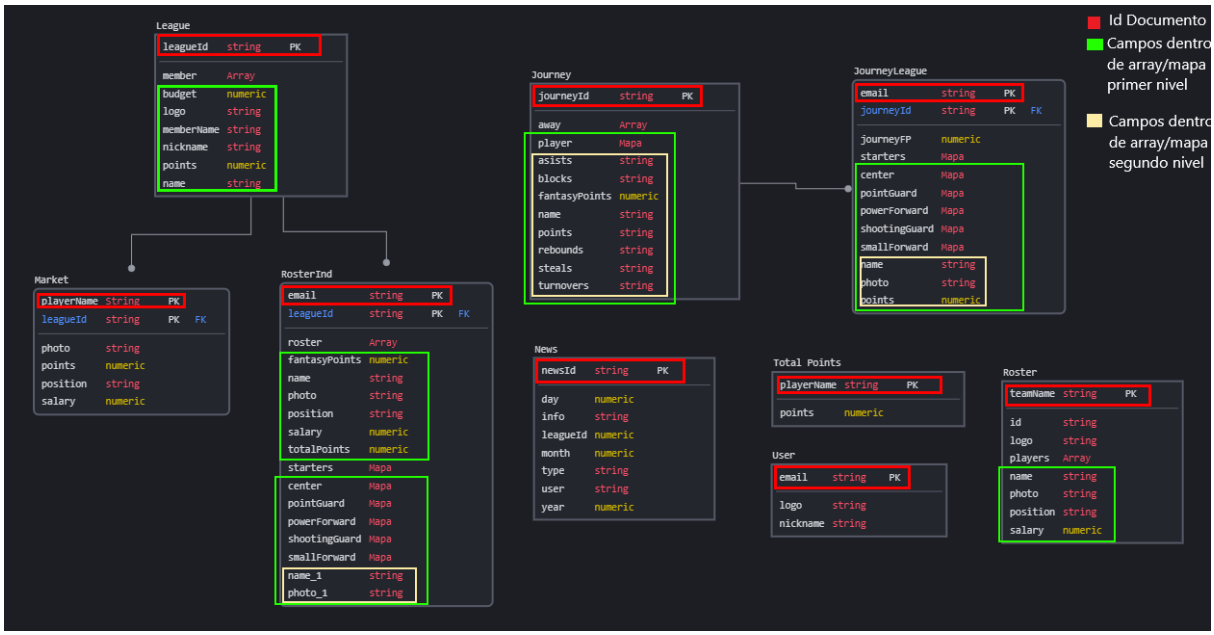


Ilustración 50: Diagrama de base de datos

6.6.1.7 Modelo de despliegue

Para finalizar la fase de diseño del sistema software conceptualiza el modelo de despliegue, el cual permite observar la arquitectura de ejecución del sistema incluyendo nodos hardware y software, además del middleware que conecta a estos nodos.

Los nodos que se incluyen en el diagrama son los siguientes:

- **Ordenador/Dispositivo móvil:** Será utilizado por los usuarios, es decir, es el nodo cliente. Pueden existir varios de estos nodos en función del número de usuarios que estén utilizando la aplicación. El entorno de ejecución donde se desplegará el artefacto aplicación será el navegador web. Mantiene la comunicación con el nodo *Servidor* explicado a continuación.
- **Servidor:** Da soporte de datos y funcionalidad al nodo anterior. Solamente existe un nodo de este tipo, el cual se ejecutará sobre node.js. Además de la comunicación con el nodo *Ordenador/Dispositivo móvil*, debe tener a su vez comunicación con el servidor de *Firebase*, representado en otro nodo, para obtener ciertos servicios.
- **Servidor Firebase:** Ofrece servicios de autenticación y base de datos al servidor de la aplicación, con el cual mantiene comunicación como se explicó anteriormente.

La comunicación entre los diferentes nodos se hace de manera distinta en función de los nodos que participen en esta. Entre los nodos *Ordenador/Dispositivo móvil* y el nodo *Servidor*, la comunicación se establece mediante peticiones HTTP desde el cliente hacia el servidor, debido a que este proporciona una API REST.

Por otro lado, la comunicación entre servidor y servidor de *Firebase* se realiza mediante las interfaces que proporciona el propio *Firebase*.

El diagrama de despliegue se muestra en la *Ilustración 51*.

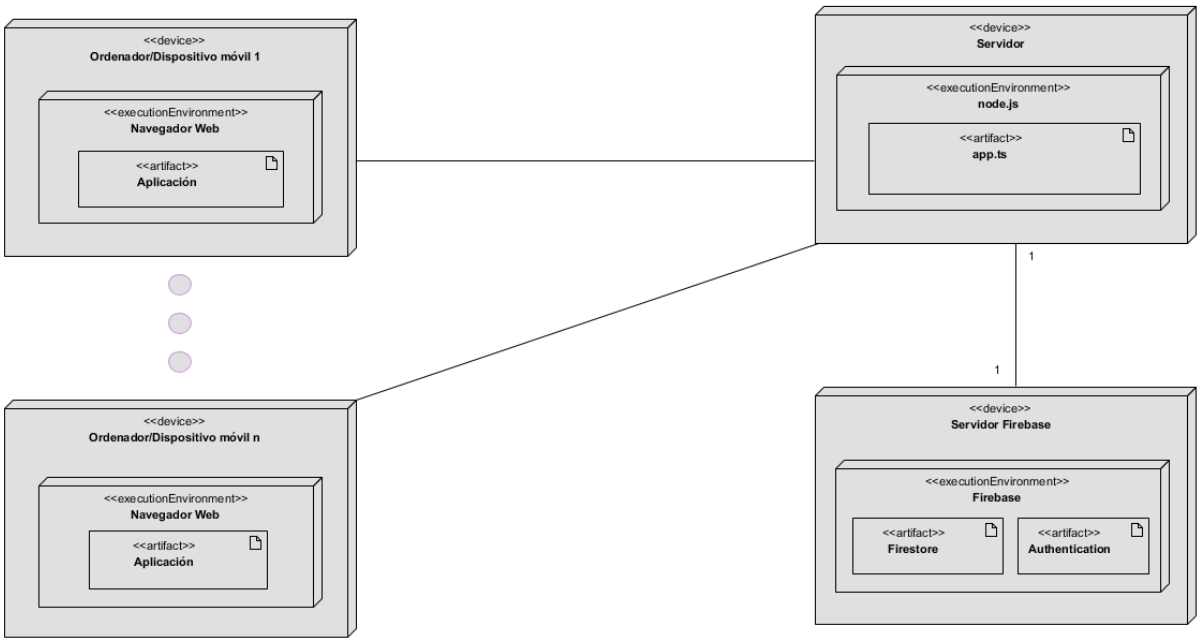


Ilustración 51: Diagrama de despliegue

6.6 Implementación

Durante la fase de implementación se ha llevado a cabo la codificación del sistema, siendo esta fase la más costosa en cuanto a carga de trabajo y tiempo a pesar la buena definición conseguida en las fases previas y el uso de herramientas y técnicas explicadas en el apartado *5 – Técnicas y Herramientas.*

La implementación del sistema ha ido variando entre la programación del servidor y aplicación web. Es decir, para lograr una funcionalidad completa primero se realizaba el servicio o los servicios en el servidor y una vez que funcionará correctamente, se pasaba a implementar la vista que necesitaba de ese servicio o servicios. Para poder detallar el desarrollo de las partes de una manera ordenada se explicarán por separado servidor y aplicación web.

Si se quieren conocer los detalles del código de la parte del servidor se ha generado una documentación técnica en formato HTML, mientras que la parte de aplicación web ha sido comentada sobre el código. Así mismo, para más información técnica sobre la implementación, estructura en directorios del sistema y módulos utilizados se puede visitar el *Anexo V – Documentación Técnica*.

Servidor

Se ha ido realizando la implementación de los diferentes servicios a ofrecer dentro de cada controlador. De manera paralela a la realización de estos servicios se iban haciendo los métodos de las clases del modelo que el servicio en cuestión necesitaba. De esta manera cuando se lograba un correcto funcionamiento del servicio se tenían métodos del controlador y del modelo desarrollados lo que ha propiciado un gran avance.

Por otra parte, cuando el desarrollo de un paquete lo ha requerido se han implementado ciertos scripts para definir tareas periódicas.

Aplicación Web

Para llevar a cabo el desarrollo de este paquete se ha dividido en dos fases: interfaz gráfica de usuario y funcionalidad.

Como se ha indicado, primero se diseñaba y se implementaba la interfaz gráfica de usuario, utilizando datos hardcodeados para visualizar el resultado que daría al obtener ciertos datos reales. Una vez la interfaz gráfica estuviera bien definida, se procedía a hacer su funcionalidad a través de los métodos para la recogida de datos desde el servidor o el envío de estos hacía él.

Por último cuando ambos componentes funcionaran correctamente se realizaba el acoplamiento, dando así por finalizada la implementación de esa funcionalidad.

6.7 Pruebas

La disciplina de pruebas juega un papel importante en el desarrollo de un sistema software ya que permite controlar la implementación y determinar cuándo una funcionalidad está finalizada. Por ello la carga de trabajo se distribuye a lo largo del ciclo de vida del desarrollo aumentando evidentemente a medida que se va implementando el código.

Se han realizado diversos tipos de pruebas en este sistema a lo largo de su desarrollo. Una de ellas han sido las pruebas unitarias por servicio, donde una vez creada la ruta para acceder a él, a través de la herramienta *Postman* se probaba la conexión. Una vez comprobada la conexión y realizada la funcionalidad involucrando los métodos del modelo, se probaban de la misma manera pero esta vez enfocadas al correcto funcionamiento completo.

Para probar la parte del cliente, de igual manera se realizaban pruebas unitarias por componente, donde inicialmente iban visualizando y probando que la interacción con el sistema era la deseada. Posteriormente cuando se producía dicha interacción se comprobaba que el componente respondía de la manera esperada.

Por último con ambas partes funcionando como se había planificado, se procedía a las pruebas que involucraban la funcionalidad conjunta.

A medida del avance en el sistema, las pruebas cada vez involucraban más funcionalidades. De este modo se comprobaba que el acoplamiento entre estas no era defectuoso.

Cuando el sistema tomó su forma final, se realizó una simulación de funcionamiento real, fingiendo el uso del sistema por parte de varios usuarios y varias semanas.

7.FUNCIONALIDAD DEL SISTEMA

Este aparatado tiene como objetivo guiar a las personas ajenas al proyecto a través del funcionamiento del sistema.

Para realizar una explicación correcta y estructurada de la funcionalidad se va a dividir el sistema en tres secciones: gestión de usuario, procedimientos de acceso a una liga y procedimientos dentro de una liga.

Para una guía más detalla de todos los pasos a seguir dentro del sistema, revisar el *Anexo VI – Manual de Usuario*.

7.1 Gestión de usuario

En esta sección se van a detallar las pantallas y procedimientos relacionados con los usuarios, desde el registro hasta la modificación de sus datos.

Se comienza con la interfaz de login, *Ilustración 52*, donde se tiene un formulario para introducir email y contraseña. A su vez, se tiene la opción para recuperar la contraseña indicada, esta opción mostrará un campo para introducir el email de la cuenta y se enviará un correo para la recuperación.

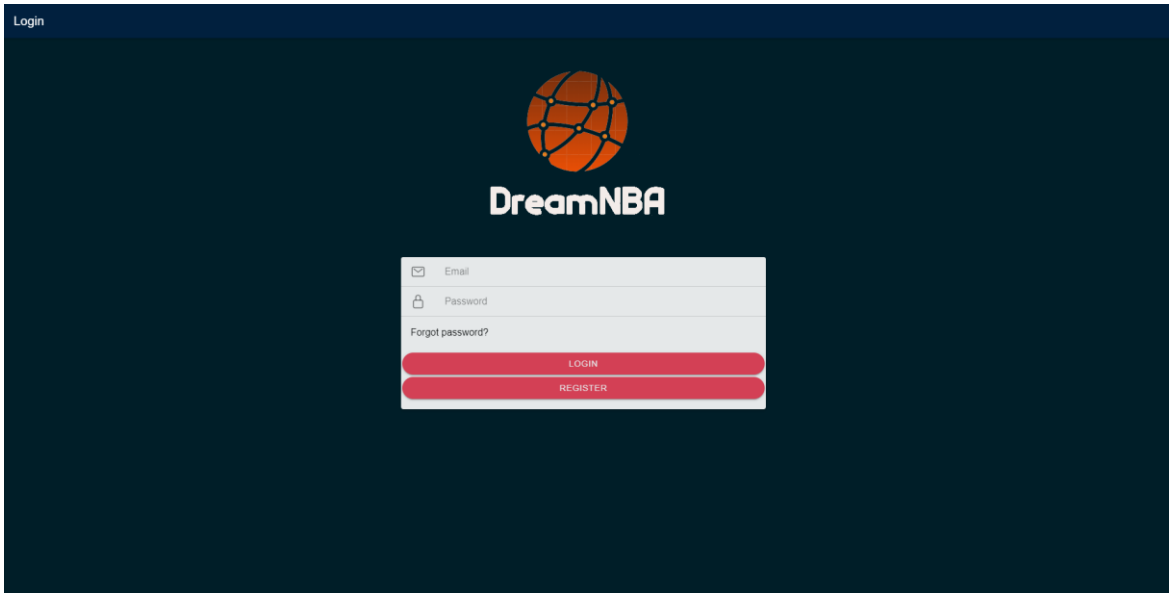


Ilustración 52: Pantalla Login

Como se aprecia en la imagen superior aparece la opción *Register*, la cual llevará al usuario a la siguiente página expuesta en la *Ilustración 53*. Esta contiene un formulario para llevar a cabo el proceso de registro. Tanto en el proceso de registro como en el de logueo se muestran mensajes de error en caso de datos incorrectos o falta de relleno de campos.

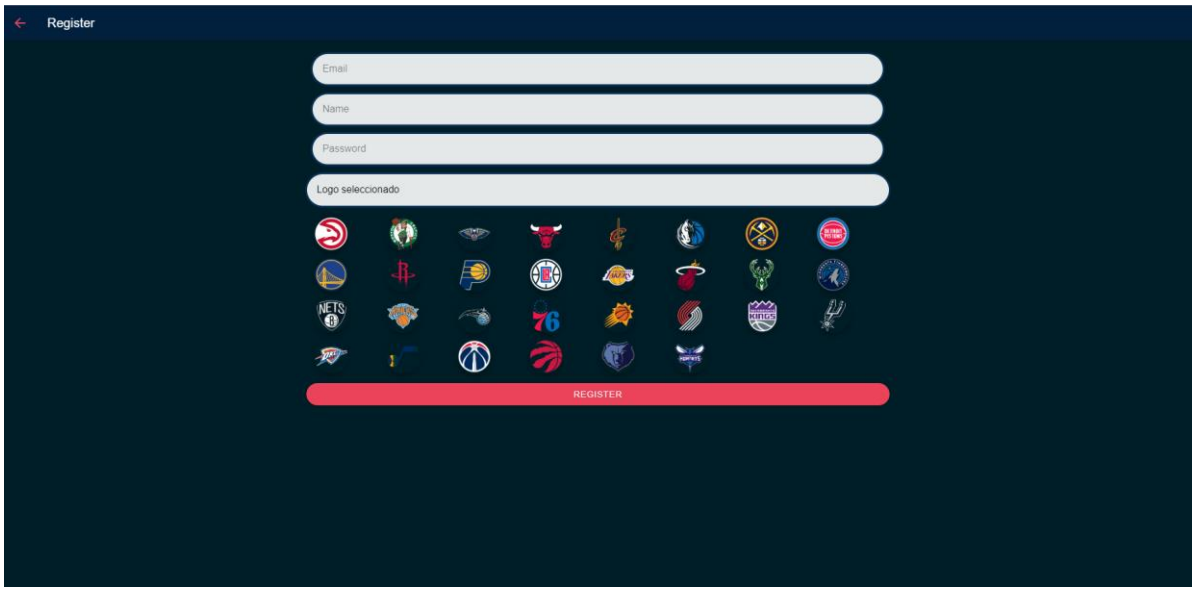


Ilustración 53: Pantalla Registro

Una vez dentro de la aplicación en una de las opciones del menú lateral se encuentra la opción de *Perfil, Ilustración 54*, donde se ofrece al usuario sus datos y las opciones para modificarlos.

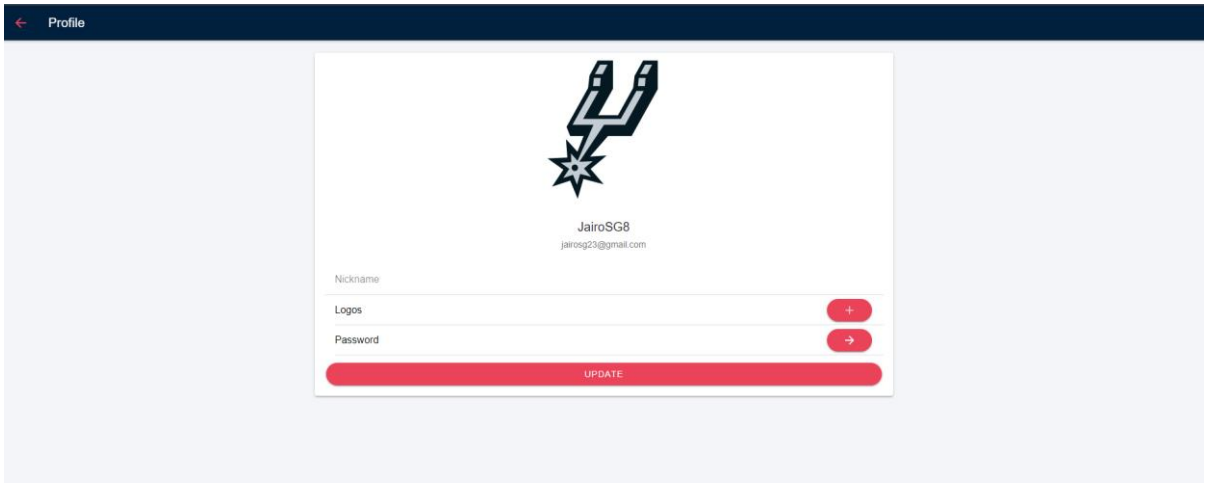


Ilustración 54: Pantalla Profile

A parte de esta opción en el menú lateral, se encuentran varias más explicadas en las secciones posteriores. Además de la acción *Logout,* este menú se observa en la *Ilustración 55*.

Ilustración 55: Menú lateral

Para finalizar con esta sección, cuando un usuario accede a la aplicación se le redirige hacia la página principal donde se muestran las noticias relacionadas con las ligas a las que este pertenece.

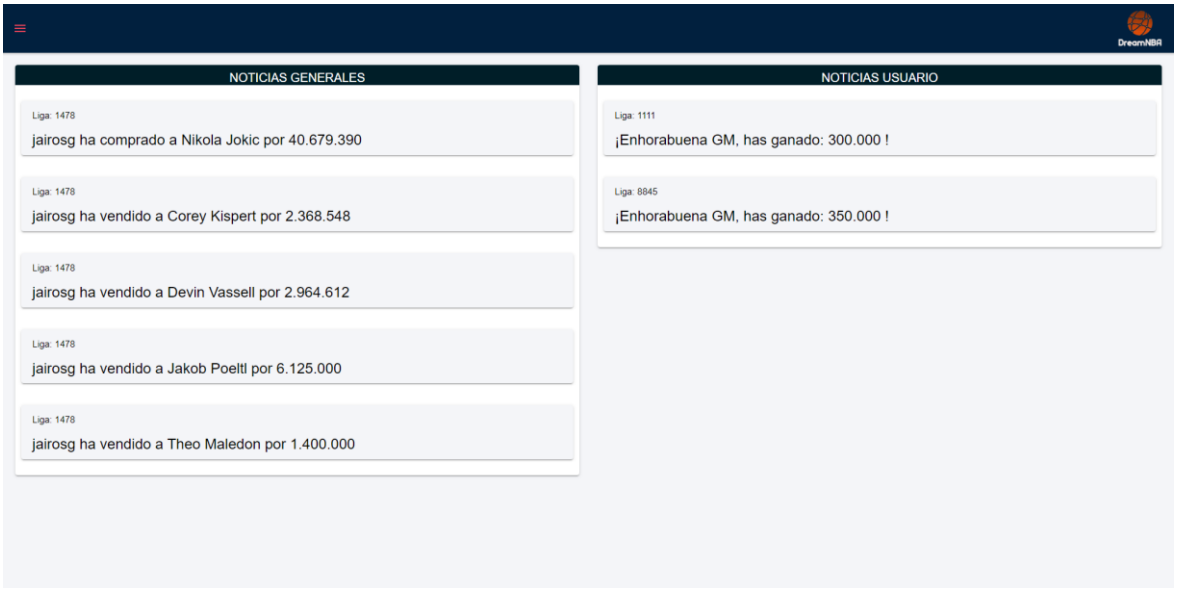


Ilustración 56: Pantalla Inicial

7.2 Procedimientos de acceso a una liga

Para acceder a una liga se tienen varias opciones. Por un lado se encuentra cuando un usuario quiere acceder a una liga que no existe o no es miembro, para esta restricción tiene las pantallas *Crear Liga* y *Añadir Liga*, a ambas se accede a través del menú lateral expuesto en la Ilustración 55. En estas ventanas se tiene un formulario donde introducir el nombre de la liga y el código en el caso de crearla o el código solamente en caso de añadirla. La Ilustración 57 e Ilustración 58 hacen referencia a estas respectivamente.

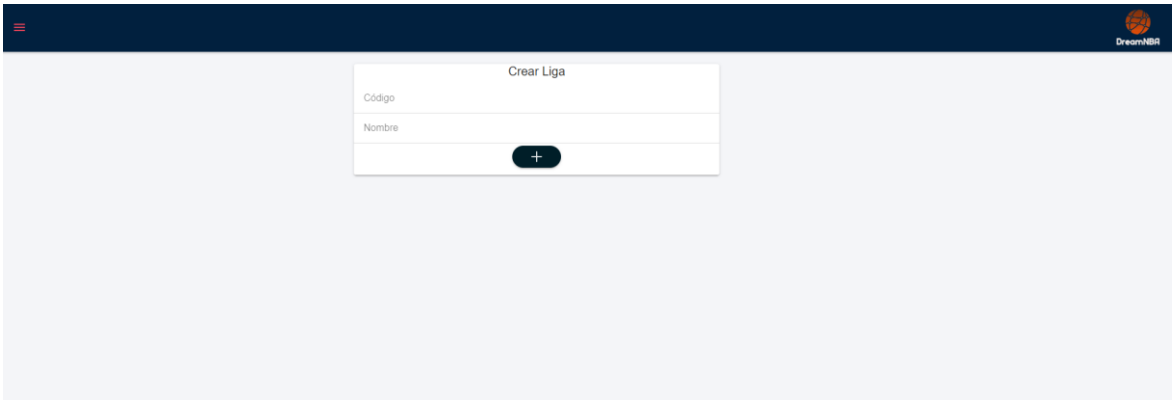


Ilustración 57: Pantalla Crear Liga

Ilustración 58: Pantalla Añadir Liga

Por otro lado, se tiene cuando un usuario ya pertenece a la liga que quiere acceder. Para ello a través del *Menú Lateral* está la opción *Ligas* que lleva a una página donde se muestra la lista de ligas de las que el usuario es miembro permitiendo acceder a estas pulsando sobre el nombre o en su caso salir de ellas a través del botón en rojo que aparece, lo tendrá que confirmar en una ventana emergente.

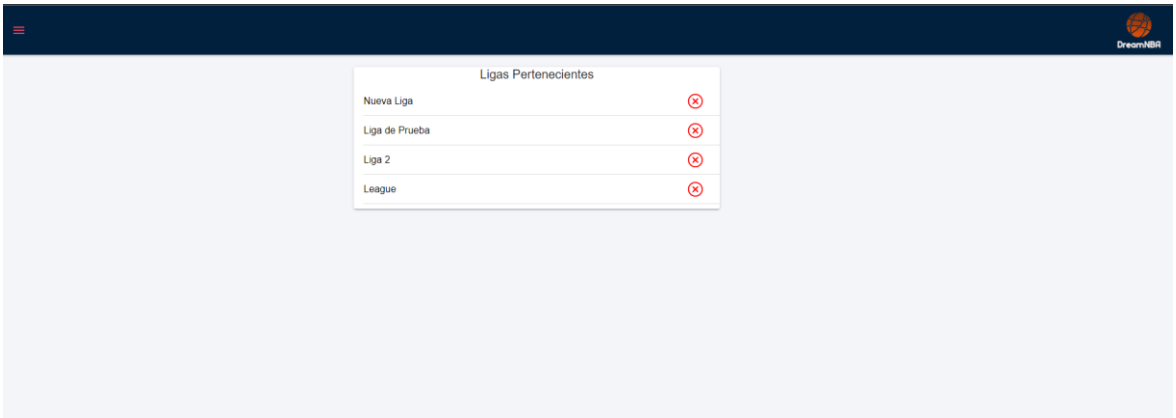


Ilustración 59: Pantalla Ligas

7.3 Procedimientos dentro de una liga

En las pantallas que se van a definir a continuación se encuentra el grueso de la aplicación.

La primera ventana que se observa al entrar a una liga es la página de *Clasificación*, *Ilustración 60*, la cual muestra el ranking general y algunos atributos relacionados con el usuario que ocupa una posición. Para visualizar la clasificación de una jornada en particular, en la esquina superior derecha se tiene un desplegable que mostrará el número de jornada que se quiere observar. Por último, si se pulsa sobre la carta de un jugador en el ranking, redireccionará al usuario hacia una página donde puede ver el roster de ese jugador, *Ilustración 61*.

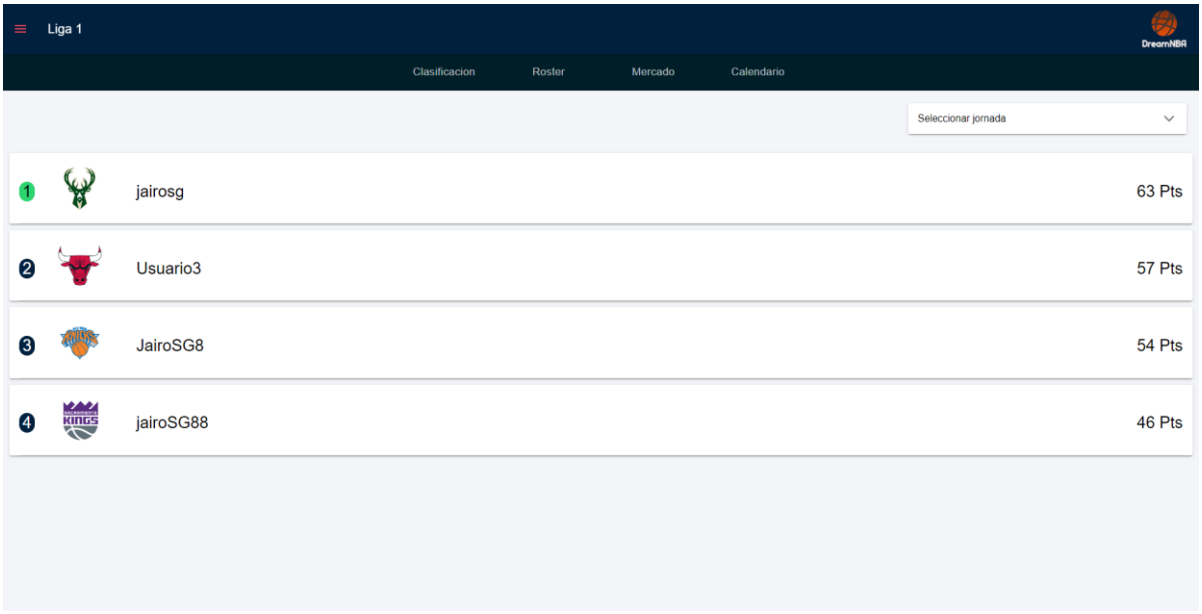


Ilustración 60: Pantalla Clasificación

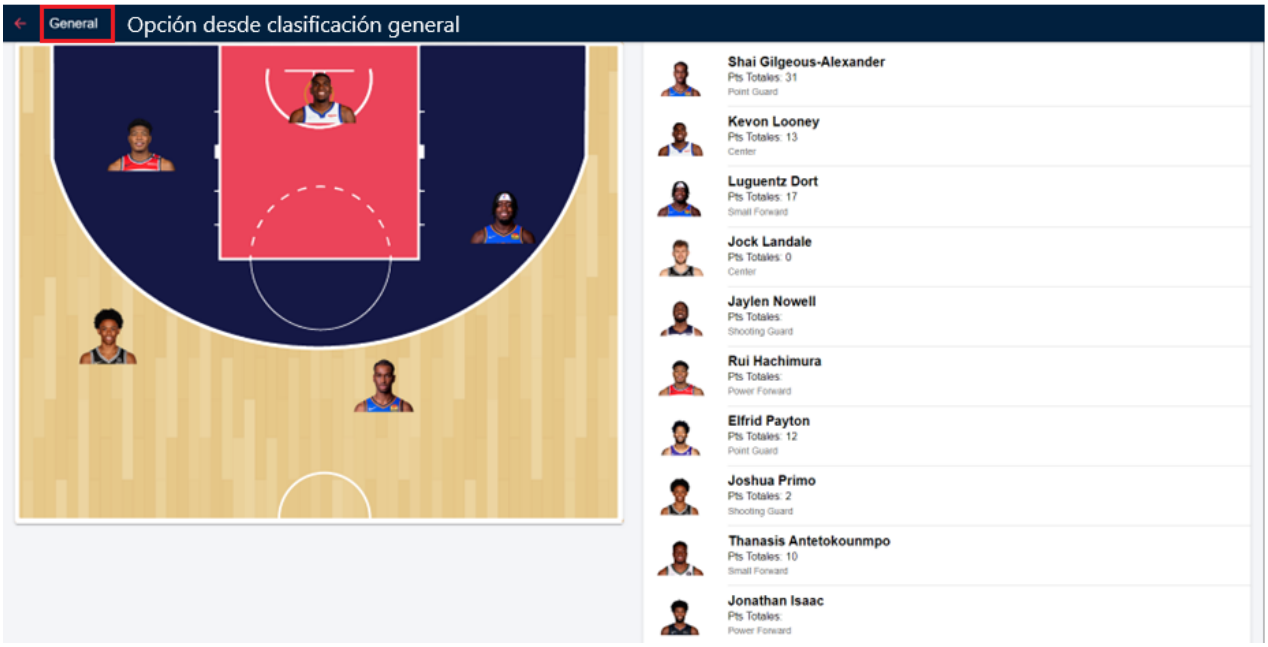


Ilustración 61: Pantalla Usuario/Amigo

Otra de las pantallas que se encuentran dentro de una liga es la página *Roster*, *Ilustración 62*, donde se dispone de dos partes diferenciadas: el quinteto y los jugadores. Empezando por el quinteto, este se despliega sobre una imagen de una mitad de cancha de baloncesto, en donde los jugadores seleccionados se posicionan sobre ella según la posición que ocupen. Siguiendo con la segunda parte de esta pantalla, los jugadores, se dispone de una lista con los jugadores que el usuario tiene, sobre cada uno se pueden realizar tres acciones, poner al quinteto, quitar del quinteto y ponerlo a la venta, representadas por tres botones..

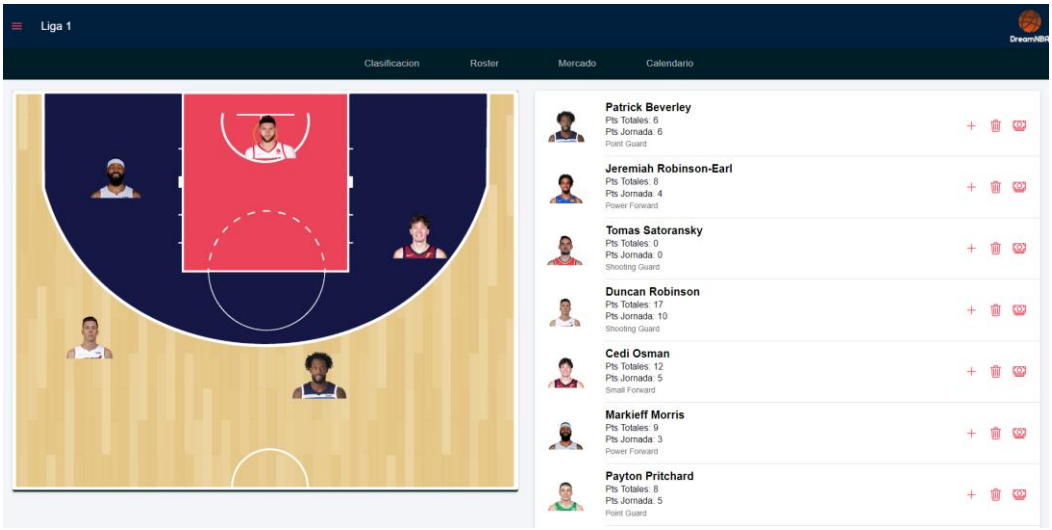


Ilustración 62: Pantalla Roster

Además de las dos anteriores mencionadas, se tiene la pantalla *Mercado.* En ella se muestra la lista de jugadores que dan forma al mercado. Sobre cada uno existen varios botones con diferentes acciones en función de algunas variables como por ejemplo, si sobre el jugador se ha hecho alguna puja o no, o por el contrario, si el jugador pertenece al usuario que ve el mercado. A parte de lo relacionado con los jugadores, se muestra el presupuesto total y el gastado en pujas.

Esta pantalla se expone en la *Ilustración 63*, así como en la *Ilustración 64* e *Ilustración 65* se muestra distintos tipos de opciones sobre un jugador.

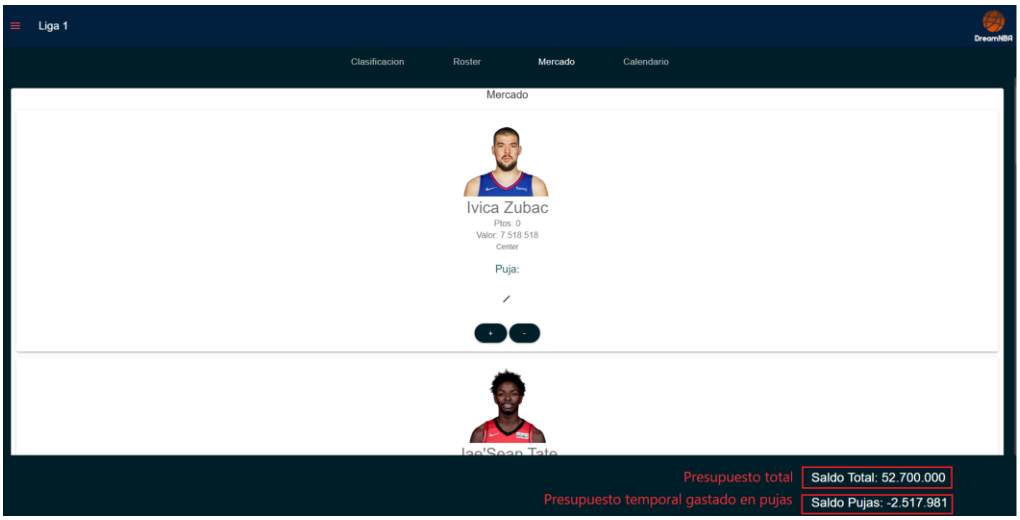


Ilustración 63: Pantalla Mercado

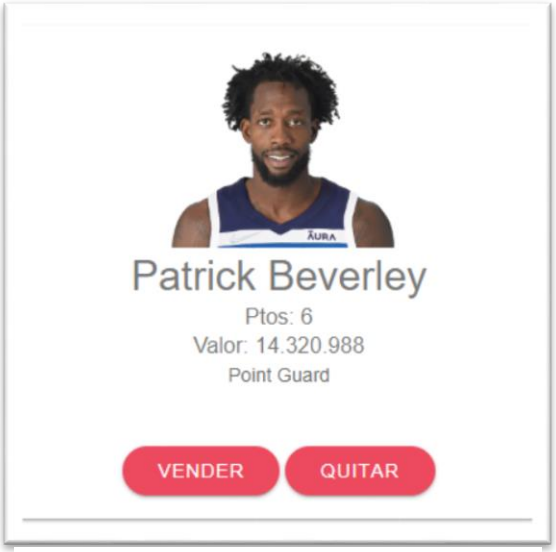


Ilustración 64: Pantalla Mercado, opción jugador propio

Ilustración 65: Pantalla Mercado, opción jugador con puja

En resumen las acciones que se puede llevar a cabo en la página *Mercado* son las siguientes:

- Pujar incrementando o decrementando en un valor de 100.000.
- Pujar introduciendo el valor a mano.
- Modificar una puja realizada a través de las opciones anteriores.
- Eliminar una puja.
- Si el jugador es propio venderlo o quitarlo del mercado.

Por último, se tiene la página *Calendario*, la cual inicialmente, cuando se accede a esta, aparecen los partidos relativos a la fecha en la que se visita. En adición , a través de un botón situado en la esquina superior derecha, se despliega un calendario para elegir la fecha de la que se desea visualizar los partidos, *Ilustración 66.*

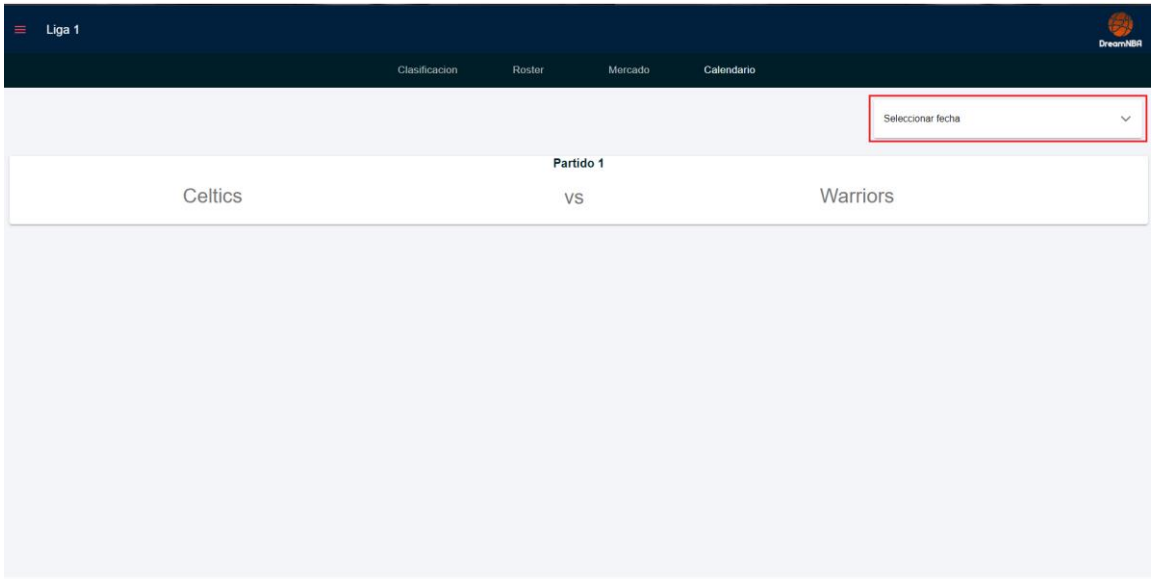


Ilustración 66: Pantalla Calendario

Dependiendo de la fecha, junto con los equipos se muestra el resultado de dicho partido, siendo la fecha posterior a la actual. Desde esta página, pulsando sobre un partido, se accede a la página de estadísticas. Esta pantalla contiene las estadísticas individuales de los jugadores que participan en un partido, *Ilustración 67*.

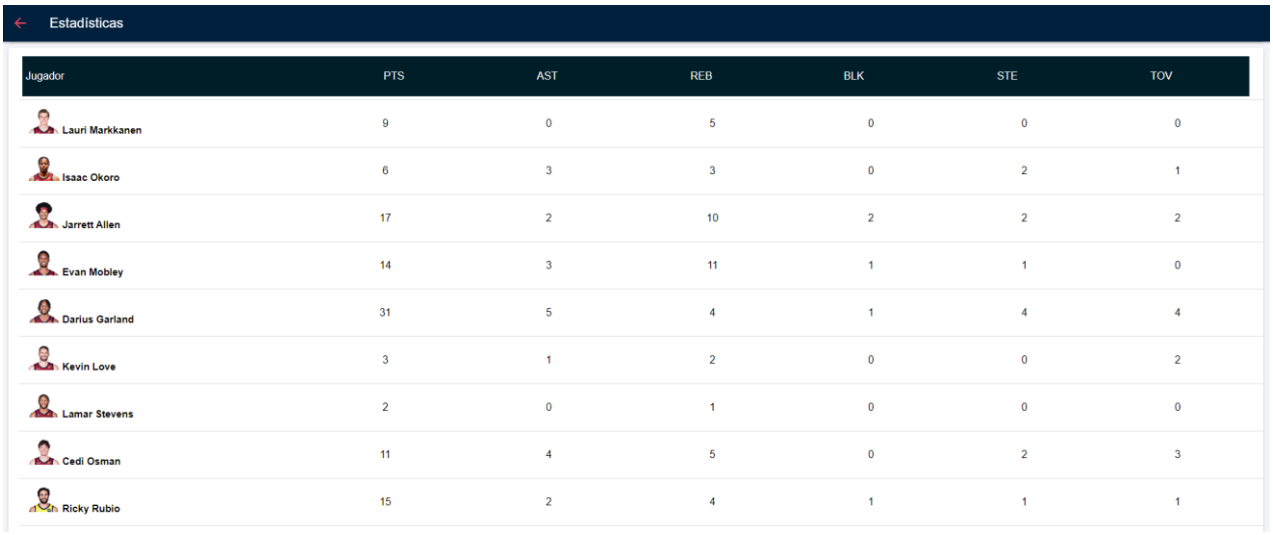


Ilustración 67: Pantalla Estadísticas

8.CONCLUSIONES Y LÍNEAS DE TRABAJO FUTURAS

En este apartado se van a recoger las conclusiones sacadas tras la finalización del proyecto Trabajo de Fin de Grado que ha implicado la realización de la aplicación web *DreamNBA*. Así mismo, se disertarán las posibles líneas de trabajo futuras que aplicar sobre este trabajo.

8.1 Conclusiones

Con vista al resultado final del proyecto, se puede afirmar que se ha logrado satisfacer los objetivos tanto personales como del sistema impuestos en las fases iniciales de desarrollo.

Haciendo un resumen hacia los requisitos del sistema, se puede decir que se ha conseguido generar un prototipo funcional que permita al usuario que esté utilizando el sistema realizar las tareas de gestión de sus datos, así como proceder a realizar las acciones propias de la aplicación como la creación de ligas, puja y compra de jugadores en el mercado, participación en las jornadas mediante la construcción de quintetos, entre otras.

Aparte de la funcionalidad lograda, se ha alcanzado la realización de una interfaz de usuario sencilla e intuitiva que permita a los usuarios la rápida navegación a través del sistema sin necesidad de un entrenamiento específico. Esta premisa puede posibilitar llegar a más gente no sólo aficionados de la NBA, aumentando el alcance de la aplicación.

Pasando a los objetivos personales, se puede decir que el desarrollo completo de un sistema software es un trabajo arduo y costoso en cuanto a tiempo y carga de trabajo. A su vez, debido al marco de trabajo elegido se ha podido comprender el proceso de cada fase en la práctica diferenciando así lo adquirido en la carrera que en su mayoría era teórico.

Por otro lado, gracias a la realización de este proyecto se ha conseguido un aprendizaje minucioso sobre ciertos lenguajes de programación como TypeScript o sobre varios marcos de trabajo de los que se ha hecho uso. Además de comprender el uso y comunicación más en profundidad entre servidor y cliente mediante el uso de *API REST* y la conexión a aplicaciones que proporcionan interfaces para su uso como *Firebase*.

Por último, he de decir que se ha conseguido una experiencia en el desarrollo de proyectos que previamente se carecía de ella, no tanto en la codificación sino en las fases previas a ella las cuales son de suma importancia.

8.2 Líneas de Trabajo Futuras

Uno de los aspectos principales a mejorar en la aplicación realizada es el tiempo necesario de recogida de ciertos datos. Algunas tareas necesitan de una cantidad notable de datos almacenados en la base de datos y que se adquieren a través de consultas compuestas, lo que conlleva a un tiempo de espera indeseado. Una de las posibles soluciones para este problema sería la migración de la base de datos hacia otra que permita una búsqueda más eficiente o una imposición de índices mayor que disminuya el tiempo de carga.

Además de esta posible mejora, se podría hacer uso de una base datos en tiempo real para ciertas tareas que permitiría dotar a la aplicación de un mayor dinamismo y que los usuarios puedan interaccionar entre sí sin notar los cambios producidos.

Aparte de las mejoras, se podría introducir nuevas funcionalidades que serían interesantes para ampliar la usabilidad de la aplicación. Una de estas opciones podría ser la implementación de un chat interno en el que los usuarios puedan intercambiar ideas o negociar una posible venta de un jugador. Así mismo, la inclusión de más acciones a realizar sobre los jugadores como una posible clausula para quitárselo a otro usuario o la implantación de gráficas para visualizar la evolución de su rendimiento.

Por otro lado, debido a la capacidad de *Ionic* para realizar aplicaciones híbridas con una sola base de código, se podría hacer uso del plugin *Capacitor* y construir la APK de la aplicación para móviles o tablets, dejando de ser solamente una aplicación web.

9. BIBLIOGRAFÍA

¿Qué es un requisito no funcional? Tipos y ejemplos. (2021). Obtenido de https://ebooksonline.es/que-es-un-requisito-no-funcional-tipos-y-ejemplos/

Berzal, F. (s.f.). *Relaciones entre clases: Diagramas de clases UML.* Obtenido de https://elvex.ugr.es/decsai/java/pdf/3C-Relaciones.pdf

Blancarte Iturralde, O. J. (2020). *Introducción a la arquitectura de software: Un enfoque práctico.*

Delgado, D. (3 de 12 de 2014). *Arquitectura de software*. Obtenido de http://arquitecturasomos4.blogspot.com/

Documentación Microsoft Project. (18 de 03 de 2022). Obtenido de https://docs.microsoft.com/es-es/projectonline/project-online

Durán Toro, A., & Bernárdez Jiménez, B. (2000). *Metodología para la Elicitación de Requisitos de Sistemas Software, versión 2.1.* Sevilla.

EcuRed. (s.f.). Obtenido de https://www.ecured.cu/Herramienta_CASE

Fundación Openjs. (s.f.). *Express*. Obtenido de https://expressjs.com/es/

García Peñalvo, F. J., & García Holgado, A. (s.f.). *Fundamentos de la vista de casos de uso.*

García Peñalvo, F. J., & García Holgado, A. (s.f.). *Modelo de dominio.* Obtenido de https://repositorio.grial.eu/bitstream/grial/1153/1/8.%20Modelo%20de%20dominio.pdf

García Peñalvo, F. J., & Moreno García, M. N. (s.f.). *Trasparencias de Ingeniería del Software, Tema 5.*

Gilani, S. (s.f.). *npm.io sportdataverse*. Obtenido de https://npm.io/package/sportsdataverse

Google. (s.f.). *Cloud Firestore* . Obtenido de https://firebase.google.com/products/firestore?gclid=Cj0KCQjwn4qWBhCvARIsAFNAMihxNrVAkzcqdeI9zlarbcwvJcLhznDrW444LJmwJfqkASGpxpKKssIaAiphEALw_wcB&gclsrc=aw.ds

Google. (s.f.). *Firebase*. Obtenido de https://firebase.google.com/?hl=es-419

Google. (s.f.). *Firebase Authentication*. Obtenido de https://firebase.google.com/products/auth?gclid=Cj0KCQjwn4qWBhCvARIsAFNAMiioeRerDzeciBzuoShdALnervz1vIoqzPuTlvdBGFYOGLdPfI3pyBAaAk9vEALw_wcB&gclsrc=aw.ds

IBM. (05 de 03 de 2021). Obtenido de El modelo de diseño: https://www.ibm.com/docs/es/rsm/7.5.0?topic=model-design

Ionic Framework. (s.f.). Obtenido de https://ionicframework.com/docs/components

Koeze, E., & Popper, N. (07 de 04 de 2020). *The New York Times*. Obtenido de https://www.nytimes.com/interactive/2020/04/07/technology/coronavirus-internet-use.html

Microsoft. (18 de 03 de 2022). Obtenido de https://docs.microsoft.com/es-es/projectonline/project-online

Microsoft. (s.f.). *Microsoft Project*. Obtenido de https://www.microsoft.com/es-es/microsoft-365/project/project-management-software

Microsoft. (s.f.). *Visual Studio Code*. Obtenido de https://code.visualstudio.com/

Moreno García, M. N. (s.f.). *Práctica 1, Estimación del esfuerzo, Gestión de proyectos.*

Moreno García, M. N. (s.f.). *Tema 2, Medición del software, Gestión de proyectos.*

Moreno García, M. N., & J., G. P. (2020). *Trasparencias Ingeniería del Software II Tema 3, Patrones.*

Moreno García, M., & García Peñalvo, F. J. (s.f.). *Transparencias Ingeniería del software, UML Vista de interacción.*

Navarro Cáceres, M., & N., M. G. (s.f.). *Práctica 2, Planificación temporal, Gestión de proyectos.*

OpenJS. (s.f.). *Node.js*. Obtenido de https://nodejs.org/es/

Parsec Media S.L. (24 de 08 de 2020). *Geeknetic*. Obtenido de https://www.geeknetic.es/Cache-web/que-es-y-para-que-sirve

Postman. (s.f.). *Postman*. Obtenido de https://www.postman.com/

Programación Web, Evolución de las aplicaciones Web. (20 de 09 de 2015). Obtenido de https://sites.google.com/site/programacionwebhegm/unidad-1-arquitectura/1-1-evolucion-de-las-aplicaciones-web

Santamaría, R. (2022). *Apuntes Sistemas Distribuidos, Tema 3 Middleware.*

Servicio de Informática, Universidad de Alicante. (s.f.). Obtenido de Modelo vista controlador: https://si.ua.es/es/documentacion/asp-net-mvc-3/1-dia/modelo-vista-controlador-mvc.html#:~:text=Modelo%20Vista%20Controlador%20(MVC)%20es,control%20en%20tres%20componentes%20distintos.

Typedoc. (s.f.). Obtenido de http://typedoc.org/

TypeScript. (s.f.). Obtenido de https://www.typescriptlang.org/

Visual Paradigm. (s.f.). Obtenido de https://www.visual-paradigm.com/

Visual Paradigm. (s.f.). *Visual Paradigm*. Obtenido de https://www.visual-paradigm.com/

Vue Styleguidist. (s.f.). Obtenido de https://vue-styleguidist.github.io/docs/Documenting.html#code-comments

Vue.js. (s.f.). Obtenido de https://vuejs.org/

Wikipedia. (12 de 04 de 2022). Obtenido de https://es.wikipedia.org/wiki/Firebase

Wikipedia. (08 de 06 de 2022). Obtenido de https://en.wikipedia.org/wiki/Software_design_pattern

ANEXO I: Plan del proyecto

Dream NBA: aplicación para la creación de ligas virtuales y gestión de un equipo propio

Trabajo de Fin de Grado

INGENIERÍA INFORMÁTICA

VNiVERSiDAD D SALAMANCA

CAMPUS DE EXCELENCIA INTERNACIONAL

Julio 2022

Autor

Jairo Sánchez García

Tutores

Gabriel Villarrubia González

Juan Francisco de Paz Santana

Héctor Sánchez San Blas

TABLA DE CONTENIDO

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

El propósito principal de este anexo es el de realizar una estimación de los costes y el tiempo que conllevará el desarrollo de este proyecto. A partir de la estimación se hará la planificación temporal.

La primera tarea que se lleva a cabo es la estimación del esfuerzo, donde se utilizará la métrica UCP(Use Case Points), explicada posteriormente, y la herramienta EZEstimate. Dará como resultado una primera aproximación en horas de lo que el proyecto puede durar.

La segunda tarea por hacer es la planificación temporal siguiendo el marco de trabajo del Proceso Unificado, *Ilustración 1*. Con la ayuda de este marco de trabajo y la herramienta Microsoft Project se definirá el tiempo que debe durar la realización de cada tarea, con lo que se obtendrá la duración total del proyecto.

Teniendo en cuenta estas dos tareas, el documento contará respectivamente con una sección para cada tarea.

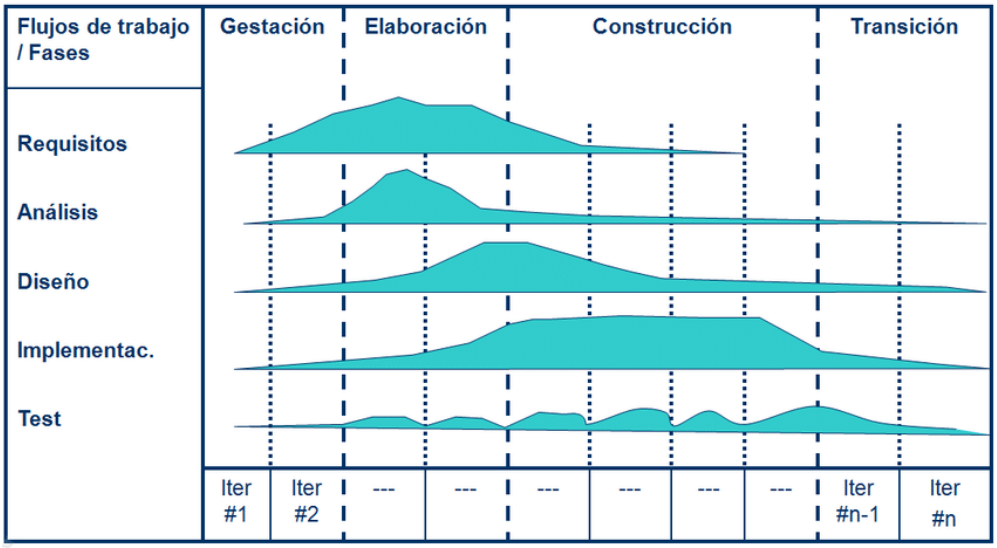


Ilustración 1: Fases y disciplinas del Proceso Unificado

2. ESTIMACIÓN DEL ESFUERZO

En este apartado se va a realizar una estimación del esfuerzo de desarrollo a través de la métrica UCP (Use Case Points), considerando actores, casos de uso y factores técnicos y de entorno mediante el cálculo de las variables UUCP, TCF y ECF. Para llevar a cabo el cálculo de estas variables y a su vez del esfuerzo de desarrollo se va a utilizar la herramienta EZEstimate.

2.1 Interacción de los actores

Los actores que van a interaccionar con el sistema son tres, "*Usuario No Logueado*", "*Usuario Logueado*" y "*Timer*". Para determinar su complejidad hay que tener en cuenta la siguiente clasificación:

- **Simple:** el actor es un sistema y la aplicación se comunica con él mediante una API.
- **Medio:** el actor es un sistema y la aplicación se comunica con él mediante un protocolo.
- **Complejo:** el actor es una persona y se comunica con la aplicación a través de una interfaz gráfica de usuario.

Por lo tanto los actores de este sistema tienen la siguiente complejidad:

ACT-0001	**Usuario No Logueado**
Complejidad	Complejo
Descripción	El actor es una persona y su interacción con el sistema es a través de una interfaz gráfica

Tabla 1: Complejidad de ACT-0001 Usuario No Logueado

ACT-0002	**Usuario Logueado**
Complejidad	Complejo
Descripción	El actor es una persona y su interacción con el sistema es a través de una interfaz gráfica

Tabla 2: Complejidad de ACT-0002 Usuario Logueado

ACT-0003	**Timer**
Complejidad	Simple
Descripción	La comunicación con el actor se realiza mediante una API

Tabla 3: Complejidad de ACT-0003 Timer

2.2 Complejidad de los casos de uso

Para deducir la complejidad que tiene un caso de uso hay que identificar el número de transacciones que tendrá este. Se cuenta como transacción la acción desde que el actor interactúa con el sistema hasta que recibe una respuesta de él, en el caso de producirse una excepción en algún caso de uso se considerará como transacción si se modifica el flujo principal o requiere de intervención por parte del actor. Para determinar la complejidad hay que tener en cuenta la siguiente clasificación:

- **Simple:** 3 o menos transacciones.
- **Medio:** entre 4 o 7 transacciones.
- **Complejo:** a partir de 7 transacciones.

Por lo tanto el número de transacciones y complejidad de los casos de uso es la siguiente:

Caso de uso	Transacciones	Complejidad
UC-0001	2	Simple
UC-0002	1	Simple
UC-0003	1	Simple
UC-0004	2	Simple
UC-0005	1	Simple
UC-0006	2	Simple
UC-0007	1	Simple
UC-0008	2	Simple
UC-0009	2	Simple
UC-0010	1	Simple
UC-0011	1	Simple
UC-0012	1	Simple
UC-0013	2	Simple
UC-0014	3	Simple
UC-0015	1	Simple
UC-0016	1	Simple
UC-0017	2	Simple
UC-0018	2	Simple
UC-0019	1	Simple
UC-0020	4	Medio
UC-0021	1	Simple
UC-0022	2	Simple
UC-0023	4	Medio
UC-0024	1	Simple
UC-0025	3	Simple
UC-0026	1	Simple
UC-0027	1	Simple
UC-0028	1	Simple
UC-0029	2	Simple
UC-0030	3	Simple

Tabla 4: Complejidad casos de uso

2.3 Factores implicados en la estimación

2.3.1 Factores de complejidad técnica

- **T1 Sistemas distribuidos**: la base de datos se encuentra separada del servidor de la aplicación. Además se va a utilizar una API externa. Valor asignado: **1.**
- **T2 Rendimiento:** se necesita una cierta rapidez en la respuesta de la aplicación y realización de acciones para no mantener a los usuarios demasiado tiempo en espera. Valor asignado: **2**.
- **T3 Eficiencia del usuario final:** se requiere que el sistema sea ágil para que el usuario pueda realizar un mayor número de acciones en el menor tiempo posible. A parte de reducir la complejidad para los usuarios menos expertos. Valor asignado: **2**.
- **T4 Procesamiento interno complejo:** el sistema no deberá llevar a cabo muchas operaciones complejas, solamente el cálculo de puntos debido a diversos factores por jornada. Valor asignado: **1**.

- **T5 Reusabilidad:** debido a la utilización de Node.js para la parte del servidor se podría reutilizar para el tratamiento de datos y de API REST por otro frontend con características similares. La parte del cliente no sería muy reutilizable a excepción del módulo de usuarios el cual podría ser utilizado en futuros desarrollos. Valor asignado: **1**.
- **T6 Facilidad de instalación:** se realiza de manera sencilla a través del despliegue del código en un servidor. Por parte de los usuarios a través de cualquier tienda de aplicaciones si se trata de la aplicación móvil o bien a través del navegador sin necesidad de instalación. Valor asignado: **1**.
- **T7 Facilidad de uso:** de manera que la aplicación llegue al mayor número de usuarios posible se requiere de un diseño sencillo e intuitivo siguiendo las doctrinas de aplicaciones del sector populares. Valor asignado: **3**.
- **T8 Portabilidad:** debido a los frameworks que se utilizan para su desarrollo se trata de un sistema portable a varias plataformas. Valor asignado: **2**.
- **T9 Facilidad de cambio:** se tiene que ser posible realizar cambios en función de posibles fallos y/o peticiones de usuarios en forma de sugerencias. Valor asignado: **2**.
- **T10 Concurrencia:** se trata de una aplicación en donde van a convivir varios usuarios a la vez realizando acciones similares. Valor asignado: **1**.
- **T11 Características especiales de seguridad:** el sistema no trata con datos sensibles. El único peligro al que se tiene que enfrentar es a la suplantación de identidad. Valor asignado: **1**.
- **T12 Acceso directo a terceras partes:** no se considera este punto. Valor asignado: **0**.
- **T13 Se requiere entrenamiento especial del usuario:** no es necesario un alto grado de conocimiento de baloncesto ni de NBA para la utilización de la aplicación. Puede ayudar tener conocimiento previo en este tipo de aplicaciones pero no sería estrictamente necesario. Valor asignado: **1**.

2.3.2 Factores del entorno

- **E1 Familiaridad con UML:** a lo largo de la carrera se ha tenido bastante contacto con un UML. Valor asignado: **3.**
- **E2 Trabajadores a tiempo parcial:** se cuenta únicamente con un desarrollador con total dedicación al proyecto. Valor asignado: **0.**
- **E3 Capacidad de los analistas:** cierta experiencia en trabajos del grado pero nunca en un proyecto de carácter real. Valor asignado: **1.**
- **E4 Experiencia en la aplicación:** se tiene considerada experiencia en el desarrollo de aplicaciones web con los lenguajes de programación y marcos de trabajo que se utilizan. Valor asignado: **3.**
- **E5 Experiencia orientada a objetos:** no se considera una amplia experiencia aunque se ha trabajado con orientación objetos en diversos trabajos durante el grado. Valor asignado: **3.**
- **E6 Motivación:** amplia motivación ya que se trata de un proyecto personal. A parte que conlleva a la finalización del grado. Valor asignado: **5**.
- **E7 Dificultad del lenguaje de programación:** los lenguajes de programación que se utilizan se permite la utilización de módulos y son de alto nivel por lo que no se considera una alta dificultad. Además, con estos lenguajes ya se ha trabajado con anterioridad por lo que rebaja en nivel de dificultad. Valor asignado: **3.**

- **E8 Estabilidad de requisitos:** se tienen unos requisitos bien definidos y especificados en donde no se deberían realizar demasiadas modificaciones, aunque hay que considerar pequeños cambios siempre. Valor asignado: **4.**

2.4 Duración estimada

Una vez introducidos todos los factores calculados en el apartado anterior en la herramienta anteriormente mencionada EZEstimate, como se observa en las ilustraciones siguientes, se obtienen los resultados, *Ilustración 5*.

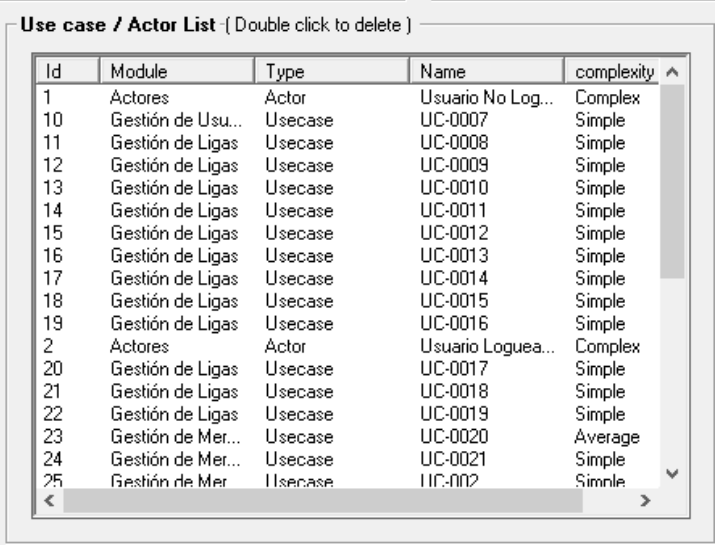


Ilustración 2: Complejidad casos de uso/actores

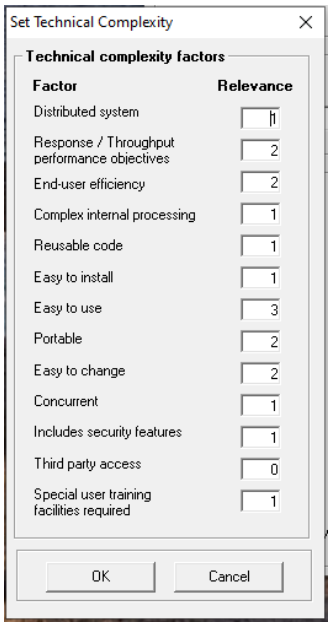


Ilustración 4: Factores de complejidad técnica

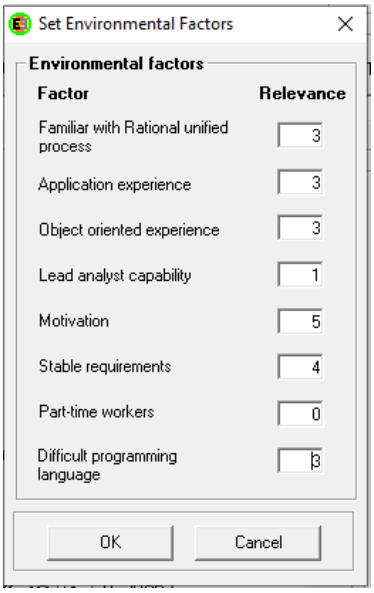


Ilustración 3: Factores del entorno

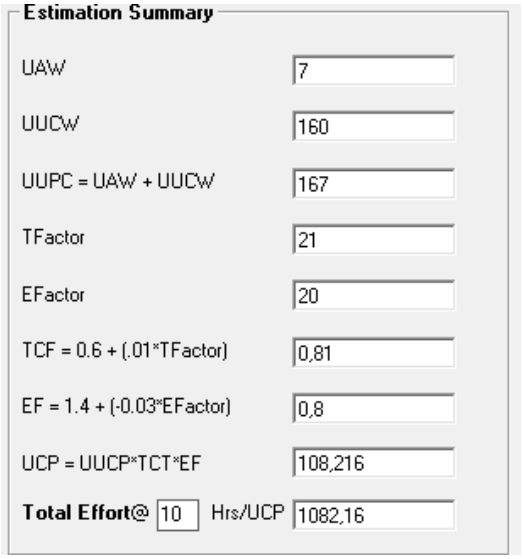


Ilustración 5: Variables y resultado final

Según la estimación realizada, los días necesarios para llevar a cabo el proyecto serían 135 días, teniendo en cuenta que participa un desarrollador a tiempo completo, es decir, 8 horas diarias, el tiempo de desarrollo en meses sería de unos 4 meses y medio. El tiempo resultante es quizás un poco elevado para un TFG pero entra dentro de los plazos establecidos para su finalización y entrega.

3. PLANIFICACIÓN TEMPORAL

A continuación, se detallarán los tiempos que se tienen que emplear para la resolución de las tareas que conforman el proyecto, dando como resultado el tiempo total en días que tomará desarrollar el proyecto completo.

Se va a adoptar el esquema de fases y disciplinas del Proceso Unificado, explicado seguidamente, para la identificación de tareas y subtareas.

- **Inicio:** en esta fase se define el alcance del proyecto y se desarrollan los casos de negocio.
- **Elaboración:** se realiza la planificación y detalle de los casos de uso, así como el diseño de la arquitectura.
- **Construcción:** se procede a hacer la construcción del proyecto.
- **Transición:** refinamiento de problemas y mejoras.

Cada una de estas fases se descomponen a su vez en una serie de disciplinas:

- **Modelado del negocio:** se centra en reuniones con los clientes o internas entre los propios participantes en el proyecto. Se definen los riesgos, costes, detalles del sistema, entre otras cosas. Se pretende modelar el proyecto implicando a todas las partes que participan en él. La mayor carga de trabajo de esta disciplina reside en la fase de inicio.
- **Requisitos:** se produce la especificación de requisitos software, incluyendo la descripción y desarrollo de los casos de uso y sus diagramas correspondientes. Tiene más peso en las fases iniciales del proyecto aunque la carga de trabajo prosigue a lo largo del ciclo de vida debido a los cambios que se pueden producir.

- **Análisis:** como su propio nombre indica se produce en esta fase se va a realizar el análisis de requisitos. Se identifican las clases de análisis y se realiza la arquitectura del modelo de análisis, aparte de definir el comportamiento dinámico de los casos de uso a través de los diagramas de secuencia y el modelado de datos con el diagrama de clases. Su carga de trabajo se concentra especialmente en la fase de elaboración, pero como pasa con la disciplina anterior se va a extender a lo largo del ciclo de vida a causa de las revisiones y refinamientos.
- **Diseño:** se produce el modelo de diseño, conformando el sistema en un nivel más bajo. Una de las tareas que se producen en esta disciplina es la realización casos de uso diseño y la elaboración del diagrama de clases de diseño. Se definen los patrones arquitectónicos y de diseño que se van a utilizar en el desarrollo del sistema. En las fases de elaboración y construcción reside la mayor carga de trabajo de esta disciplina.
- **Implementación:** se hacen las tareas relacionadas con la programación del sistema. En las primeras fases su carga de trabajo es casi indiferente debido a la poca información y modelado que se tiene del sistema, por ello aumenta de manera considerable en la fase de construcción donde implica el mayor tiempo de trabajo de esta.
- **Pruebas:** en esta disciplina se evalúan los aspectos relevantes del sistema y el correcto funcionamiento de este evitando arrastrar fallos a lo largo del ciclo de vida y permitiendo una corrección temprana de los mismos, de esta manera se remedian realizar cambios muy costosos que impliquen un retraso en la entrega. Por lo que a medida que se va teniendo un sistema más completo la carga de trabajo de pruebas aumentará, siendo relevante en la fase de transición donde se dispone de una versión casi completa del sistema.

Para llevar a cabo la planificación temporal se ha hecho uso de la herramienta Microsoft Project, en donde lo primero es definir la fecha de inicio del proyecto y el calendario sobre el que se va a trabajar, *Ilustración 6* e *Ilustración 7*.

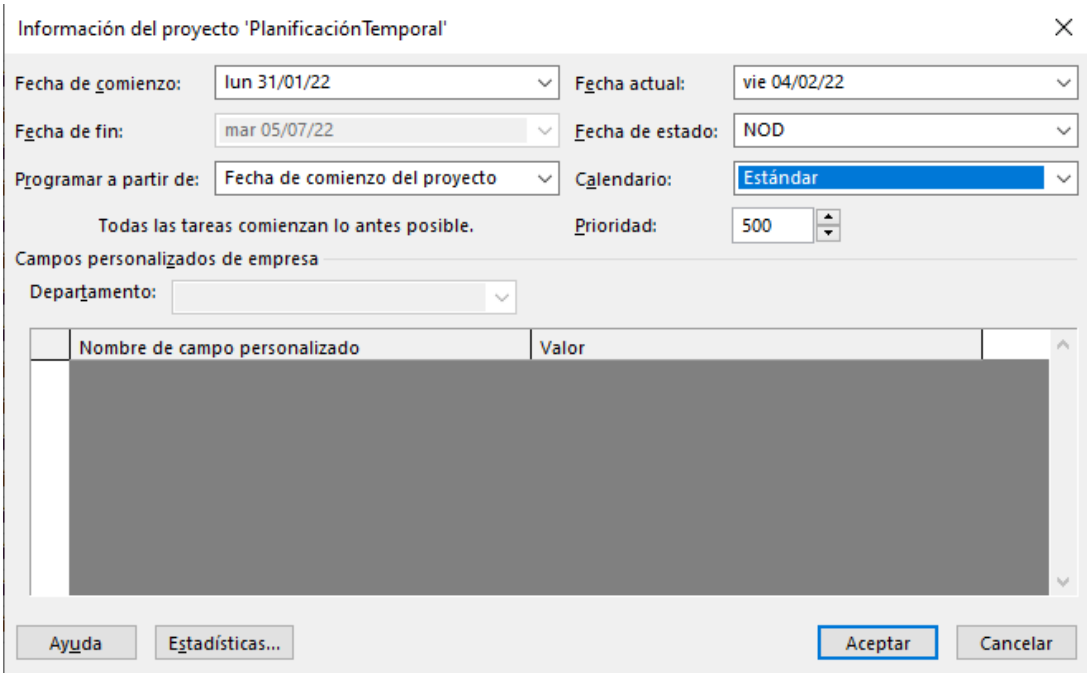


Ilustración 6: Información del proyecto

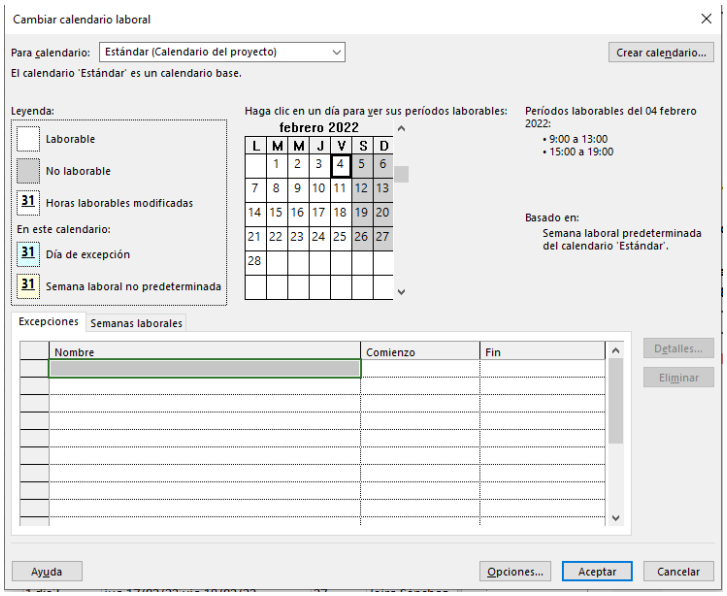


Ilustración 7: Calendario de trabajo

En las siguientes ilustraciones se puede observar la definición de tareas siguiendo el esquema explicado anteriormente.

2	**Inicio**	**14,5 días?**	**lun 31/01/22**	**vie 18/02/22**		
3	**Iteración 1**	**6,5 días?**	**lun 31/01/22**	**mar 08/02/22**		
4	**Modelado del negocio**	**1 día?**	**lun 31/01/22**	**lun 31/01/22**		
5	Objetivos generales	1 día?	lun 31/01/22	lun 31/01/22		Jairo Sánchez
6	**Requisitos**	**2,5 días**	**mar 01/02/2.**	**jue 03/02/22**		
7	Identificación de requisitos	0,5 días	mar 01/02/2.	mar 01/02/22	5	Jairo Sánchez
8	Definición de actores y escenarios	2 días	mar 01/02/2.	jue 03/02/22	7	Jairo Sánchez
9	**Análisis**	**1 día**	**jue 03/02/22**	**vie 04/02/22**		
10	Definición de las clases	1 día	jue 03/02/22	vie 04/02/22	8	Jairo Sánchez
11	**Diseño**	**1 día?**	**vie 04/02/22**	**lun 07/02/22**		
12	Definición del modelo de diseño	1 día?	vie 04/02/22	lun 07/02/22	10	Jairo Sánchez
13	**Implementación**	**1 día?**	**lun 07/02/22**	**mar 08/02/22**		
14	Definición de lengaujes de programación y marcos de trabajo	1 día?	lun 07/02/22	mar 08/02/22	12	Jairo Sánchez
15	Fin Iteración 1	0 días	mar 08/02/2.	mar 08/02/22	3	
16	**Iteración 2**	**8 días?**	**mar 08/02/2.**	**vie 18/02/22**	**15**	
17	**Modelado del negocio**	**1 día?**	**mar 08/02/2.**	**mié 09/02/22**		
18	Refinamiento de objetivos	1 día?	mar 08/02/2.	mié 09/02/22		Jairo Sánchez
19	**Requisitos**	**3 días?**	**mié 09/02/2.**	**lun 14/02/22**		
20	Refinamiento final de requisitos	1 día?	mié 09/02/2.	jue 10/02/22	18	Jairo Sánchez
21	Refinamiento final de actores y escenarios	1 día?	jue 10/02/22	vie 11/02/22	20	Jairo Sánchez
22	Diagrama de casos de uso	1 día?	vie 11/02/22	lun 14/02/22	21	Jairo Sánchez
23	**Análisis**	**2 días?**	**lun 14/02/22**	**mié 16/02/22**		
24	Definición de paquetes	0,5 días	lun 14/02/22	lun 14/02/22	22	Jairo Sánchez
25	Diagrama de clases	1,5 días?	mar 15/02/2.	mié 16/02/22	24	Jairo Sánchez
26	**Diseño**	**1 día?**	**mié 16/02/2.**	**jue 17/02/22**		
27	Refinamiento clases de diseño	1 día?	mié 16/02/2.	jue 17/02/22	25	Jairo Sánchez
28	**Implementación**	**1 día?**	**jue 17/02/22**	**vie 18/02/22**		
29	Creación del proyecto en el entorno de desarrollo	1 día?	jue 17/02/22	vie 18/02/22	27	Jairo Sánchez
30	Fin Iteración 2	0 días	vie 18/02/22	vie 18/02/22	16	
31	Fin Inicio	0 días	vie 18/02/22	vie 18/02/22	2	

Ilustración 8: Tareas fase Inicio

32	**Elaboración**	**31,5 días?**	**vie 18/02/22**	**lun 04/04/22**	**31**	
33	**Iteración 1**	**8 días?**	**vie 18/02/22**	**mié 02/03/22**		
34	**Modelado del negocio**	**1 día?**	**vie 18/02/22**	**lun 21/02/22**		
35	Investigación de recursos	1 día?	vie 18/02/22	lun 21/02/22		Jairo Sánchez
36	**Requisitos**	**2 días**	**lun 21/02/22**	**mié 23/02/22**		
37	Descripción de casos de uso del paquete "Gestión de Usuarios"	2 días	lun 21/02/22	mié 23/02/22	35	Jairo Sánchez
38	**Análisis**	**2 días**	**mié 23/02/2**	**vie 25/02/22**		
39	Elaboración de diagramas de secuencia del paquete "Gestión de Usuarios"	2 días	mié 23/02/2	vie 25/02/22	37	Jairo Sánchez
40	**Diseño**	**2 días**	**vie 25/02/22**	**mar 01/03/22**		
41	Elaboración clases de diseños	2 días	vie 25/02/22	mar 01/03/22	39	Jairo Sánchez
42	**Implementación**	**1 día?**	**mar 01/03/2**	**mié 02/03/22**		
43	Programas para el funcionamiento del lenguaje y entorno	1 día?	mar 01/03/2	mié 02/03/22	41	Jairo Sánchez
44	**Pruebas**	**1 día?**	**mar 01/03/2**	**mié 02/03/22**		
45	Ejecución de dichos programas	1 día?	mar 01/03/2	mié 02/03/22	41	Jairo Sánchez
46	Fin Iteración 1	0 días	mié 02/03/2	mié 02/03/22	33	
47	**Iteración 2**	**13 días?**	**mié 02/03/2**	**lun 21/03/22**	**46**	
48	**Modelado del negocio**	**0,5 días?**	**mié 02/03/2**	**mié 02/03/22**		
49	Investigación de recursos	0,5 días?	mié 02/03/2	mié 02/03/22		Jairo Sánchez
50	**Requisitos**	**3,5 días?**	**mié 02/03/2**	**lun 07/03/22**		
51	Descripción de casos de uso del paquete "Gestión de Ligas"	2 días	mié 02/03/2	vie 04/03/22		Jairo Sánchez
52	Descripción de casos de uso del paquete "Gestión de Roster"	1,5 días?	vie 04/03/22	lun 07/03/22	51	Jairo Sánchez
53	**Análisis**	**3,5 días**	**mar 08/03/2**	**vie 11/03/22**	**52**	
54	Elaboración de diagramas de secuencia del paquete "Gestión de Roster"	1,5 días	mar 08/03/2	mié 09/03/22	52	Jairo Sánchez
55	Elaboración de diagramas de secuencia del paquete "Gestión de Ligas"	2 días	mié 09/03/2	vie 11/03/22	54	Jairo Sánchez
56	**Diseño**	**5 días**	**vie 11/03/22**	**vie 18/03/22**		
57	Realización casos de uso diseño del paquete "Gestión de Usuario"	2 días	vie 11/03/22	mar 15/03/22	55	Jairo Sánchez
58	Realización casos de uso diseño del paquete "Gestión de Ligas"	3 días	mar 15/03/2	vie 18/03/22	57	Jairo Sánchez
59	**Implementación**	**1 día?**	**vie 18/03/22**	**lun 21/03/22**		
60	Creación primeras clases	1 día?	vie 18/03/22	lun 21/03/22	58	Jairo Sánchez
61	**Prueba**	**0,5 días**	**vie 18/03/22**	**vie 18/03/22**		
62	Depuración de clases	0,5 días	vie 18/03/22	vie 18/03/22	58	Jairo Sánchez
63	Fin Iteración 2	0 días	lun 21/03/22	lun 21/03/22	47	

Ilustración 9: Tareas fase Elaboración I

64	**Iteración 3**	**9,5 días?**	**lun 21/03/22**	**vie 01/04/22**	**63**	**Jairo Sánchez**
65	**Modelado del negocio**	**1 día?**	**lun 21/03/22**	**mar 22/03/22**		
66	Investigación sobre recursos/necesidades	1 día?	lun 21/03/22	mar 22/03/22		Jairo Sánchez
67	**Requisitos**	**2 días**	**mar 22/03/2**	**jue 24/03/22**		
68	Descripción de casos de uso del paquete "Gestión de Mercado"	2 días	mar 22/03/2	jue 24/03/22	66	Jairo Sánchez
69	**Análisis**	**1 día?**	**jue 24/03/22**	**vie 25/03/22**		
70	Elaboración diagramas de secuencia del paquete "Gestión de Mercado"	1 día?	jue 24/03/22	vie 25/03/22	68	Jairo Sánchez
71	**Diseño**	**4,5 días?**	**vie 25/03/22**	**jue 31/03/22**		
72	Realización casos de uso diseño del paquete "Gestión de Roster"	2 días	vie 25/03/22	mar 29/03/22	70	Jairo Sánchez
73	Realización casos de uso diseño del paquete "Gestión de Mercado"	2,5 días?	mar 29/03/2	jue 31/03/22	72	Jairo Sánchez
74	**Implementación**	**1 día?**	**vie 01/04/22**	**vie 01/04/22**		
75	Creación primeras vistas	1 día?	vie 01/04/22	vie 01/04/22	73	Jairo Sánchez
76	**Prueba**	**0,5 días?**	**vie 01/04/22**	**vie 01/04/22**		
77	Interacción y funcionamiento primeras vistas	0,5 días?	vie 01/04/22	vie 01/04/22	73	Jairo Sánchez
78	Fin Iteración 3	1 día?	lun 04/04/22	lun 04/04/22	64	
79	Fin Elaboración	0 días	lun 04/04/22	lun 04/04/22	32	

Ilustración 10: Tareas fase Elaboración II

80	**Construcción**	**52,5 días?**	**mar 05/04/2**	**jue 16/06/22**	**79**	
81	**Iteración 1**	**11 días?**	**mar 05/04/2**	**mar 19/04/22**		
82	**Modelado del negocio**	**0,5 días?**	**mar 05/04/2**	**mar 05/04/22**		
83	Investigación de recursos y herramientas	0,5 días?	mar 05/04/2	mar 05/04/22		Jairo Sánchez
84	**Requisitos**	**0,5 días?**	**mar 05/04/2**	**mar 05/04/22**		
85	Revisión de requisitos funcionales	0,5 días?	mar 05/04/2	mar 05/04/22	83	Jairo Sánchez
86	**Análisis**	**1 día?**	**mié 06/04/2**	**mié 06/04/22**		
87	Diagramas de comunicación	1 día?	mié 06/04/2	mié 06/04/22	85	Jairo Sánchez
88	**Diseño**	**1,5 días?**	**jue 07/04/22**	**vie 08/04/22**		
89	Patrón de diseño y patrón arquitectónico	1,5 días?	jue 07/04/22	vie 08/04/22	87	Jairo Sánchez
90	**Implementación**	**7 días**	**vie 08/04/22**	**mar 19/04/22**		
91	Elaboración subsistema "Gestión de Usuarios"	7 días	vie 08/04/22	mar 19/04/22	89	Jairo Sánchez
92	**Pruebas**	**0,5 días?**	**mar 19/04/2**	**mar 19/04/22**		
93	Pruebas atómicas de cada componente/vista	0,5 días?	mar 19/04/2	mar 19/04/22	91	Jairo Sánchez
94	Pruebas atómicas de cada servicio	0,5 días	mar 19/04/2	mar 19/04/22	91	Jairo Sánchez
95	Fin Iteración 1	0 días	mar 19/04/2	mar 19/04/22	81	
96	**Iteración 2**	**16 días?**	**mié 20/04/2**	**mié 11/05/22**	**95**	
97	**Modelado del negocio**	**0,5 días**	**mié 20/04/2**	**mié 20/04/22**		
98	Investigación sobre recursos y herramientas	0,5 días	mié 20/04/2	mié 20/04/22		Jairo Sánchez
99	**Requisitos**	**0,5 días**	**mié 20/04/2**	**mié 20/04/22**		
100	Revisión requisitos no funcionales	0,5 días	mié 20/04/2	mié 20/04/22		Jairo Sánchez
101	**Análisis**	**0,5 días?**	**mié 20/04/2**	**mié 20/04/22**		
102	Vista de arquitectura	0,5 días?	mié 20/04/2	mié 20/04/22		Jairo Sánchez
103	**Diseño**	**1 día?**	**mié 20/04/2**	**jue 21/04/22**		
104	Refinamiento casos de uso diseño	1 día?	mié 20/04/2	jue 21/04/22	102	Jairo Sánchez
105	**Implementación**	**14 días**	**jue 21/04/22**	**mié 11/05/22**		
106	Elaboración subsistema "Gestión de Ligas"	14 días	jue 21/04/22	mié 11/05/22	104	Jairo Sánchez
107	**Pruebas**	**0,5 días**	**mié 11/05/2**	**mié 11/05/22**		
108	Pruebas atómicas de cada componente/vista	0,5 días	mié 11/05/2	mié 11/05/22	106	Jairo Sánchez
109	Pruebas atómicas de cada servicio	0,5 días	mié 11/05/2	mié 11/05/22	106	Jairo Sánchez
110	Fin Iteración 2	0 días	mié 11/05/2	mié 11/05/22	96	

Ilustración 11: Tareas fase Construcción I

111	**Iteración 3**	**14,5 días?**	**jue 12/05/22**	**mié 01/06/22**	**110**	
112	**Modelado del negocio**	**0,5 días?**	**jue 12/05/22**	**jue 12/05/22**		
113	Investigación sobre recursos y herramientas	0,5 días?	jue 12/05/22	jue 12/05/22		Jairo Sánchez
114	**Requisitos**	**1,5 días?**	**jue 12/05/22**	**vie 13/05/22**		
115	Elaboración documentación	1,5 días?	jue 12/05/22	vie 13/05/22		Jairo Sánchez
116	**Análisis**	**0,5 días**	**vie 13/05/22**	**vie 13/05/22**		
117	Refinamiento diagramas de secuencia y clases de diseño	0,5 días	vie 13/05/22	vie 13/05/22	115	Jairo Sánchez
118	**Diseño**	**0,5 días?**	**vie 13/05/22**	**vie 13/05/22**		
119	Confirmación modelo del diseño	0,5 días?	vie 13/05/22	vie 13/05/22	115	Jairo Sánchez
120	**Implementación**	**12 días**	**lun 16/05/22**	**mar 31/05/22**		
121	Elaboración subsistema"Gestión de Mercado"	12 días	lun 16/05/22	mar 31/05/22	119	Jairo Sánchez
122	**Pruebas**	**0,5 días?**	**mié 01/06/2**	**mié 01/06/22**		
123	Pruebas atómicas de cada componente/vista	0,5 días?	mié 01/06/2	mié 01/06/22	121	Jairo Sánchez
124	Pruebas atómicas de cada servicio	0,5 días?	mié 01/06/2	mié 01/06/22	121	Jairo Sánchez
125	Fin Iteración 3	0 días	mié 01/06/2	mié 01/06/22	111	
126	**Iteración 4**	**11 días?**	**mié 01/06/2**	**jue 16/06/22**	**125**	
127	**Modelado del negocio**	**0,5 días**	**mié 01/06/2**	**mié 01/06/22**		
128	Investigación sobre recursos y herramientas	0,5 días	mié 01/06/2	mié 01/06/22		Jairo Sánchez
129	**Requisitos**	**1 día?**	**mié 01/06/2**	**jue 02/06/22**		
130	Elaboración documentación	1 día?	mié 01/06/2	jue 02/06/22		Jairo Sánchez
131	**Análisis**	**1 día**	**jue 02/06/22**	**vie 03/06/22**		
132	Elaboración documentación	1 día	jue 02/06/22	vie 03/06/22	130	Jairo Sánchez
133	**Diseño**	**2 días**	**vie 03/06/22**	**mar 07/06/22**		
134	Elaboración documentación	2 días	vie 03/06/22	mar 07/06/22	132	Jairo Sánchez
135	**Implementación**	**7 días**	**mar 07/06/2**	**jue 16/06/22**		
136	Elaboración subsistema "Gestión de Roster"	7 días	mar 07/06/2	jue 16/06/22	134	Jairo Sánchez
137	**Pruebas**	**0,5 días**	**mar 07/06/2**	**mar 07/06/22**		
138	Pruebas atómicas de cada componente/vista	0,5 días	mar 07/06/2	mar 07/06/22	134	Jairo Sánchez
139	Pruebas atómicas de cada servicio	0,5 días	mar 07/06/2	mar 07/06/22	134	Jairo Sánchez
140	Fin Iteración 4	0 días	jue 16/06/22	jue 16/06/22	126	
141	Fin Construcción	0 días	jue 16/06/22	jue 16/06/22	80	

Ilustración 12: Tareas fase Construcción II

142	**Transición**	**13 días?**	**jue 16/06/22**	**mar 05/07/22**	**141**	
143	**Iteración 1**	**7 días?**	**jue 16/06/22**	**lun 27/06/22**		
144	**Requisitos**	**0,5 días?**	**jue 16/06/22**	**jue 16/06/22**		
145	Refinamiento de documentación	0,5 días?	jue 16/06/22	jue 16/06/22		Jairo Sánchez
146	**Análisis**	**0,5 días**	**jue 16/06/22**	**jue 16/06/22**		
147	Refinamiento de documentación	0,5 días	jue 16/06/22	jue 16/06/22		Jairo Sánchez
148	**Diseño**	**0,5 días?**	**vie 17/06/22**	**vie 17/06/22**		
149	Refinamiento de documentación	0,5 días?	vie 17/06/22	vie 17/06/22	147	Jairo Sánchez
150	**Implmentación**	**6 días**	**vie 17/06/22**	**lun 27/06/22**		
151	Resolución de problemas y mejora del código	6 días	vie 17/06/22	lun 27/06/22	149	Jairo Sánchez
152	**Pruebas**	**3 días**	**vie 17/06/22**	**mié 22/06/22**		
153	Pruebas de integración entre subsistemas	3 días	vie 17/06/22	mié 22/06/22	149	Jairo Sánchez
154	Pruebas generales del sistema	3 días	vie 17/06/22	mié 22/06/22	149	Jairo Sánchez
155	Fin Iteración 1	0 días	lun 27/06/22	lun 27/06/22	143	
156	**Iteración 2**	**6 días**	**lun 27/06/22**	**mar 05/07/22**	**155**	
157	**Implementación**	**4,5 días**	**lun 27/06/22**	**vie 01/07/22**		
158	Refinamiento del código	2 días	lun 27/06/22	mié 29/06/22		Jairo Sánchez
159	Elaborar documentación	2,5 días	mié 29/06/2:	vie 01/07/22	158	Jairo Sánchez
160	**Pruebas**	**1,5 días**	**lun 04/07/22**	**mar 05/07/22**		
161	Simulación real del sistema	1,5 días	lun 04/07/22	mar 05/07/22	159	Jairo Sánchez
162	Fin Iteración 2	0 días	mar 05/07/2:	mar 05/07/22	156	
163	Fin Transición	0 días	mar 05/07/2:	mar 05/07/22	142	
164	Fin Proyecto	0 días	mar 05/07/2:	mar 05/07/22	1	

Ilustración 13:Tareas fase Transición

Por lo tanto, según la planificación de estas tareas expuestas, el proyecto comienza el 31 de enero y finaliza el 5 de julio, lo que hace un total de 111,5 días laborables para llevarlo a cabo. Esta cifra de días dista con la obtenida en la estimación realizada en el *Apartado 2* en 23 días aproximadamente. El resultado conseguido en esta sección se asemeja más a la realidad y entra dentro de los plazos de entrega del proyecto, incluso produciéndose algún retraso en él cabría la posibilidad de entregarlo dentro de estos plazos. Así mismo, sería posible utilizar sábados y domingos para seguir con el desarrollo y de esta manera conseguir una mayor holgura.

1	**Planificación Temporal**	**111,5 días?**	**lun 31/01/22**	**mar 05/07/22**	
2	**Inicio**	**14,5 días?**	**lun 31/01/22**	**vie 18/02/22**	
31	Fin Inicio	0 días	vie 18/02/22	vie 18/02/22	2
32	**Elaboración**	**31,5 días?**	**vie 18/02/22**	**lun 04/04/22**	**31**
79	Fin Elaboración	0 días	lun 04/04/22	lun 04/04/22	32
80	**Construcción**	**52,5 días?**	**mar 05/04/2:**	**jue 16/06/22**	**79**
141	Fin Construcción	0 días	jue 16/06/22	jue 16/06/22	80
142	**Transición**	**13 días?**	**jue 16/06/22**	**mar 05/07/22**	**141**
163	Fin Transición	0 días	mar 05/07/2:	mar 05/07/22	142
164	Fin Proyecto	0 días	mar 05/07/2:	mar 05/07/22	1

Ilustración 14: Tiempo total y de cada fase

Por último, se muestra el diagrama de Gantt resultante de esta planificación:

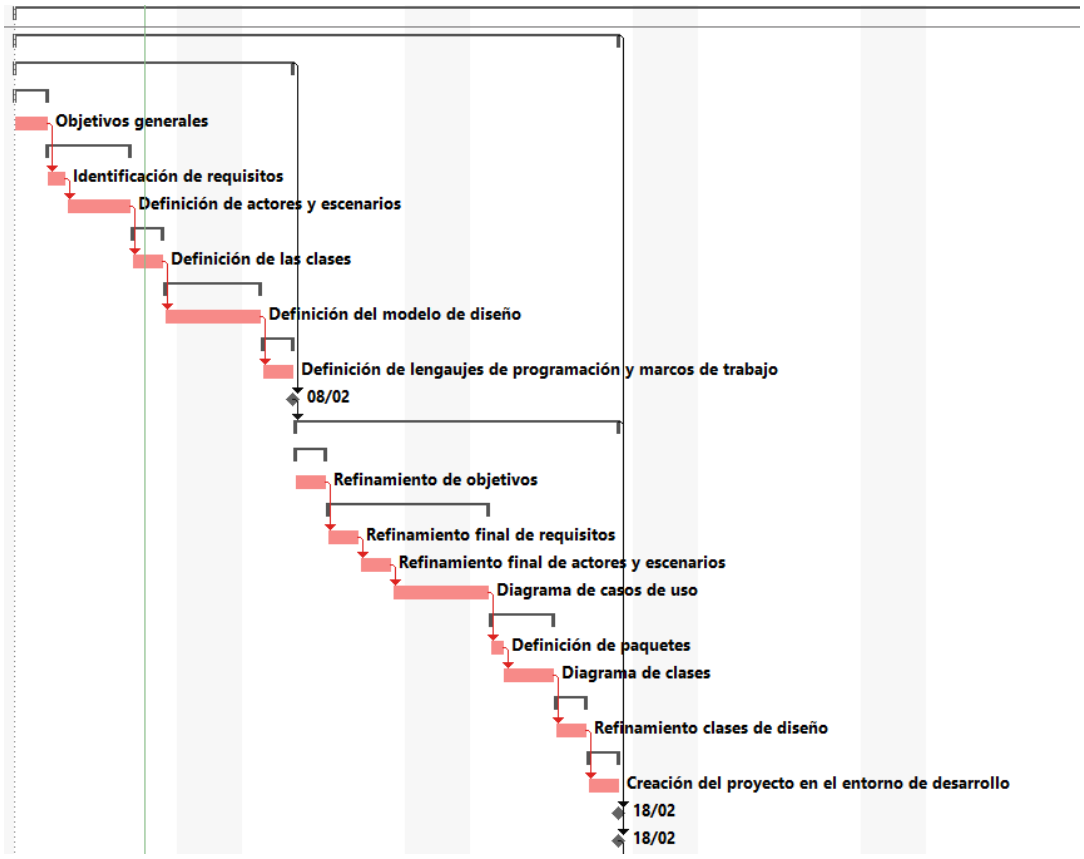


Ilustración 15: Diagrama de Gantt I

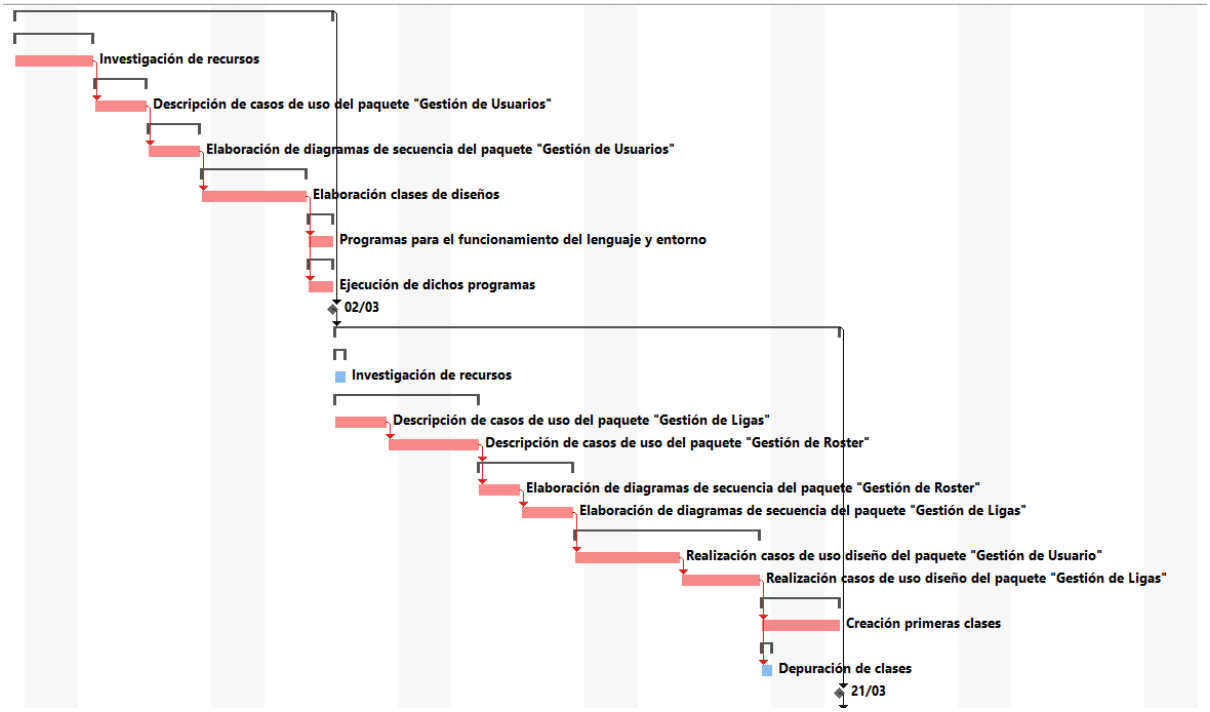


Ilustración 16: Diagrama de Gantt II

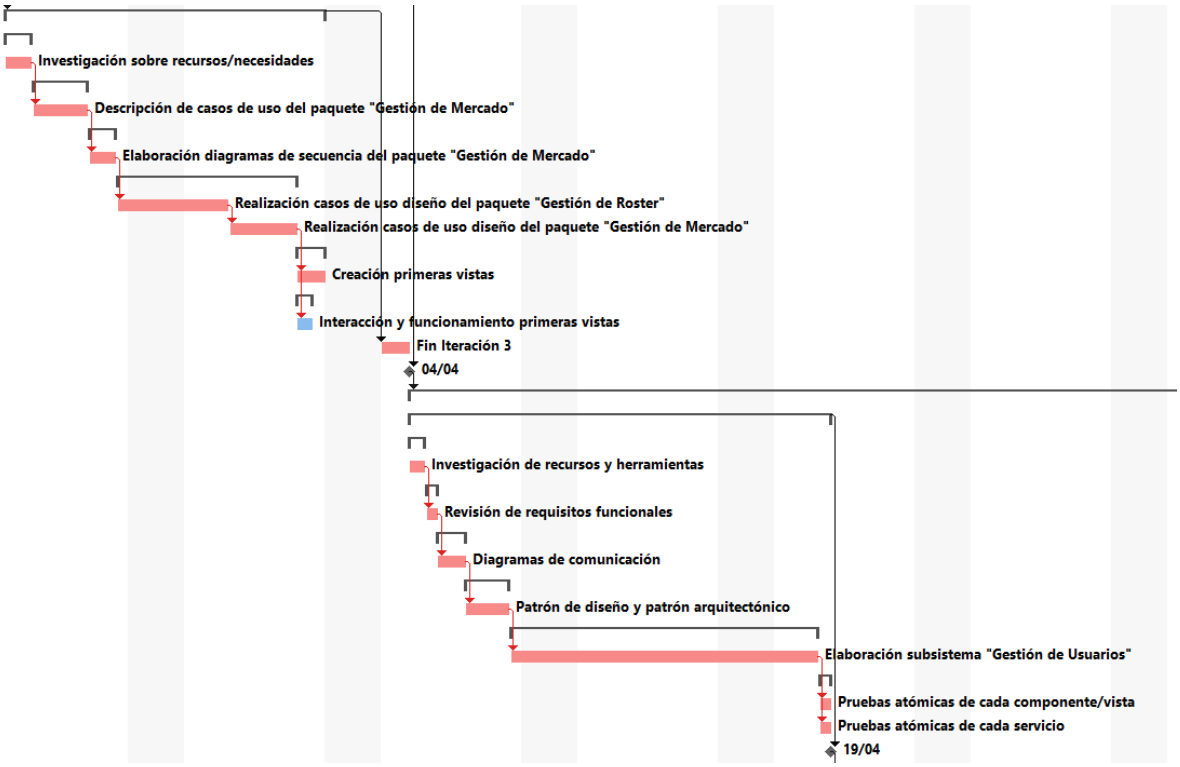


Ilustración 17: Diagrama de Gantt III

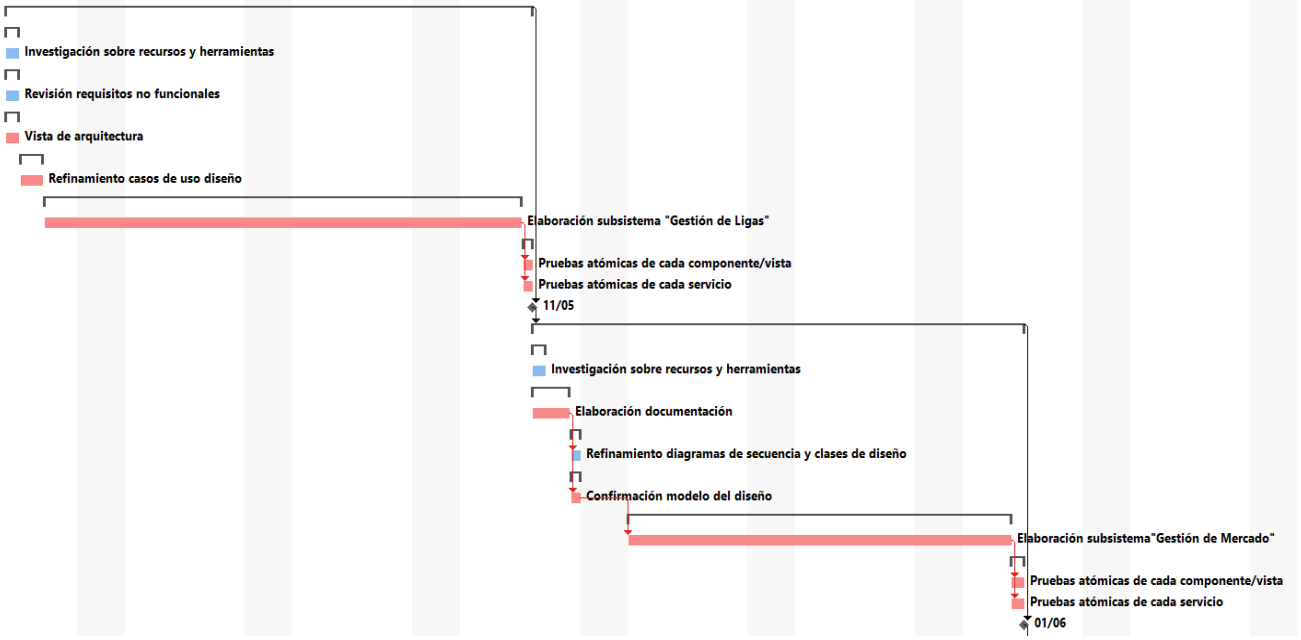


Ilustración 18: Diagrama de Gantt IV

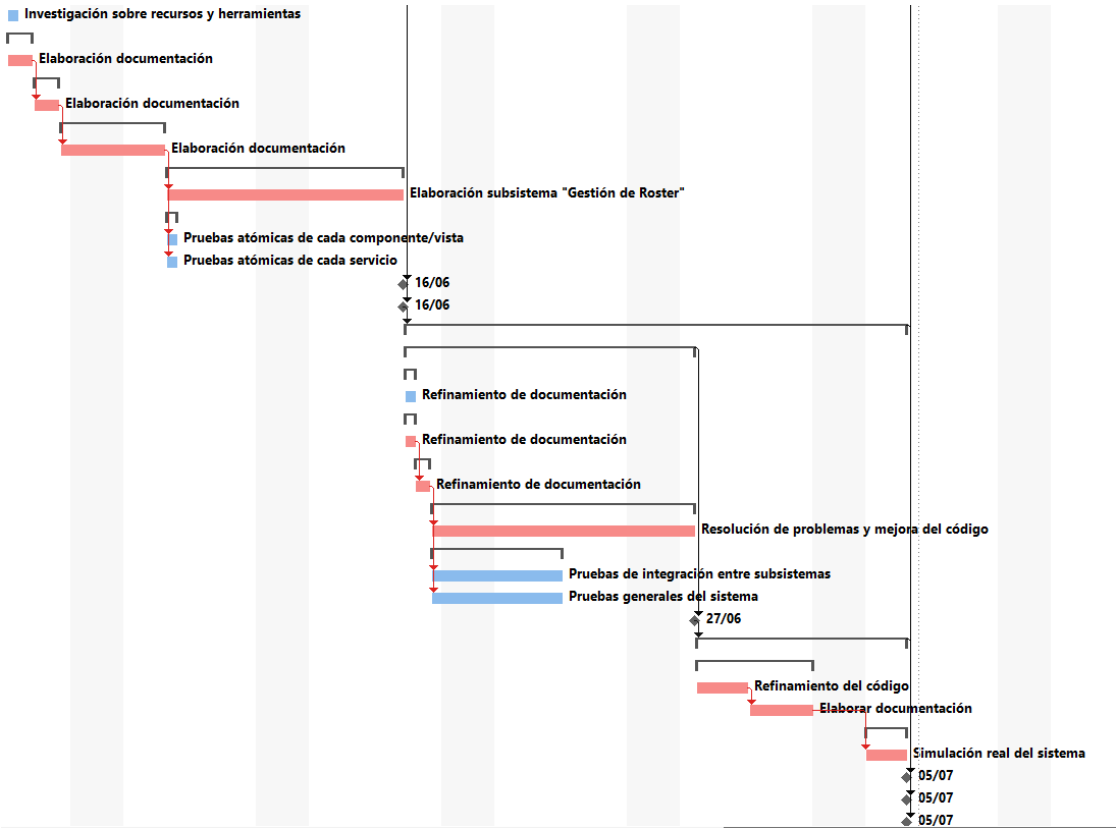


Ilustración 19: Diagrama de Gantt V

Un último aspecto para destacar es que al tratarse de un proyecto con un sólo desarrollador y unas fechas de entrega tan marcadas es inevitable en algunos puntos no tener una sobrecarga de trabajo, en proyectos reales se debería de evitar esta consigna.

4. BIBLIOGRAFÍA

Documentación Microsoft Project. (18 de 03 de 2022). Obtenido de https://docs.microsoft.com/es-es/projectonline/project-online

Moreno García, M. N. *Práctica 1, Estimación del esfuerzo, Gestión de proyectos.*

Moreno García, M. N. *Tema 2, Medición del software, Gestión de proyectos.*

Navarro Cáceres, M., & N., M. G. *Práctica 2, Planificación temporal, Gestión de proyectos.*

ANEXO II: Especificación de requisitos software

Dream NBA: aplicación para la creación de ligas virtuales y gestión de un equipo propio

Trabajo de Fin de Grado

INGENIERÍA INFORMÁTICA

VNiVERSiDAD D SALAMANCA

CAMPUS DE EXCELENCIA INTERNACIONAL

Julio 2022

Autor

Jairo Sánchez García

Tutores

Gabriel Villarrubia González

Juan Francisco de Paz Santana

Héctor Sánchez San Blas

TABLA DE CONTENIDO

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

El objetivo de este anexo es la especificación de requisitos software para el sistema que se expone.

El sistema trata de una aplicación web en donde el usuario puede crear y participar en ligas virtuales junto con otros usuarios para competir entre sí. Además tendrán que gestionar un equipo por cada liga de formen parte, mediante la compraventa de jugadores.

Se va a seguir la estructura proporcionada por la metodología de Durán y Bernárdez para la obtención y documentación de requisitos, siendo esta la siguiente:

- Participantes en el proyecto.
- Objetivos del sistema.
- Catálogo de requisitos del sistema.
    - Requisitos de información.
    - Requisitos funcionales.
        - Diagramas de casos de uso.
        - Definición de actores.
        - Casos de uso del sistema.
    - Requisitos no funcionales.
- Matriz de rastreabilidad objetivos/requisitos.

2. PARTICIPANTES

Se cuenta con cuatro participantes para el desarrollo de este proyecto, donde uno es el alumno y los tres restantes son los tutores. Todos pertenecen a la misma organización que es la Universidad de Salamanca.

Participante	**Sánchez García, Jairo**
Organización	Universidad de Salamanca
Rol	Desarrollador
Desarrollador	Sí
Cliente	No
Usuario	No
Comentarios	Ninguno

Tabla 1: Participante Sánchez García, Jairo

Participante	**Villarrubia González, Gabriel**
Organización	Universidad de Salamanca
Rol	Tutor
Desarrollador	No
Cliente	No
Usuario	No
Comentarios	Ninguno

Tabla 2: Participante Villarrubia González, Gabriel

Participante	**Sánchez San Blas, Héctor**
Organización	Universidad de Salamanca
Rol	Tutor
Desarrollador	No
Cliente	No
Usuario	No
Comentarios	Ninguno

Tabla 3: Sánchez San Blas, Héctor

Participante	De Paz Santana, Juan Francisco
Organización	Universidad de Salamanca
Rol	Tutor
Desarrollador	No
Cliente	No
Usuario	No
Comentarios	Ninguno

Tabla 4: Participante De Paz Santana, Juan Francisco

3. OBJETIVOS DEL SISTEMA

En este apartado se van a exponer los objetivos que el sistema tiene que cumplir para llevar a cabo las funcionalidades de este.

OBJ-0001	Gestión de los usuarios
Versión	1.0
Autores	Jairo Sánchez García
Fuentes	
Descripción	El sistema deberá permitir a los usuarios darse de alta en el sistema, entrar y salir de él. Además debe permitir la modificación de sus datos.
Subobjetivos	Ninguno
Importancia	Vital
Urgencia	Inmediata
Estado	Validado
Estabilidad	Alta
Comentarios	Ninguno

Tabla 5: OBJ-0001 Gestión de los usuario

OBJ-0002	Gestión de las ligas
Versión	1.0
Autores	Jairo Sánchez García
Fuentes	
Descripción	El sistema deberá permitir la creación de ligas, la adhesión y salida de los usuarios a estas. Así como el correcto funcionamiento dentro de cada liga, como el control de la clasificación, el mercado o los equipos.
Subobjetivos	Ninguno
Importancia	Vital
Urgencia	Inmediata
Estado	Validado
Estabilidad	Alta
Comentarios	Ninguno

Tabla 6: OBJ-0002 Gestión de las ligas

OBJ-0003	Gestión de las jornadas
Versión	1.0
Autores	Jairo Sánchez García
Fuentes	
Descripción	El sistema deberá generar correctamente la puntuación de cada jornada en función de las estadísticas. Además de la actualización del presupuesto y de la puntuación total.
Subobjetivos	Ninguno
Importancia	Vital
Urgencia	Inmediata
Estado	Validado
Estabilidad	Alta
Comentarios	Ninguno

Tabla 7: OBJ-003 Gestión de las jornadas

OBJ-0004	Gestión del roster
Versión	1.0
Autores	Jairo Sánchez García
Fuentes	
Descripción	El sistema deberá generar un roster para cada usuario de manera equilibrada, además de permitir al propietario poner y quitar a los jugadores del quinteto inicial o poner a la venta estos.
Subobjetivos	Ninguno
Importancia	Vital
Urgencia	Inmediata
Estado	Validado
Estabilidad	Alta
Comentarios	Ninguno

Tabla 8: OBJ-0004 Gestión del roster

OBJ-0005	Gestión del mercado
Versión	1.0
Autores	Jairo Sánchez García
Fuentes	
Descripción	El sistema deberá permitir pujar por jugadores, modificar o eliminar las pujas realizadas, vender y quitar jugadores propios.
Subobjetivos	Ninguno
Importancia	Vital
Urgencia	Inmediata
Estado	Validado
Estabilidad	Alta
Comentarios	Ninguno

Tabla 9: OBJ-0005 Gestión del mercado

OBJ-0006	Gestión del calendario
Versión	1.0
Autores	Jairo Sánchez García
Fuentes	
Descripción	El sistema deberá permitir visualizar los partidos y resultados por fecha, así como las estadísticas por parte de los jugadores.
Subobjetivos	Ninguno
Importancia	Vital
Urgencia	Inmediata
Estado	Validado
Estabilidad	Alta
Comentarios	Ninguno

Tabla 10: OBJ-0006 Gestión del calendario

OBJ-0007	Gestión de las noticias
Versión	1.0
Autores	Jairo Sánchez García
Fuentes	
Descripción	El sistema deberá permitir visualizar las noticias relacionadas con este.
Subobjetivos	Ninguno
Importancia	Vital
Urgencia	Inmediata
Estado	Validado
Estabilidad	Alta
Comentarios	Ninguno

Tabla 11: OBJ-0007 Gestión de las noticias

4. CATÁLOGO DE REQUISITOS DEL SISTEMAS

4.1 Requisitos de información

A continuación, se van a ver los requisitos de información, los cuales indican los datos que el sistema deberá almacenar.

IRQ-0001	**Información sobre los usuarios**	
Versión	1.0	
Autores	Jairo Sánchez García	
Fuentes		
Dependencias	Ninguno	
Descripción	El sistema deberá almacenar información correspondiente a los usuarios. En concreto:	
Datos específicos	• Email • Logo • Nickname	
Tiempo de vida	**Medio**	**Máximo**
Ocurrencias simultáneas	**Medio**	**Máximo**
Importancia	Importante	
Urgencia	Inmediata	
Estado	Validado	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla 12: IRQ-0001 Información sobre los usuarios

IRQ-0002	**Información sobre los jugadores**
Versión	1.0
Autores	Jairo Sánchez García
Fuentes	
Dependencias	Ninguno
Descripción	El sistema deberá almacenar información correspondiente a los jugadores. En concreto:

Datos específicos	• Nombre • Foto • Posición • Salario • Puntos Totales • Asistencias • Tapones • Fantasy Points • Rebotes • Robos • Pérdidas • Puntos	
Tiempo de vida	**Medio**	**Máximo**
Ocurrencias simultáneas	**Medio**	**Máximo**
Importancia	Importante	
Urgencia	Inmediata	
Estado	Validado	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla 13: IRQ-0002 Información sobre los jugadores

IRQ-0003	**Información sobre las ligas**
Versión	1.0
Autores	Jairo Sánchez García
Fuentes	
Dependencias	• IRQ-0001 Información sobre los usuarios • IRQ-0004 Información sobre el mercado • IRQ-000 Información sobre los roster
Descripción	El sistema deberá almacenar información correspondiente a las ligas. En concreto:
Datos específicos	• Id • Miembros • Puntos por miembro • Presupuesto por miembro

	• Mercado • Rosters individuales	
Tiempo de vida	**Medio**	**Máximo**
Ocurrencias simultáneas	**Medio**	**Máximo**
Importancia	Importante	
Urgencia	Inmediata	
Estado	Validado	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla 14: IRQ-0003 Información sobre las ligas

IRQ-0004	**Información sobre los roster**	
Versión	1.0	
Autores	Jairo Sánchez García	
Fuentes		
Dependencias	• IRQ-002 Información sobre los jugadores	
Descripción	El sistema deberá almacenar información correspondiente a los roster de cada usuario. En concreto:	
Datos específicos	• Nombre de usuario • Jugadores • Starters que haya guardado el usuario	
Tiempo de vida	**Medio**	**Máximo**
Ocurrencias simultáneas	**Medio**	**Máximo**
Importancia	Importante	
Urgencia	Inmediata	
Estado	Validado	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla 15: IRQ-0004 Información sobre los roster

IRQ-0005	**Información sobre las jornadas**	
Versión	1.0	
Autores	Jairo Sánchez García	
Fuentes		
Dependencias	• IRQ-002 Información sobre los jugadores	
Descripción	El sistema deberá almacenar información correspondiente a las jornadas. En concreto:	
Datos específicos	• Id • Id de liga • Nombre usuario • Puntos jornada • Starters de cada usuario de cada liga en la jornada en concreto • Jugadores con sus estadísticas que participaron en la jornada	
Tiempo de vida	**Medio**	**Máximo**
Ocurrencias simultáneas	**Medio**	**Máximo**
Importancia	Importante	
Urgencia	Inmediata	
Estado	Validado	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla 16: IRQ-0005 Información sobre las jornadas

IRQ-0006	**Información sobre las noticias**	
Versión	1.0	
Autores	Jairo Sánchez García	
Fuentes		
Dependencias	Ninguno	
Descripción	El sistema deberá almacenar información correspondiente a las noticias generadas en el sistema. En concreto:	
Datos específicos	• Id • Id de liga • Día • Mes • Año • Información • Tipo • Nombre usuario	
Tiempo de vida	**Medio**	**Máximo**
Ocurrencias simultáneas	**Medio**	**Máximo**
Importancia	Importante	
Urgencia	Inmediata	
Estado	Validado	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla 17: IRQ-0006 Información sobre las noticias

IRQ-0007	**Información sobre el mercado**
Versión	1.0
Autores	Jairo Sánchez García
Fuentes	
Dependencias	Ninguno
Descripción	El sistema deberá almacenar información correspondiente al mercado en cada liga. En concreto:
Datos específicos	• Nombre del jugador • Foto del jugador

	• Puntos del jugador • Posición del jugador • Salario del jugador • Propietario • Pujador • Dinero de puja	
Tiempo de vida	**Medio**	**Máximo**
Ocurrencias simultáneas	**Medio**	**Máximo**
Importancia	Importante	
Urgencia	Inmediata	
Estado	Validado	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla 18: IRQ-0007 Información sobre el mercado

4.2 Requisitos funcionales

En este apartado se van a definir los requisitos funcionales que el sistema deberá proporcionar y la manera en que actúan en cada caso.

4.2.1 Diagrama de paquetes

En la *Ilustración 1* se muestra el Diagrama de Paquetes, en donde se aprecia una jerarquía de los paquetes que conforman el sistema. Estos paquetes son: Gestión de Usuarios, Gestión de Ligas, Gestión de Mercado y Gestión de Roster; en los apartados posteriores se entrará en detalle de la composición de cada uno.

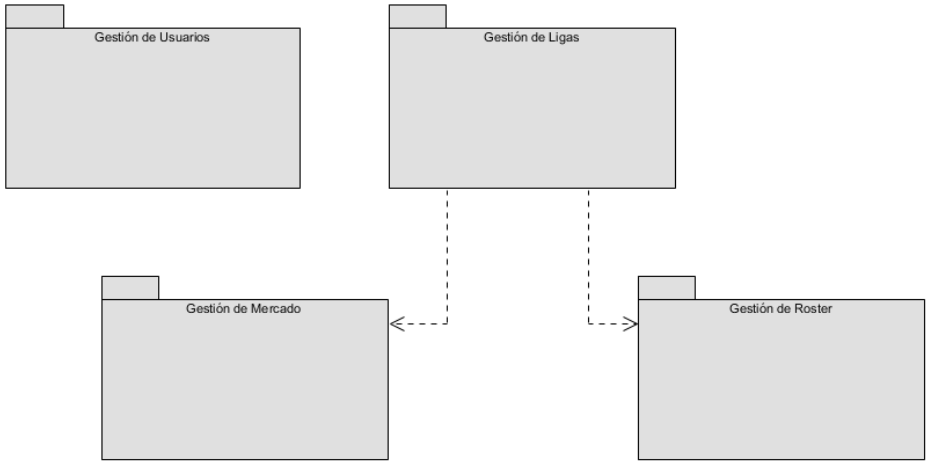


Ilustración 1: Diagrama de Paquetes

4.2.2 Definición de actores

En el sistema participarán tres tipos de actores, definidos a continuación.

- Actor Usuario no logueado: se trata de cualquier persona que interactúe con el sistema pero que no esté dado de alta en él.
- Actor Usuario logueado: es el usuario que se da alta en el sistema y accede a este a través de una autenticación.
- Actor Temporizador: actor que actualiza periódicamente algunas tareas del sistema necesarias para el correcto funcionamiento de este, además de la consistencia de datos.

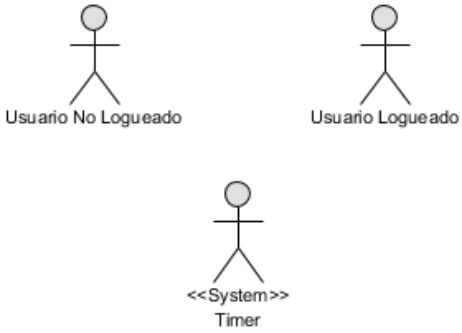


Ilustración 2: Actores del sistema

En las siguientes tablas se realiza la especificación de estos actores:

ACT-0001	**Usuario No Logueado**
Versión	1.0
Autores	Jairo Sánchez García
Fuentes	
Descripción	Actor que va a interactuar con el sistema pero sin estar dado de alta en él.
Comentarios	Ninguno

Tabla 19: ACT-0001 Usuario No Logueado

ACT-0002	**Usuario Logueado**
Versión	1.0
Autores	Jairo Sánchez García
Fuentes	
Descripción	Actor el cual se ha dado de alta en el sistema y ha pasado por un proceso de autenticación para acceder a este.
Comentarios	Ninguno

Tabla 20: ACT-0002 Usuario Logueado

ACT-0003	Timer
Versión	1.0
Autores	Jairo Sánchez García
Fuentes	
Descripción	Actor va a lanzar algunas tareas de manera periódica o cuando sea necesario para el correcto funcionamiento del sistema.
Comentarios	Ninguno

Tabla 21: ACT-0003 Timer

4.2.3 Casos de uso del sistema

Para definir las secuencias de acciones que el sistema tiene que llevar a cabo, se va a realizar la especificación de casos de uso.

4.2.3.1 Gestión de Usuario

Este paquete se compone con los casos de uso, expuestos en la *Ilustración 3*, relacionados con el manejo de las acciones que podrán hacer los usuarios antes de entrar al sistema y con las correspondientes a la modificación de sus datos personales.

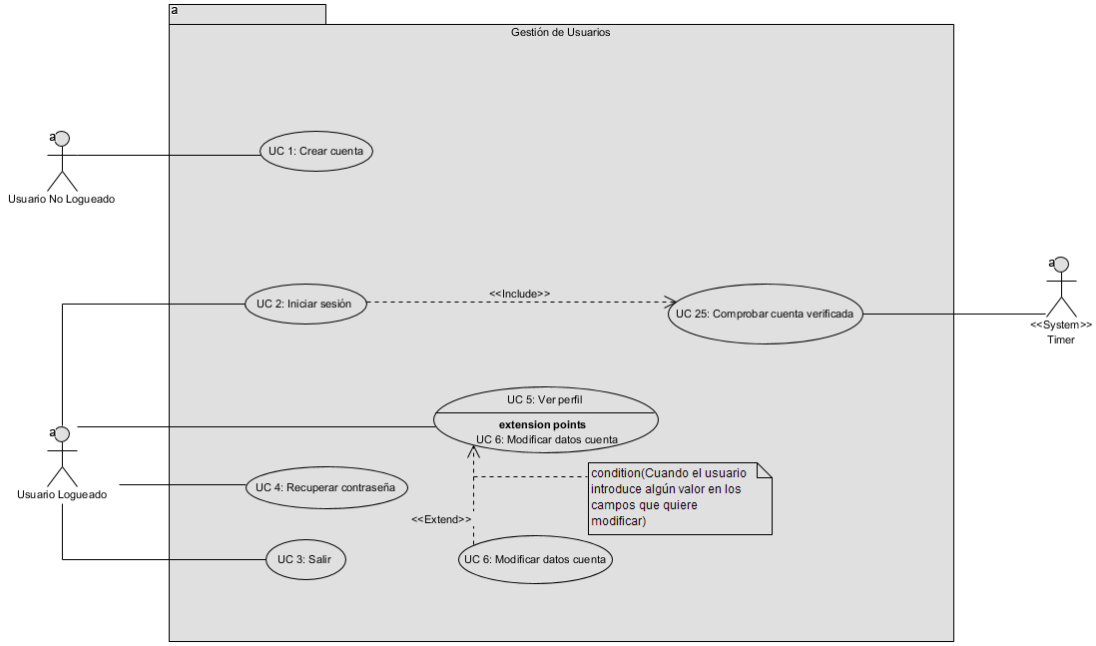


Ilustración 3: Diagrama de casos de uso, Gestión de Usuarios

UC-0001	**Crear cuenta**	
Versión	1.0	
Autores	Jairo Sánchez García	
Fuentes		
Objetivos asociados	• OBJ-0001 Gestión de los usuarios	
Casos de uso asociados		
Descripción	El sistema deberá comportarse tal como se describe a continuación cuando un usuario no logueado seleccione la opción '*Registrarse*'.	
Precondición	Ninguna	
Secuencia normal	**Paso**	**Acción**
	1	El sistema mostrará al usuario un formulario con los campos a rellenar, como nickname, contraseña y logo.
	2	El actor '*Usuario No Logueado*' (ACT-0001), tendrá que rellenar dichos campos con los datos correctos y realizar la opción '*Registro*'.
	3	Si los campos han sido rellenados con datos correctos, seguidamente el sistema enviará un correo para la confirmación de la cuenta.
Postcondición	Ninguna	
Excepciones	**Paso**	**Acción**
	3	En caso de que algún campo tenga un valor incorrecto o el usuario ya posea una cuenta con ese email en el sistema, se le notificará el error. Este caso de uso se repetiría hasta que decida salir de la zona de registro, o se complete satisfactoriamente.
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Frecuencia esperada		
Importancia	Vital	
Urgencia	Inmediata	
Estado	Válido	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla 22: UC-0001 Crear cuenta

UC-0002	**Iniciar sesión**	
Versión	1.0	
Autores	Jairo Sánchez García	
Fuentes		
Objetivos asociados	• OBJ-0001 Gestión de los usuarios	
Casos de uso asociados	• UC-0001 Crear cuenta	
Descripción	El sistema deberá comportarse tal como se describe a continuación, cuando un '*Usuario No Logueado'* (ACT-0001) accede a la página de login del sistema.	
Precondición	Realización anteriormente del caso de uso '*Crear cuenta'* (UC-0001).	
Secuencia normal	**Paso**	**Acción**
	1	El sistema mostrará los campos usuario y contraseña.
	2	El actor '*Usuario No Logueado'* (ACT-0001), tendrá que rellenar dichos campos con los datos correctos.
	3	Se realiza el caso de uso *'Comprobar cuenta verificada'* (UC-0007), para corroborar que el usuario ha verificado su cuenta.
	4	Si los campos han sido rellenados con datos correctos y el usuario tiene la cuenta verificada, seguidamente este accederá al sistema.
Postcondición	Ninguna	
Excepciones	**Paso**	**Acción**
	4	Si el usuario no tiene la cuenta verificada o los datos introducidos en los campos son incorrectos, se le mostrará en cada caso el error. El caso de uso sigue en ejecución hasta que se logue o abandone.
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Frecuencia esperada		
Importancia	Vital	
Urgencia	Inmediata	
Estado	Válido	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla 23: UC-0002 Iniciar sesión

UC-0003	**Salir**	
Versión	1.0	
Autores	Jairo Sánchez García	
Fuentes		
Objetivos asociados	• OBJ-0001 Gestión de los usuarios	
Casos de uso asociados	• UC-0002 Iniciar sesión	
Descripción	El sistema deberá comportarse tal como se describe a continuación, cuando un '*Usuario Logueado*' (ACT-0002) selecciona la opción '*Logout*'	
Precondición	Se ha tenido que realizar el acceso al sistema para poder llevar a cabo esta acción.	
Secuencia normal	**Paso**	**Acción**
	1	El actor solicita salir del sistema.
	2	El sistema cierra la sesión del actor y le redirige a la pantalla de *'Login'*.
Postcondición	El actor se encuentra fuera del sistema.	
Excepciones	**Paso**	**Acción**
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Frecuencia esperada		
Importancia	Vital	
Urgencia	Inmediata	
Estado	Válido	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla 24: UC-0003 Salir

UC-0004	**Recuperar contraseña**	
Versión	1.0	
Autores	Jairo Sánchez García	
Fuentes		
Objetivos asociados	• OBJ-0001 Gestión de los usuarios	
Casos de uso asociados	• UC-0001 Crear cuenta	
Descripción	El sistema deberá comportarse tal como se describe a continuación, cuando un '*Usuario No Logueado'* (ACT-0001) selecciona la opción '*Forgot password?*'	
Precondición	Se necesita que el actor se haya registrado en el sistema.	
Secuencia normal	**Paso**	**Acción**
	1	El sistema mostrará un campo para introducir el email.
	2	El actor rellenará tal campo y enviará la información.
	3	Seguidamente el sistema envía un correo con un link en donde poner la contraseña querida.
	4	El actor rellena el campo con una nueva contraseña y se le redirige a la página de '*Login'*.
Postcondición	Ninguna	
Excepciones	**Paso**	**Acción**
	4	El actor ignora el correo o no introduce la nueva contraseña, el caso de uso quedará sin efecto.
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Frecuencia esperada		
Importancia	Vital	
Urgencia	Inmediata	
Estado	Válido	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla 25: ACT-0004 Recuperar contraseña

UC-0005	**Ver perfil**	
Versión	1.0	
Autores	Jairo Sánchez García	
Fuentes		
Objetivos asociados	• OBJ-0001 Gestión de los usuarios	
Casos de uso asociados	• UC-0001 Iniciar sesión	
Descripción	El sistema deberá comportarse tal como se describe a continuación, cuando un '*Usuario Logueado'* (ACT-0001) solicita ver su perfil.	
Precondición	Se ha tenido que realizar el acceso al sistema para poder llevar a cabo esta acción.	
Secuencia normal	**Paso**	**Acción**
	1	El sistema mostrará al actor la información de su perfil las opciones disponibles para editarlo.
Postcondición	Ninguna	
Excepciones	**Paso**	**Acción**
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Frecuencia esperada		
Importancia	Vital	
Urgencia	Inmediata	
Estado	Válido	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla 26: UC-0005 Ver perfil

UC-0006	**Modificar datos cuenta**	
Versión	1.0	
Autores	Jairo Sánchez García	
Fuentes		
Objetivos asociados	• OBJ-0001 Gestión de los usuarios	
Casos de uso asociados	• UC-0005 Ver perfil	
Descripción	El sistema deberá comportarse tal como se describe a continuación, cuando el actor '*Usuario Logueado*' (ACT-0001) realiza cualquier opción disponible para modificar sus datos.	
Precondición	Haber hecho previamente el caso de uso '*Ver perfil*' (UC-0005).	
Secuencia normal	**Paso**	**Acción**
	1	El sistema mostrará distintos campos para modificar (nickname, logo, contraseña).
	2	El actor rellena el campo nickname/logo y pulsa '*Update.*'
	3	El sistema guarda los nuevos datos.
Postcondición	Los datos se cambian en el perfil del usuario.	
Excepciones	**Paso**	**Acción**
	2	Si el actor selecciona la opción de modificar contraseña, se llevará a cabo el caso de uso '*Recuperar contraseña*' (UC-0004).
	2	En caso de que el actor realizara la acción '*Update*' el caso de uso quedaría sin efecto alguno, así como en el caso de no rellenar ningún campo.
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Frecuencia esperada		
Importancia	Vital	
Urgencia	Inmediata	
Estado	Válido	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla 27: UC-0006 Modificar datos cuenta

UC-0007	**Comprobar cuenta verificada**	
Versión	1.0	
Autores	Jairo Sánchez García	
Fuentes		
Objetivos asociados	• OBJ-0001 Gestión de los usuarios	
Casos de uso asociados	• UC-0002 Iniciar sesión	
Descripción	El sistema deberá comportarse tal como se describe a continuación, cuando se haga el paso 3 del caso de uso '*Iniciar sesión´*(UC-0002).	
Precondición		
Secuencia normal	**Paso**	**Acción**
	1	El sistema comprobará que el usuario tiene la cuenta verificada
Postcondición	Ninguna	
Excepciones	**Paso**	**Acción**
	-	-
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Frecuencia esperada		
Importancia	Vital	
Urgencia	Inmediata	
Estado	Válido	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla 28: UC-0007 Comprobar cuenta verificada

4.2.3.2 Gestión de Ligas

El paquete Gestión de Ligas está compuesto por los casos de uso expuestos en la *Ilustración 4*, los cuales permiten realizar las acciones necesarias para garantizar al sistema el buen manejo de las funcionalidad asociadas a las ligas.

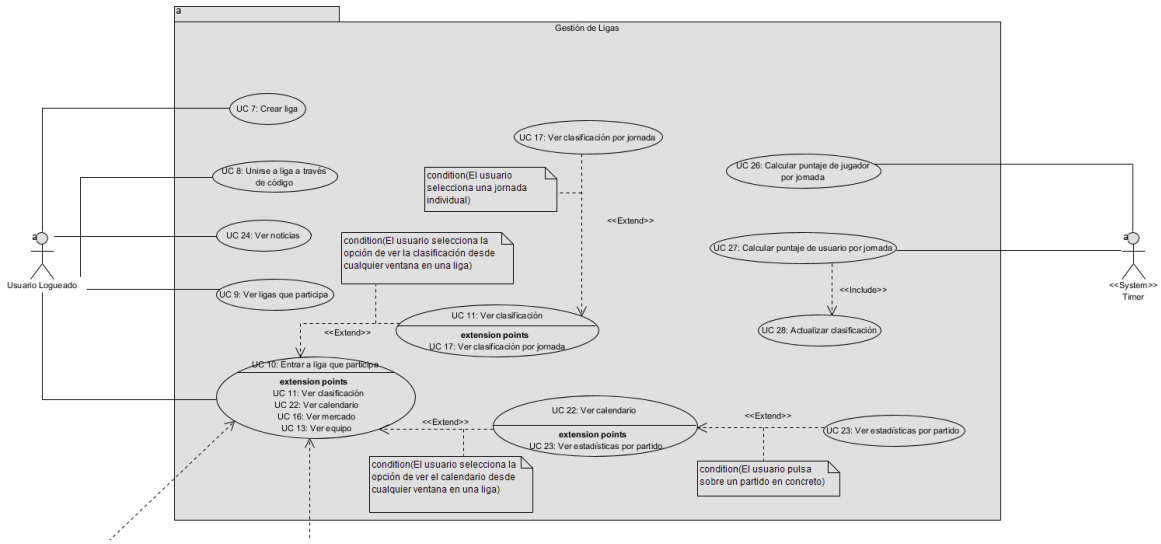


Ilustración 4: Diagrama de casos de uso, Gestión de Ligas

UC-0008	**Crear liga**	
Versión	1.0	
Autores	Jairo Sánchez García	
Fuentes		
Objetivos asociados	• OBJ-0002 Gestión de las ligas	
Casos de uso asociados	• UC-0002 Iniciar sesión	
Descripción	El sistema deberá comportarse tal como se describe a continuación, cuando el actor '*Usuario Logueado*' (ACT-0002) seleccione la opción '*Crear liga*'.	
Precondición	Se añade una nueva liga al sistema.	
Secuencia normal	**Paso**	**Acción**
	1	El sistema mostrará los campos a rellenar.
	2	El usuario rellena los campos y sigue adelante.
	3	Si los campos son correctos, se genera un equipo inicial y el sistema redirige al usuario a la página principal de la liga
Postcondición	Ninguna	

Excepciones	**Paso**	**Acción**
	3	En caso de algún error con los datos, el sistema se lo hará saber al usuario y no redirige al usuario.
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Frecuencia esperada		
Importancia	Vital	
Urgencia	Inmediata	
Estado	Válido	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla 29: UC-0008 Crear liga

UC-0009	**Unirse a liga a través del código**	
Versión	1.0	
Autores	Jairo Sánchez García	
Fuentes		
Objetivos asociados	• OBJ-0002 Gestión de las ligas	
Casos de uso asociados	• UC-0008 Crear liga	
Descripción	El sistema deberá comportarse tal como se describe a continuación, cuando el actor '*Usuario Logueado'* (ACT-0002) seleccione la opción '*Añadir liga'*.	
Precondición	La liga a la que el actor se quiere unir debe existir.	
Secuencia normal	**Paso**	**Acción**
	1	El sistema mostrará el campo a rellenar.
	2	El usuario rellena los campos y continúa.
	3	Si los campos son correctos, se genera un equipo inicial y el sistema redirige al usuario a la página principal de la liga
Postcondición	Se añade un nuevo usuario a liga.	
Excepciones	**Paso**	**Acción**

	3	En caso de algún error con los datos, el sistema se lo hará saber al usuario y no redirige al usuario.
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Frecuencia esperada		
Importancia	Vital	
Urgencia	Inmediata	
Estado	Válido	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla 30: UC-0009 Unirse a liga a través del código

UC-0010	**Ver ligas que participa**	
Versión	1.0	
Autores	Jairo Sánchez García	
Fuentes		
Objetivos asociados	• OBJ-0002 Gestión de las ligas	
Casos de uso asociados	• UC-0008 Crear liga • UC-0009 Unirse a liga a través del código	
Descripción	El sistema deberá comportarse tal como se describe a continuación, cuando el actor '*Usuario Logueado'* (ACT-0002) seleccione la opción '*Ligas*.	
Precondición	Ninguna	
Secuencia normal	**Paso**	**Acción**
	1	El sistema mostrará al usuario las ligas a las que pertenece.
Postcondición	Ninguna	
Excepciones	**Paso**	**Acción**
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Frecuencia esperada		
Importancia	Vital	

Urgencia	Inmediata
Estado	Válido
Estabilidad	Alta
Comentarios	Ninguno

Tabla 31: UC-0010 Ver ligas que participa

UC-0011	**Entrar a liga que participa**	
Versión	1.0	
Autores	Jairo Sánchez García	
Fuentes		
Objetivos asociados	• OBJ-0002 Gestión de las ligas	
Casos de uso asociados	• UC-0010 Ver ligas que participa	
Descripción	El sistema deberá comportarse tal como se describe a continuación, cuando el actor '*Usuario Logueado*' (ACT-0002) seleccione una liga mostrada durante el caso de uso '*Ver ligas que participa*' (UC-0010).	
Precondición	Ninguna	
Secuencia normal	**Paso**	**Acción**
	1	El usuario selecciona la liga a entrar.
	2	El sistema le redirecciona a la página principal de la liga seleccionada
Postcondición	Ninguna	
Excepciones	**Paso**	**Acción**
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Frecuencia esperada		
Importancia	Vital	
Urgencia	Inmediata	
Estado	Válido	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla 32: UC-0011 Entrar a liga que participa

UC-0012	**Ver clasificación**	
Versión	1.0	
Autores	Jairo Sánchez García	
Fuentes		
Objetivos asociados	• OBJ-0002 Gestión de las ligas • OBJ-0003 Gestión de las jornadas	
Casos de uso asociados	• UC-0011 Entrar a liga que participa • UC-0019 Actualizar clasificación	
Descripción	El sistema deberá comportarse tal como se describe a continuación, cuando el actor '*Usuario Logueado'* (ACT-0002) seleccione la opción '*Clasificación'* o se encuentre en la página principal de la liga.	
Precondición	El actor debe estar dentro de una liga.	
Secuencia normal	**Paso**	**Acción**
	1	El sistema mostrará al usuario la clasificación la liga en cuestión.
Postcondición	Ninguna	
Excepciones	**Paso**	**Acción**
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Frecuencia esperada		
Importancia	Vital	
Urgencia	Inmediata	
Estado	Válido	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla 33: UC-0012 Ver clasificación

UC-0013	**Ver clasificación por jornada**	
Versión	1.0	
Autores	Jairo Sánchez García	
Fuentes		
Objetivos asociados	• OBJ-0002 Gestión de las ligas • OBJ-0003 Gestión de las jornadas	
Casos de uso asociados	• UC-0012 Ver clasificación • UC-0018 Calcular puntaje por jornada	
Descripción	El sistema deberá comportarse tal como se describe a continuación, cuando el actor '*Usuario Logueado'* (ACT-0002) seleccione la opción '*Seleccionar jornada'* después de la realización del caso de uso '*Ver clasificación*' (UC-0012).	
Precondición	Ninguna	
Secuencia normal	**Paso**	**Acción**
	1	El sistema mostrará al usuario la clasificación resultante tras la finalización de la jornada en concreto.
Postcondición	Ninguna	
Excepciones	**Paso**	**Acción**
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Frecuencia esperada		
Importancia	Vital	
Urgencia	Inmediata	
Estado	Válido	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla 34: UC-0013 Ver clasificación por jornada

UC-0014	**Ver calendario**	
Versión	1.0	
Autores	Jairo Sánchez García	
Fuentes		
Objetivos asociados	• OBJ-0002 Gestión de las ligas • OBJ-0006 Gestión del calendario	
Casos de uso asociados	• UC-0011 Entrar liga que participa	
Descripción	El sistema deberá comportarse tal como se describe a continuación, cuando el actor '*Usuario Logueado'* (ACT-0002) seleccione la opción '*Calendario'*.	
Precondición	El actor debe estar dentro de una liga.	
Secuencia normal	**Paso**	**Acción**
	1	El sistema mostrará el/los partido/s que hay en la fecha en que se consulta.
	2	Si el usuario selecciona otra fecha, el sistema le mostrará el/los partido/s de la fecha en cuestión.
Postcondición	Ninguna	
Excepciones	**Paso**	**Acción**
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Frecuencia esperada		
Importancia	Vital	
Urgencia	Inmediata	
Estado	Válido	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla 35: UC-0014 Ver calendario

UC-0015	**Ver estadísticas por partido**	
Versión	1.0	
Autores	Jairo Sánchez García	
Fuentes		
Objetivos asociados	• OBJ-0002 Gestión de las ligas • OBJ-0006 Gestión del calendario	
Casos de uso asociados	• UC-0014 Ver calendario	
Descripción	El sistema deberá comportarse tal como se describe a continuación, cuando el actor '*Usuario Logueado'* (ACT-0002) seleccione un partido después de la realización del caso de uso '*Ver calendario'* (UC-0014).	
Precondición	Ninguna	
Secuencia normal	**Paso**	**Acción**
	1	El sistema mostrará al usuario las estadísticas individuales de cada jugador que participe en el partido seleccionado.
Postcondición	Ninguna	
Excepciones	**Paso**	**Acción**
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Frecuencia esperada		
Importancia	Vital	
Urgencia	Inmediata	
Estado	Válido	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla 36: UC-0015 Ver estadísticas por partido

UC-0016	**Ver noticias**	
Versión	1.0	
Autores	Jairo Sánchez García	
Fuentes		
Objetivos asociados	• OBJ-0002 Gestión de las ligas • OBJ-0003 Gestión de las jornadas • OBJ-0005 Gestión del mercado • OBJ-0007 Gestión de las noticias	
Casos de uso asociados		
Descripción	El sistema deberá comportarse tal como se describe a continuación, cuando el actor '*Usuario Logueado'* (ACT-0002) entra a la aplicación.	
Precondición	Ninguna	
Secuencia normal	**Paso**	**Acción**
	1	El sistema deberá mostrar las noticias tanto personales como generales de cada liga de la que forma parte.
Postcondición	Se añade un nuevo usuario a liga.	
Excepciones	**Paso**	**Acción**
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Frecuencia esperada		
Importancia	Vital	
Urgencia	Inmediata	
Estado	Válido	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla 37: UC-0016 Ver noticias

UC-0017	**Calcular puntaje de jugador por jornada**	
Versión	1.0	
Autores	Jairo Sánchez García	
Fuentes		
Objetivos asociados	• OBJ-0002 Gestión de las ligas • OBJ-0003 Gestión de las jornadas	
Casos de uso asociados		
Descripción	El sistema deberá comportarse tal como se describe a continuación, cuando el actor '*Timer'* (ACT-0003) genere un evento diariamente para recopilar lo ocurrido en la jornada y calcular los puntos.	
Precondición	Ninguna	
Secuencia normal	**Paso**	**Acción**
	1	El actor deberá calcular los puntos a partir de unas estadísticas recogidas de los jugadores.
	2	Se almacenarán los puntos junto a las estadísticas en la base de datos.
	3	Se realiza el caso de uso '*Calcular puntaje de usuario por jornada'* (UC-0018).
Postcondición	Se tiene información de una nueva jornada.	
Excepciones	**Paso**	**Acción**
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Frecuencia esperada		
Importancia	Vital	
Urgencia	Inmediata	
Estado	Válido	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla 38: UC-0017 Calcular puntaje de jugador por jornada

UC-0018	**Calcular puntaje de usuario por jornada**	
Versión	1.0	
Autores	Jairo Sánchez García	
Fuentes		
Objetivos asociados	• OBJ-0002 Gestión de las ligas • OBJ-0003 Gestión de las jornadas	
Casos de uso asociados	• UC-0017 Calcular puntaje de jugador por jornada	
Descripción	El sistema deberá comportarse tal como se describe a continuación, cuando acabe la realización del caso de uso '*Calcular puntaje por jornada*' (UC-0017).	
Precondición	Los puntos que ha hecho cada jugador en la jornada se han calculado.	
Secuencia normal	**Paso**	**Acción**
	1	El actor tendrá que calcular los puntos obtenidos por usuario y liga en función de los jugadores que este tenga.
	2	Se actualizará el presupuesto de cada usuario por cada liga en función de estos puntos.
	3	Se realiza el caso de uso '*Actualizar clasificación'* (UC-0019).
Postcondición		
Excepciones	**Paso**	**Acción**
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Frecuencia esperada		
Importancia	Vital	
Urgencia	Inmediata	
Estado	Válido	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla 39: UC-0018 Calcular puntaje de usuario por jornada

UC-0019	**Actualizar clasificación**	
Versión	1.0	
Autores	Jairo Sánchez García	
Fuentes		
Objetivos asociados	• OBJ-0002 Gestión de las ligas • OBJ-0003 Gestión de las jornadas	
Casos de uso asociados	• UC-0017 Calcular puntaje de jugador por jornada • UC-0018 Calcular puntaje de usuario por jornada	
Descripción	El sistema deberá comportarse tal como se describe a continuación, cuando acabe la realización del caso de uso '*Calcular puntaje de usuario por jornada*' (UC-0018).	
Precondición	Los puntos que ha hecho cada usuario en la jornada se han calculado.	
Secuencia normal	**Paso**	**Acción**
	1	El actor tendrá que actualizar la clasificación general de cada liga en función de los puntos obtenidos.
Postcondición	Ninguna	
Excepciones	**Paso**	**Acción**
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Frecuencia esperada		
Importancia	Vital	
Urgencia	Inmediata	
Estado	Válido	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla 40: UC-0019 Actualizar clasificación

4.2.3.3 Gestión de Mercado

Los casos de uso que conforman al paquete Gestión de Mercado, apreciables en la *Ilustración 5*, permiten realizar las tareas necesarias para garantizar una buena gestión de las funcionalidades asociadas al mercado.

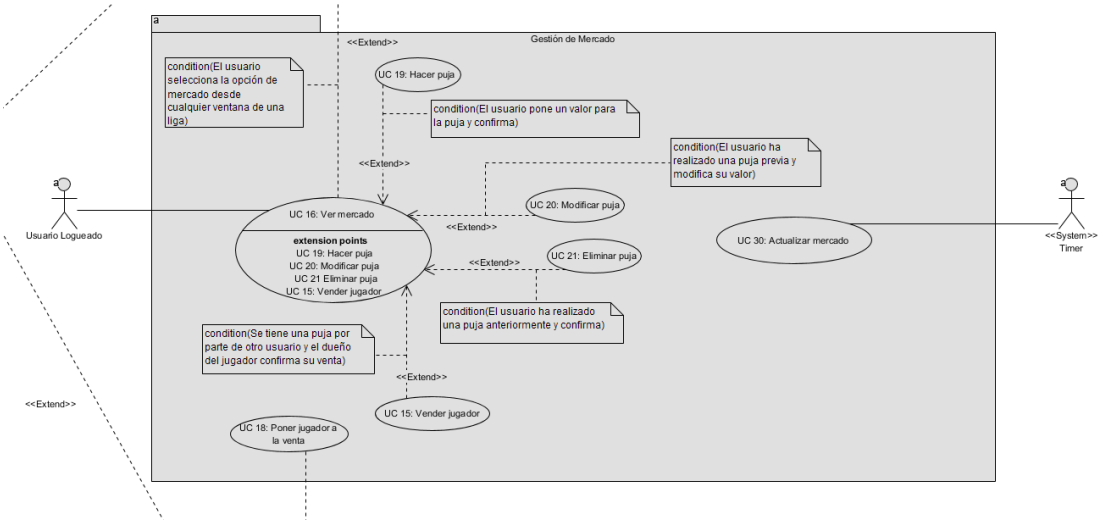


Ilustración 5: Diagrama de casos de uso, Gestión de Mercado

UC-0020	**Vender jugador**	
Versión	1.0	
Autores	Jairo Sánchez García	
Fuentes		
Objetivos asociados	• OBJ-0002 Gestión de las ligas • OBJ-0004 Gestión del roster • OBJ-0005 Gestión del mercado	
Casos de uso asociados	• UC-0021 Ver mercado	
Descripción	El sistema deberá comportarse tal como se describe a continuación, cuando el actor '*Usuario Logueado'* (ACT-0001) seleccione la acción '*Vender',* después de realizar el caso de uso '*Ver mercado'* (UC-0021).	
Precondición	El jugador para vender se ha tenido que poner en el mercado y tiene una puja por parte de otro usuario.	
Secuencia normal	**Paso**	**Acción**
	1	El sistema le indica al actor si quiere confirmar la venta o cancelar.
	2	Si el actor confirma, vende el jugador y se añade el dinero de la venta a su presupuesto.

Postcondición	El roster de otro usuario se ha actualizado con un nuevo jugador y su presupuesto ha bajado.	
Excepciones	**Paso**	**Acción**
	2	Si el actor cancela, el caso de uso queda sin efecto alguno.
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Frecuencia esperada		
Importancia	Vital	
Urgencia	Inmediata	
Estado	Válido	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla 41: UC-0020 Vender jugador

UC-0021	**Ver mercado**	
Versión	1.0	
Autores	Jairo Sánchez García	
Fuentes		
Objetivos asociados	• OBJ-0002 Gestión de las ligas • OBJ-0005 Gestión del mercado	
Casos de uso asociados	• UC-0011 Entrar liga que participa	
Descripción	El sistema deberá comportarse tal como se describe a continuación, cuando el actor '*Usuario Logueado'* (ACT-0001) seleccione la acción '*Mercado'*.	
Precondición	El actor tiene que estar dentro de una liga.	
Secuencia normal	**Paso**	**Acción**
	1	El sistema mostrará al usuario los jugadores con la información necesaria (nombre, posición, valor,…).
Postcondición	Ninguna	
Excepciones	**Paso**	**Acción**
Rendimiento	**Paso**	**Tiempo máximo**

	-	-
Frecuencia esperada		
Importancia	Vital	
Urgencia	Inmediata	
Estado	Válido	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla 42: UC-0021 Ver mercado

UC-0022	**Poner jugador a la venta**	
Versión	1.0	
Autores	Jairo Sánchez García	
Fuentes		
Objetivos asociados	• OBJ-0002 Gestión de las ligas • OBJ-0005 Gestión del mercado	
Casos de uso asociados	• UC-0028 Ver equipo	
Descripción	El sistema deberá comportarse tal como se describe a continuación, cuando el actor '*Usuario Logueado'* (ACT-0001) seleccione la acción '*Poner a la venta'* después de realizar el caso de uso '*Ver equipo*' (UC-0028).	
Precondición	Ninguna	
Secuencia normal	**Paso**	**Acción**
	1	El actor seleccionará el jugador que quiere poner a la venta, es decir, mandarlo al mercado.
	2	El sistema añadirá este jugador al mercado.
Postcondición	El mercado se actualiza con un nuevo jugador.	
Excepciones	**Paso**	**Acción**
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Frecuencia esperada		
Importancia	Vital	

Urgencia	Inmediata
Estado	Válido
Estabilidad	Alta
Comentarios	Ninguno

Tabla 43: UC-0022 Poner jugador a la venta

UC-0023	**Hacer puja**	
Versión	1.0	
Autores	Jairo Sánchez García	
Fuentes		
Objetivos asociados	• OBJ-0002 Gestión de las ligas • OBJ-0005 Gestión del mercado	
Casos de uso asociados	• UC-0021 Ver mercado	
Descripción	El sistema deberá comportarse tal como se describe a continuación, cuando el actor '*Usuario Logueado'* (ACT-0001) seleccione la acción '*Pujar,* después de realizar el caso de uso '*Ver mercado'* (UC-0021).	
Precondición	El actor ha debido de determinar un valor para la puja.	
Secuencia normal	**Paso**	**Acción**
	1	El sistema mostrará al usuario si confirma o cancela la puja.
	2	Si el actor confirma la puja, seguidamente el sistema la hace efectiva y devuelve un mensaje indicándolo.
Postcondición	Se sobrescribe el presupuesto del usuario.	
Excepciones	**Paso**	**Acción**
	2	Si el actor cancela la puja, el caso de uso queda sin efecto.
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Frecuencia esperada		
Importancia	Vital	
Urgencia	Inmediata	
Estado	Válido	

Estabilidad	Alta
Comentarios	Ninguno

Tabla 44: UC-0023 Hacer puja

UC-0024	**Modificar puja**	
Versión	1.0	
Autores	Jairo Sánchez García	
Fuentes		
Objetivos asociados	• OBJ-0002 Gestión de las ligas • OBJ-0005 Gestión del mercado	
Casos de uso asociados	• UC-0021 Ver mercado • UC-0023 Hacer puja	
Descripción	El sistema deberá comportarse tal como se describe a continuación, cuando el actor '*Usuario Logueado'* (ACT-0001) seleccione la acción '*Editar',* después de realizar el caso de uso '*Ver mercado'* (UC-0021).	
Precondición	El actor ha tenido que realizar una puja previa por el jugador en cuestión.	
Secuencia normal	**Paso**	**Acción**
	1	El sistema mostrará al usuario varias maneras de modificar el valor de la puja, bien sea incrementando, decrementando o introduciendo el valor a mano.
	2	El actor modifica el valor de alguna de las posibles maneras.
	3	Se realiza el caso de uso '*Hacer puja'* (UC-0023).
Postcondición	Se sobrescribe la puja anterior y el presupuesto del usuario.	
Excepciones	**Paso**	**Acción**
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Frecuencia esperada		
Importancia	Vital	
Urgencia	Inmediata	
Estado	Válido	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla 45: UC-0024 Modificar puja

UC-0025	**Eliminar puja**	
Versión	1.0	
Autores	Jairo Sánchez García	
Fuentes		
Objetivos asociados	• OBJ-0002 Gestión de las ligas • OBJ-0005 Gestión del mercado	
Casos de uso asociados	• UC-0021 Ver mercado • UC-0023 Hacer puja	
Descripción	El sistema deberá comportarse tal como se describe a continuación, cuando el actor '*Usuario Logueado'* (ACT-0001) seleccione la acción '*Eliminar',* después de realizar el caso de uso '*Ver mercado'* (UC-0021).	
Precondición	El actor ha tenido que realizar una puja previa por el jugador en cuestión.	
Secuencia normal	**Paso**	**Acción**
	1	El sistema mostrará al usuario si desea cancelar o confirmar.
	2	Si el actor confirma la eliminación, el sistema elimina la puja por ese jugador y devuelve un mensaje haciéndoselo saber.
Postcondición	Se sobrescribe el presupuesto del usuario.	
Excepciones	**Paso**	**Acción**
	2	Si el cancela la eliminación, el caso de uso queda sin efecto.
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Frecuencia esperada		
Importancia	Vital	
Urgencia	Inmediata	
Estado	Válido	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla 46: UC-0025 Eliminar puja

UC-0026	**Actualizar mercado**	
Versión	1.0	
Autores	Jairo Sánchez García	
Fuentes		
Objetivos asociados	• OBJ-0002 Gestión de las ligas • OBJ-0005 Gestión del mercado	
Casos de uso asociados	• UC-0030 Actualizar roster	
Descripción	El sistema deberá comportarse tal como se describe a continuación, cuando el caso de uso '*Actualizar roster'* (UC-0030) finalice.	
Precondición	Ninguna	
Secuencia normal	**Paso**	**Acción**
	1	Se tendrá que eliminar los jugadores del mercado y añadir unos nuevos por cada liga existente.
Postcondición	Se obtiene un mercado con 10 jugadores nuevos.	
Excepciones	**Paso**	**Acción**
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Frecuencia esperada		
Importancia	Vital	
Urgencia	Inmediata	
Estado	Válido	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla 47: UC-0026 Actualizar mercado

4.2.3.4 Gestión de Roster

Por último, se tiene el paquete Gestión de Roster compuesto por los casos de uso que se pueden observar en la *Ilustración 6*, en donde están definidas las acciones a realizar para el manejo de los rosters que los usuarios posean.

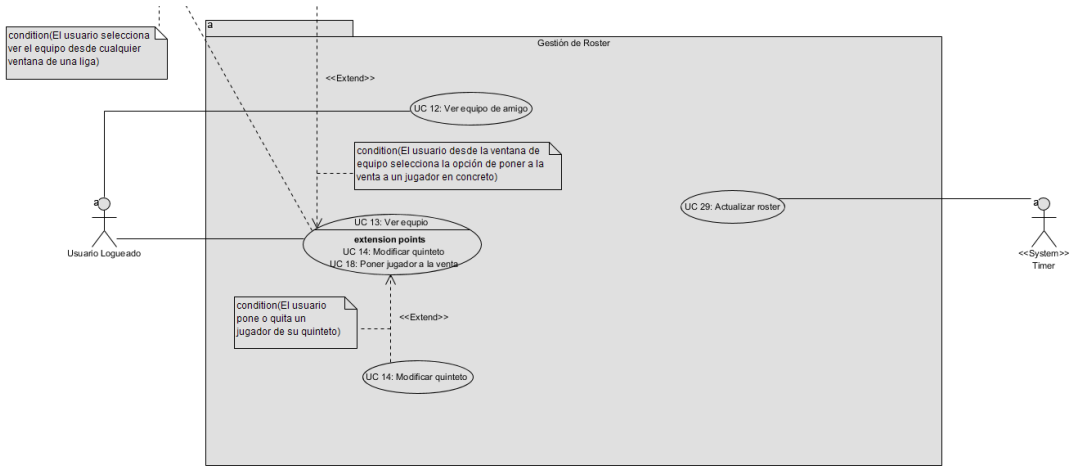


Ilustración 6: Diagrama de casos de uso, Gestión de Roster

UC-0027	**Ver equipo de amigo**	
Versión	1.0	
Autores	Jairo Sánchez García	
Fuentes		
Objetivos asociados	• OBJ-0002 Gestión de las ligas • OBJ-0004 Gestión del roster	
Casos de uso asociados	• UC-0011 Entrar a liga que participa • UC-0012 Ver clasificación	
Descripción	El sistema deberá comportarse tal como se describe a continuación, cuando el actor '*Usuario Logueado*' (ACT-0002), después de la realización bien del caso de uso '*Entrar a liga que participa*' (UC-0011) o bien '*Ver clasificación*' (UC-0012), pulsa sobre el usuario del que quiere ver el equipo.	
Precondición		
Secuencia normal	**Paso**	**Acción**
	1	El sistema le mostrará al usuario el equipo del usuario que haya seleccionado.
Postcondición	Ninguna	
Excepciones	**Paso**	**Acción**

Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Frecuencia esperada		
Importancia	Vital	
Urgencia	Inmediata	
Estado	Válido	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla 48: UC-0027 Ver equipo de amigo

UC-0028	**Ver equipo**	
Versión	1.0	
Autores	Jairo Sánchez García	
Fuentes		
Objetivos asociados	• OBJ-0002 Gestión de las ligas • OBJ-0004 Gestión del roster	
Casos de uso asociados	• UC-0011 Entrar a liga que participa	
Descripción	El sistema deberá comportarse tal como se describe a continuación, cuando el actor '*Usuario Logueado'* (ACT-0002), seleccione la opción 'Roster' dentro de una liga.	
Precondición	El usuario ha tenido que acceder a una liga.	
Secuencia normal	**Paso**	**Acción**
	1	El sistema mostrará los jugadores que conforman el roster del usuario, así como los starters, si este los tuviera. Además de las distintas opciones que se pueden llevar a cabo (añadir y quitar del quinteto y poner a la venta).
Postcondición	Ninguna	
Excepciones	**Paso**	**Acción**
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Frecuencia esperada		

Importancia	Vital
Urgencia	Inmediata
Estado	Válido
Estabilidad	Alta
Comentarios	Ninguno

Tabla 49: UC-0028 Ver equipo

UC-0029	**Modificar quinteto**	
Versión	1.0	
Autores	Jairo Sánchez García	
Fuentes		
Objetivos asociados	• OBJ-0002 Gestión de las ligas • OBJ-0004 Gestión del roster	
Casos de uso asociados	• UC-0014 Ver equipo	
Descripción	El sistema deberá comportarse tal como se describe a continuación, cuando el actor '*Usuario Logueado*' (ACT-0002), después de la realización del caso de uso '*Ver equipo*' (UC-0028), selecciona alguna de las opciones para modificar los starters.	
Precondición	Ninguna	
Secuencia normal	**Paso**	**Acción**
	1	El actor realiza la acción de añadir o quitar un jugador.
	2	El sistema modifica sobre el quinteto poniendo o quitando al jugador en cuestión.
	3	El actor realiza '*Guardar*'.
	4	El sistema guarda el nuevo quinteto.
Postcondición	El usuario tiene un quinteto nuevo.	
Excepciones	**Paso**	**Acción**
	3	Si el actor guarda los datos , el caso de uso quedará sin efecto y por lo tanto los cambios no se llevarán a cabo.
Rendimiento	**Paso**	**Tiempo máximo**
	-	-

Frecuencia esperada	
Importancia	Vital
Urgencia	Inmediata
Estado	Válido
Estabilidad	Alta
Comentarios	Ninguno

Tabla 50: UC-0029 Modificar quinteto

UC-0030	**Actualizar roster**	
Versión	1.0	
Autores	Jairo Sánchez García	
Fuentes		
Objetivos asociados	• OBJ-0002 Gestión de las ligas • OBJ-0004 Gestión del roster	
Casos de uso asociados		
Descripción	El sistema deberá comportarse tal como se describe a continuación, cuando un temporizador diariamente guarda los cambios en los equipos de los usuarios que han podido ocurrir en ese periodo.	
Precondición	Ninguna	
Secuencia normal	**Paso**	**Acción**
	1	Se ven las pujas realizadas sobre un jugador, y se determinar un ganador.
	2	Se le añade ese jugador al equipo del ganador de la puja.
	3	Si el jugador pertenece a un usuario y no tiene pujas, el sistema lo comprará y se quitará de su equipo
	2	El actor incrementará o disminuirá el presupuesto de los
Postcondición	Ninguna	
Excepciones	**Paso**	**Acción**
Rendimiento	**Paso**	**Tiempo máximo**
	-	-
Frecuencia esperada		

Importancia	Vital
Urgencia	Inmediata
Estado	Válido
Estabilidad	Alta
Comentarios	Ninguno

Tabla 51: UC-0030 Actualizar roster

4.3 Requisitos no funcionales

En esta parte se va a desarrollar los requisitos no funcionales, estos van a imponer al sistema unas restricciones en el diseño y en la implementación. El cumplimiento de estos le da al sistema un cierto grado de calidad, así mismo, el no cumplimiento de los requisitos no funcionales puede provocar que no se satisfagan las necesidades necesarias.

NFR-0001	**Portabilidad**
Versión	1.0
Autores	Jairo Sánchez García
Fuentes	
Objetivos asociados	Ninguno
Requisitos asociados	Ninguno
Descripción	El sistema debe funcionar correctamente tanto desde la web como desde Android e IOS.
Importancia	Vital
Urgencia	Inmediata
Estado	Válido
Estabilidad	Alta
Comentarios	Ninguno

Tabla 52: NFR-0001 Portabilidad

NFR-0002	Usabilidad
Versión	1.0
Autores	Jairo Sánchez García
Fuentes	
Objetivos asociados	Ninguno
Requisitos asociados	Ninguno
Descripción	El sistema debe ser manejable tanto por usuarios expertos como no expertos.
Importancia	Vital
Urgencia	Inmediata
Estado	Válido
Estabilidad	Alta
Comentarios	Ninguno

Tabla 53: NFR-0002 Usabilidad

NFR-0003	Privacidad
Versión	1.0
Autores	Jairo Sánchez García
Fuentes	
Objetivos asociados	Ninguno
Requisitos asociados	Ninguno
Descripción	El sistema debe garantizar la privacidad de los datos personales sensibles pertenecientes a los usuarios, además de las operaciones independientes a cada uno.
Importancia	Vital
Urgencia	Inmediata
Estado	Válido

Estabilidad	Alta
Comentarios	Ninguno

Tabla 54: NFR-0003 Privacidad

NFR-0004	**Almacenamiento**
Versión	1.0
Autores	Jairo Sánchez García
Fuentes	
Objetivos asociados	Ninguno
Requisitos asociados	Ninguno
Descripción	El sistema debe guardar los datos en sistemas externos al mismo para la optimización del espacio y garantizar una consistencia en los datos global al sistema.
Importancia	Vital
Urgencia	Inmediata
Estado	Válido
Estabilidad	Alta
Comentarios	Ninguno

Tabla 55: NFR-0004 Almacenamiento

5. MATRIZ DE RASTREABILIDAD

La matriz de rastreabilidad ayuda a la monitorización de los requisitos a lo largo del desarrollo del proyecto, donde se vincula a cada requisito con su objetivo asociado. En la tabla posterior se observa la relación que existe entre ellos en este sistema.

TRM-0001	OBJ-0001	OBJ-0002	OBJ-0003	OBJ-0004	OBJ-0005	OBJ-0006	OBJ-0007	IRQ-0001	IRQ-0002	IRQ-0003	IRQ-0004	IRQ-0005	IRQ-0006	IRQ-0007
UC-0001	X	-	-	-	-	-	-	X	-	-	-	-	-	-
UC-0002	X	-	-	-	-	-	-	X	-	-	-	-	-	-
UC-0003	X	-	-	-	-	-	-	-	-	-	-	-	-	-
UC-0004	X	-	-	-	-	-	-	X	-	-	-	-	-	-
UC-0005	X	-	-	-	-	-	-	X	-	-	-	-	-	-
UC-0006	X	-	-	-	-	-	-	X	-	-	-	-	-	-
UC-0007	X	-	-	-	-	-	-	X	-	-	-	-	-	-
UC-0008	-	X	-	-	-	-	-	-	X	X	X	-	-	-
UC-0009	-	X	-	-	-	-	-	-	X	X	X	-	-	-
UC-0010	-	X	-	-	-	-	-	-	-	X	-	-	-	-
UC-0011	-	X	-	-	-	-	-	-	-	X	-	-	-	-
UC-0012	-	X	X	-	-	-	-	-	X	-	X	-	-	-
UC-0013	-	X	X	-	-	-	-	-	-	X	-	X	-	-
UC-0014	-	X	-	-	-	X	-	-	-	-	-	-	-	-
UC-0015	-	X	-	-	-	X	-	-	-	-	-	-	-	-
UC-0016	-	X	X	-	X	-	X	-	X	X	-	X	X	X
UC-0017	-	X	X	-	-	-	-	-	X	-	X	X	-	-
UC-0018	-	X	X	-	-	-	-	-	-	-	-	X	-	-
UC-0019	-	X	X	-	-	-	-	-	-	X	-	X	-	-
UC-0020	-	X	-	X	X	-	-	-	-	-	X	-	-	X
UC-0021	-	X	-	-	X	-	-	-	-	-	-	-	-	X
UC-0022	-	X	-	-	X	-	-	-	-	-	X	-	-	X
UC-0023	-	X	-	-	X	-	-	-	-	-	-	-	-	X

UC-0024	-	X	-	-	X	-	-	-	-	-	-	-	-	X
UC-0025	-	X	-	-	X	-	-	-	-	-	-	-	-	X
UC-0026	-	X	-	-	X	-	-	-	-	-	-	-	-	X
UC-0027	-	X	-	X	-	-	-	-	-	-	X	-	-	-
UC-0028	-	X	-	X	-	-	-	-	-	-	X	-	-	-
UC-0029	-	X	-	X	-	-	-	-	-	-	X	-	-	-
UC-0030	-	X	-	X	-	-	-	-	-	-	-	-	-	-

Tabla 56: Matriz de trazabilidad

6. BIBLIOGRAFÍA

¿Qué es un requisito no funcional? Tipos y ejemplos. (2021). Obtenido de https://ebooksonline.es/que-es-un-requisito-no-funcional-tipos-y-ejemplos/

Durán Toro, A., & Bernárdez Jiménez, B. (2000). *Metodología para la Elicitación de Requisitos de Sistemas Software, versión 2.1.* Sevilla.

García Peñalvo, F. J., & García Holgado, A. *Fundamentos de la vista de casos de uso.*

García Peñalvo, F. J., & Moreno García, M. N. *Trasparencias de Ingeniería del Software, Tema 5.*

ANEXO III: Análisis de requisitos

Dream NBA: aplicación para la creación de ligas virtuales y gestión de un equipo propio

Trabajo de Fin de Grado

INGENIERÍA INFORMÁTICA

VNiVERSiDAD D SALAMANCA

CAMPUS DE EXCELENCIA INTERNACIONAL

Julio 2022

Autor

Jairo Sánchez García

Tutores

Gabriel Villarrubia González

Juan Francisco de Paz Santana

Héctor Sánchez San Blas

TABLA DE CONTENIDO

ÍNDICES DE ILUSTRACIONES

1. INTRODUCCIÓN

El objetivo de este anexo es ahondar en la manera que se van a comportar los casos de uso especificados en el *Anexo II: Especificación de requisitos*. Para ello, se va a detallar el modelo del dominio a partir de un diagrama de clases, las relaciones y comunicación entre las clases de análisis y la agrupación de estas en una vista de arquitectura. El anexo incluye los siguientes apartados:

- **Modelo del dominio**: se especifican los objetos reales que participan en el sistema así como las relaciones entre ellos. Se desarrolla el diagrama de clases y la ilustración de cada una.
- **Realización de casos de uso**: se muestra el comportamiento de los casos de uso y el intercambio de mensajes existentes entre objetos.
- **Clases de análisis**: Se detalla las relaciones y comunicaciones entre las clases de análisis agrupadas en paquetes.
- **Vista de arquitectura del modelo de análisis**: se agrupan las clases de análisis en una vista de arquitectura, MVC.

2. MODELO DE DOMINIO

Uno de los aspectos claves de la fase de análisis de requisitos es realizar una correcta conceptualización de los objetos que participan en el sistema de manera física y relaciones que existen entre sí. A partir de los requisitos de información especificados en el *Anexo II: Especificación de requisitos software*, se va a determinar las entidades del modelo de negocio.

Para representar el modelo del dominio se realiza el siguiente diagrama de clases:

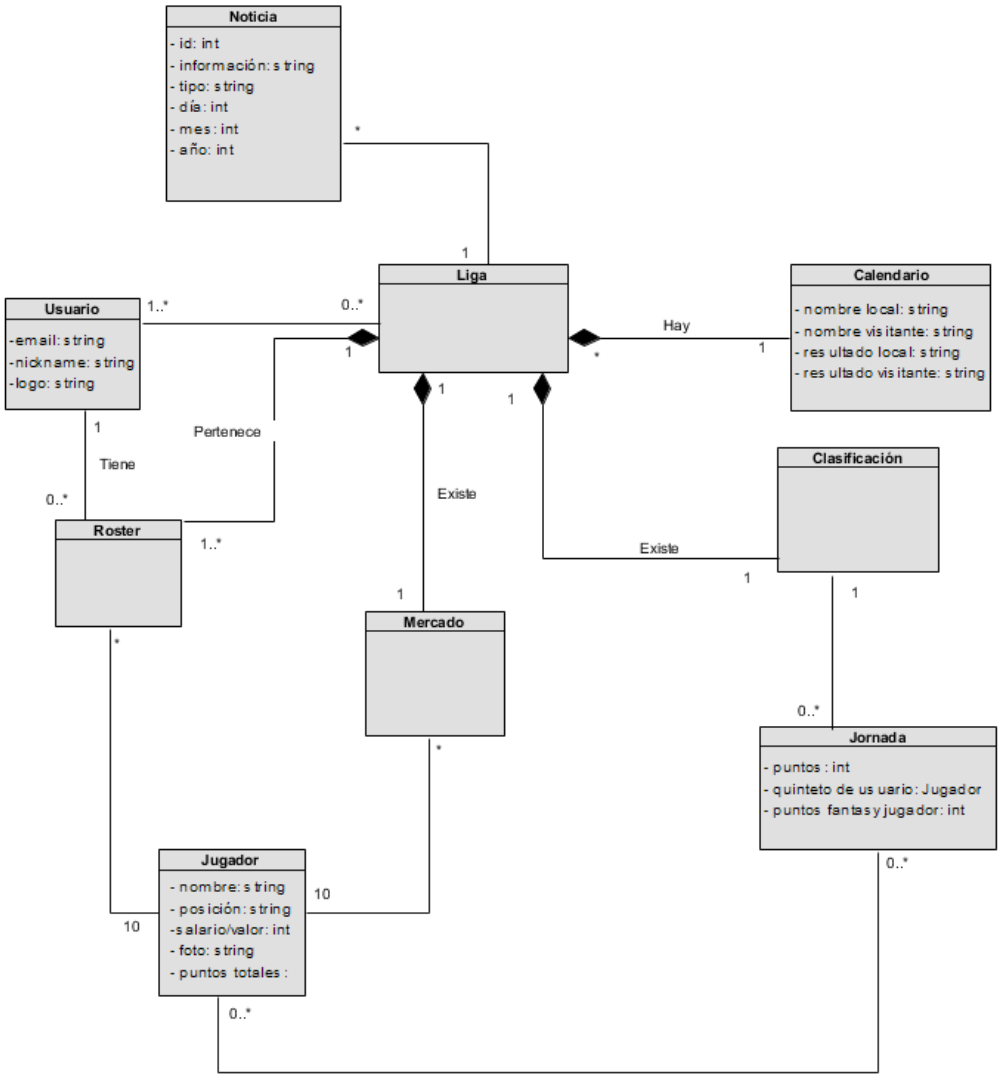


Ilustración 1: Diagrama de clases

A continuación se van a explicar las clases que aparecen en la *Ilustración 1*:

- **Usuario**: representa a los usuarios que tienen una cuenta en el sistema. Dicho usuario puede poseer desde 0 a n rosters del mismo modo en el que puede estar en 0 o en n ligas. Esta clase tiene los atributos email, nickname y logo, de tipo string todos.
- **Roster**: constituye el grupo de 10 jugadores que da forma a un roster. En donde el roster pertenece únicamente a un usuario y está sólo en una liga.
- **Jugador**: simboliza a los jugadores de la vida real. De ellos se guarda el nombre, posición, salario/valor, foto y puntos totales; los tipos para estos atributos son de tipo int para puntos totales y salario/valor, para el resto de tipo string. En referencia a las relaciones,

se tiene que un jugador participa desde 0 a n jornadas, está en varios mercados y en varios rosters.

- **Mercado**: contiene una lista de 10 jugadores que dan forma a un mercado individual para cada liga.
- **Liga**: esta clase está compuesta por distintas clases como Mercado, Calendario, Clasificación y Roster. En donde en la liga participan desde 1 a muchos usuarios, debido a que uno es el número mínimo para que exista.
- **Calendario**: representa al calendario de partidos reales de la NBA, el cual pertenece a varias ligas. Guarda los nombres y resultados de los equipos locales y visitantes como información, estos atributos son de tipo string.
- **Clasificación**: simboliza el ranking de una liga. Cada clasificación está ligada a una liga en concreto, y está formada desde 0 hasta varias jornadas.
- **Jornada**: constituye la información que se genera tras la realización de una jornada, donde se guardan los puntos y quintetos de los usuarios, puntos Fantasy calculados. Una jornada forma parte de una clasificación y en ella participan desde ningún jugador hasta muchos.
- **Noticia**: esta clase representa a las noticias que se dan lugar en el sistema. Almacena la información necesaria para su utilización y comprensión como un identificador, la información del mensaje de la noticia, el tipo, identificador de la liga, día, mes y año. Los tipos para los atributos son de tipo entero excepto para la información del mensaje y el tipo que serán string.

3. REALIZACIÓN DE CASOS DE USO-ANÁLISIS

Este apartado va a explicar el comportamiento dinámico de los casos de uso, definidos en el *Anexo II*, y la secuencia de mensajes intercambiado por los objetos de los diferentes paquetes que conforman el sistema mediante la utilización de diagramas de secuencia.

3.1 Diagramas de secuencia del Paquete Gestión de Usuarios

Caso de uso 0001 **Crear cuenta**

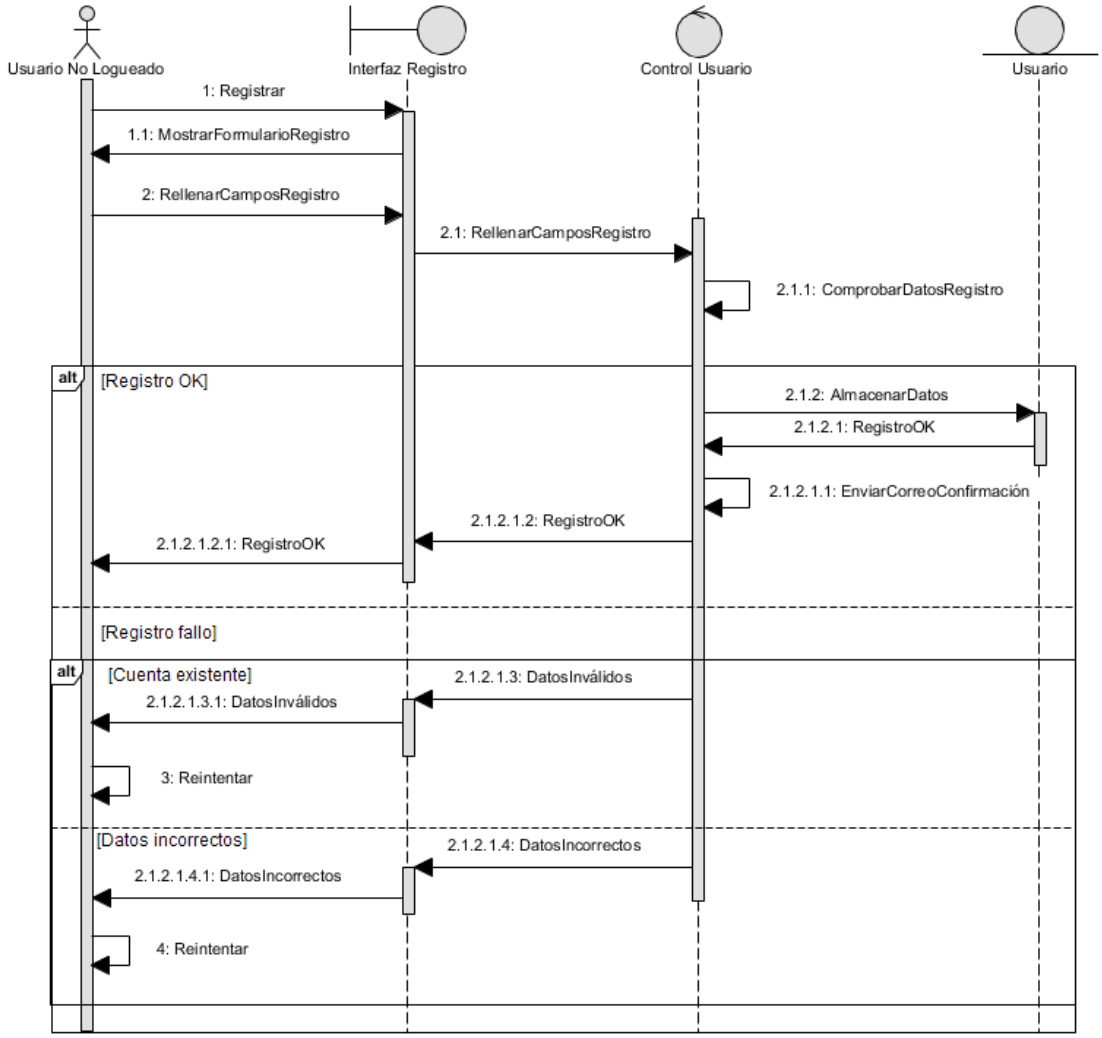


Ilustración 2: UC-0001 Crear cuenta

Caso de uso 0002 **Iniciar sesión**

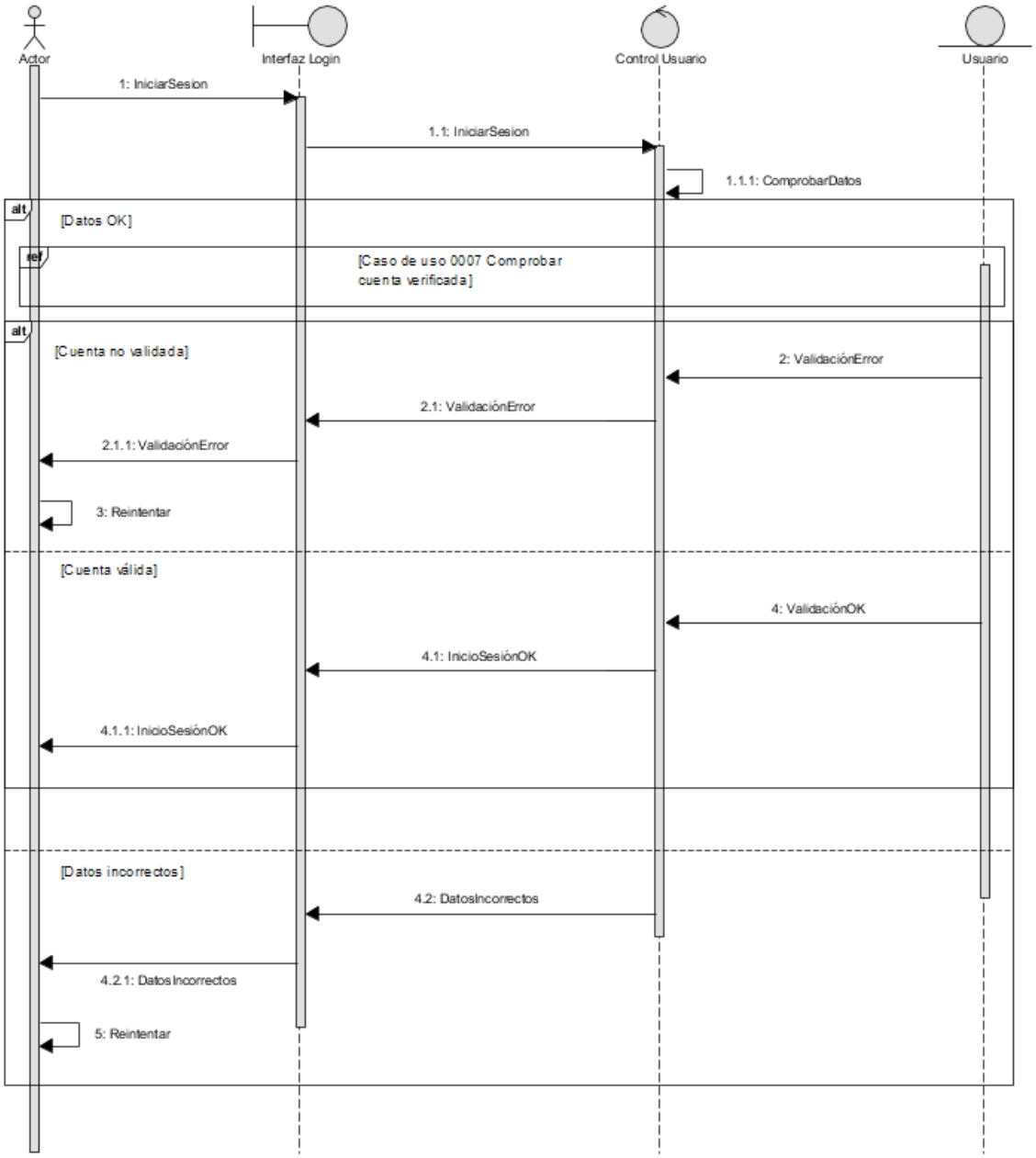


Ilustración 3: UC-0002 Iniciar sesión

Caso de uso 0003 **Salir**

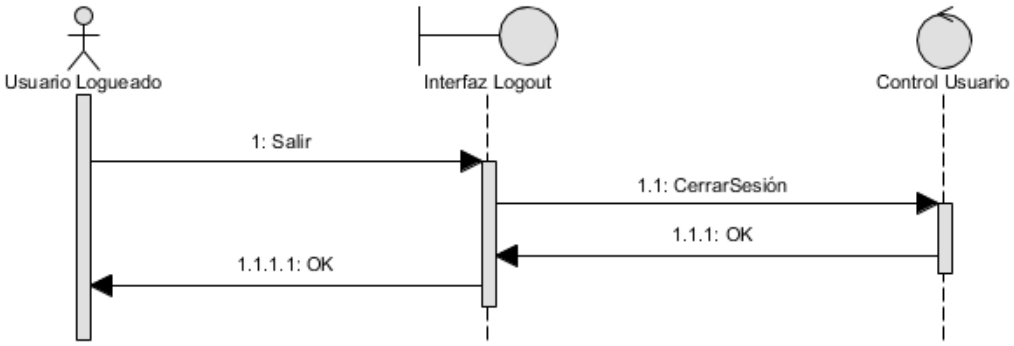


Ilustración 4: UC-0003 Salir

Caso de uso 0004 **Recuperar contraseña**

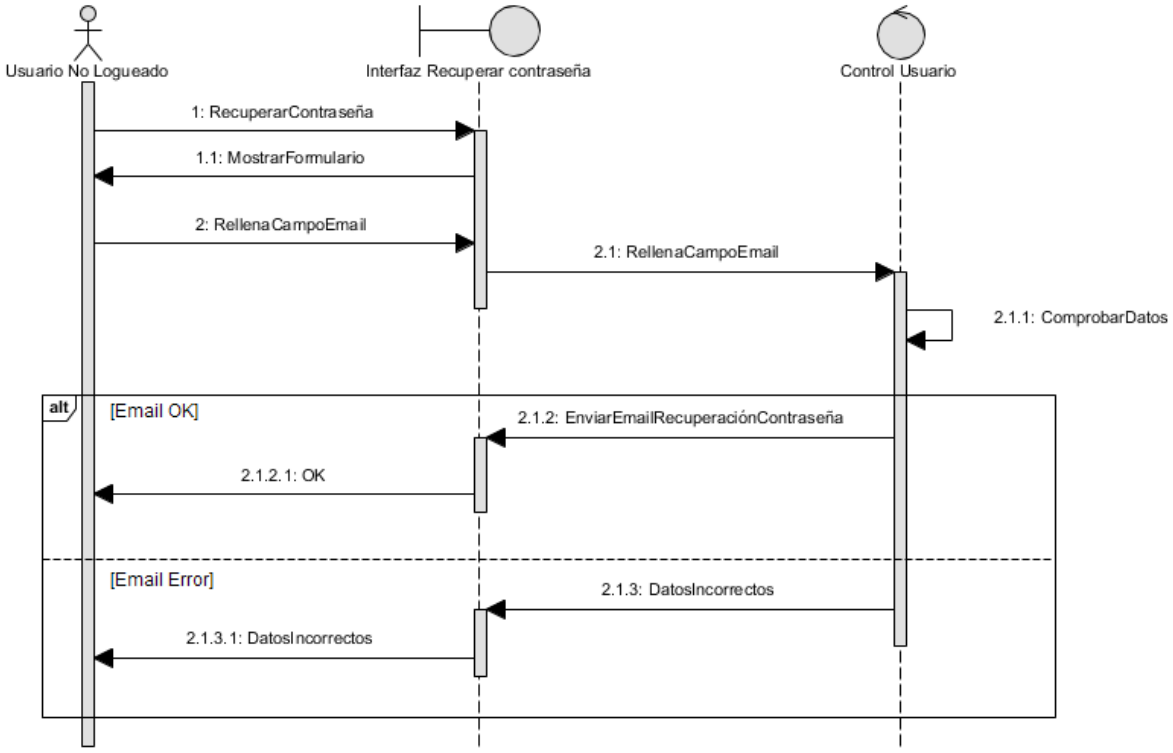


Ilustración 5: UC 0004 Recuperar contraseña

Caso de uso 0005 **Ver perfil**

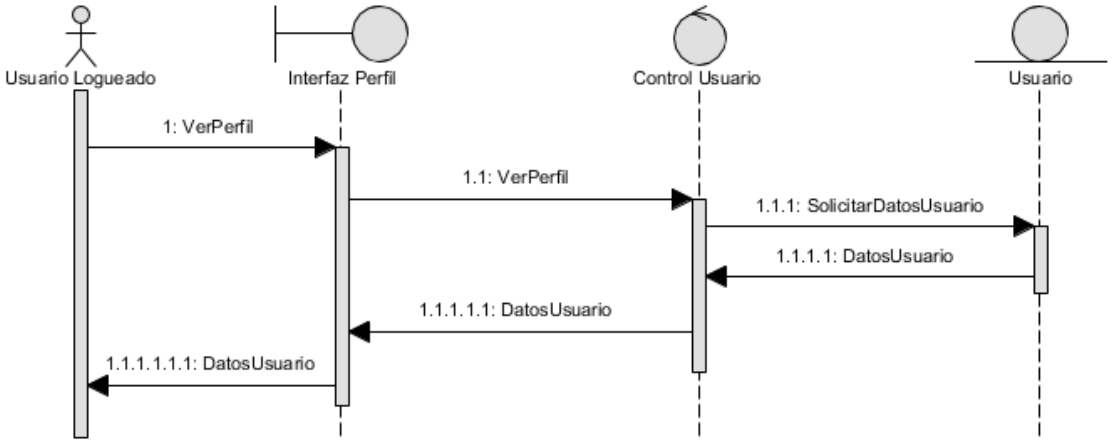


Ilustración 6: UC-0005 Ver perfil

Caso de uso 0006 **Modificar datos cuenta**

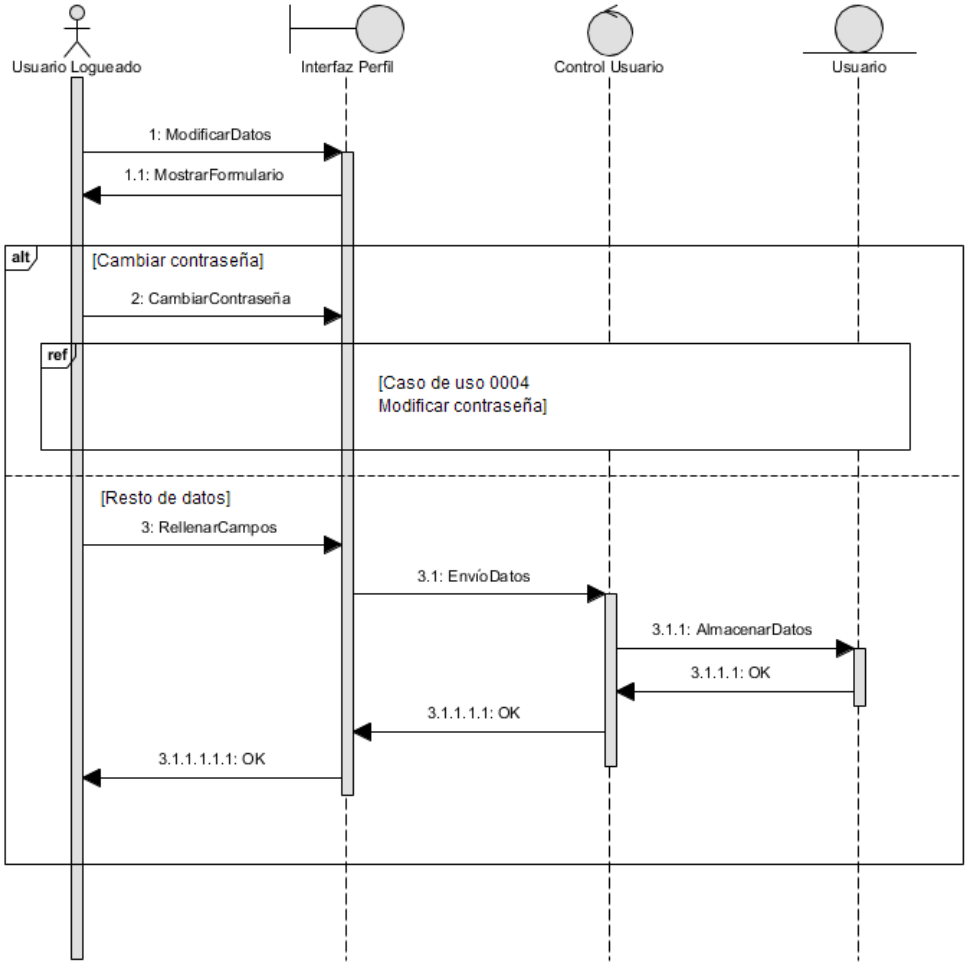


Ilustración 7: UC-0006 Modificar datos cuenta

Caso de uso 0007 **Comprobar cuenta verificada**

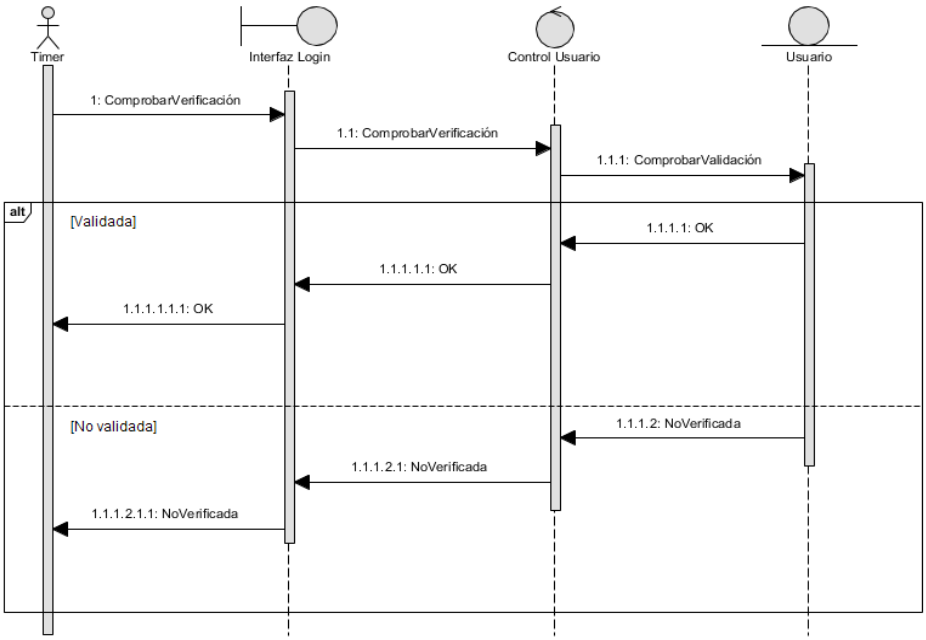


Ilustración 8: UC-0007 Comprobar cuenta verificada

3.2 Diagramas de secuencia del Paquete Gestión de Ligas

Caso de uso 0008 **Crear liga**

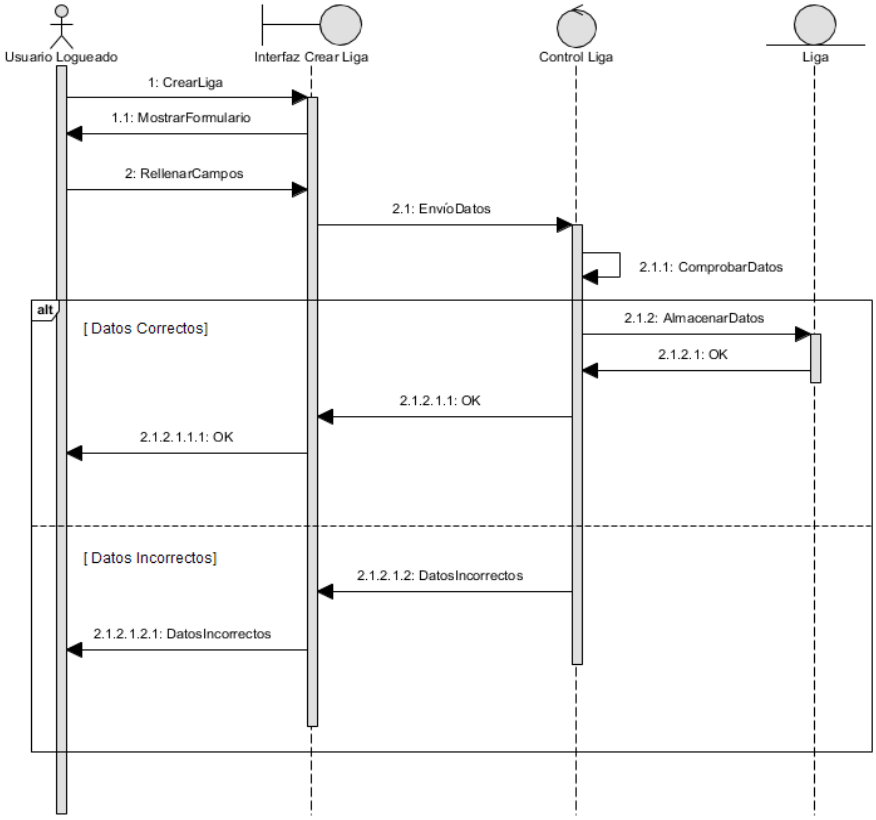


Ilustración 9: UC-0008 Crear liga

Caso de uso 0009 **Unirse a liga a través de código**

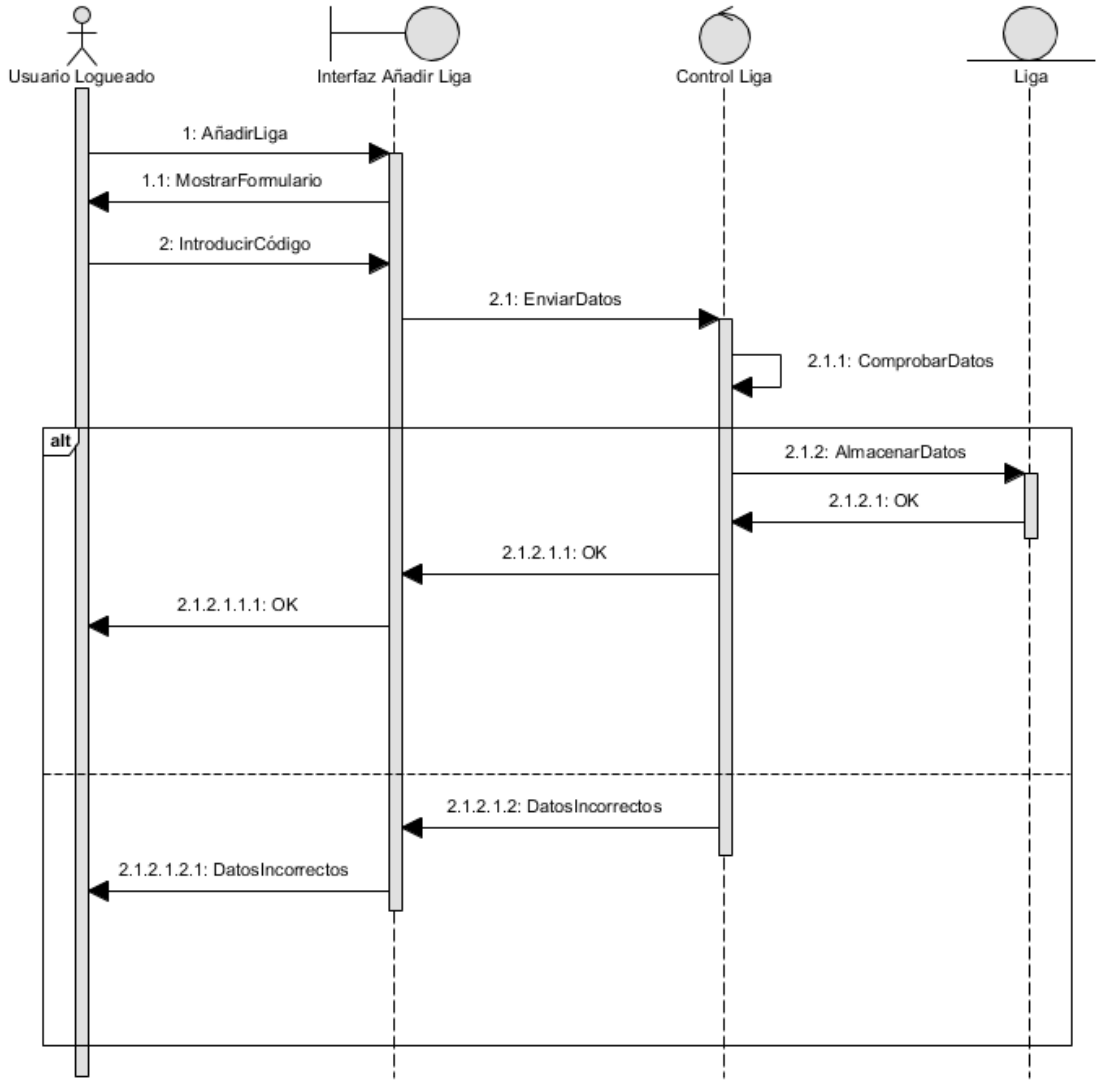


Ilustración 10: UC-0009 Unirse a liga a través de código

Caso de uso 0010 **Ver ligas que participa**

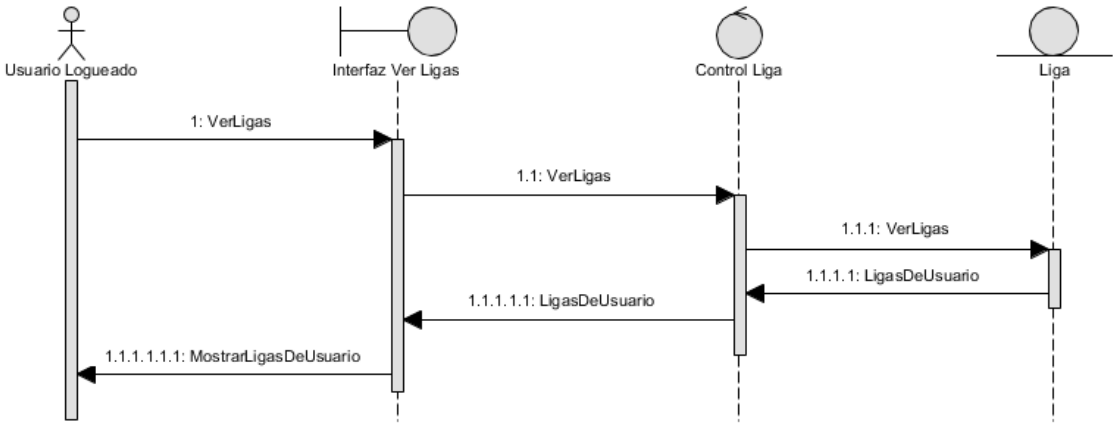


Ilustración 11: UC-0010 Ver ligas que participa

Caso de uso 0011 **Entrar a liga que participa**

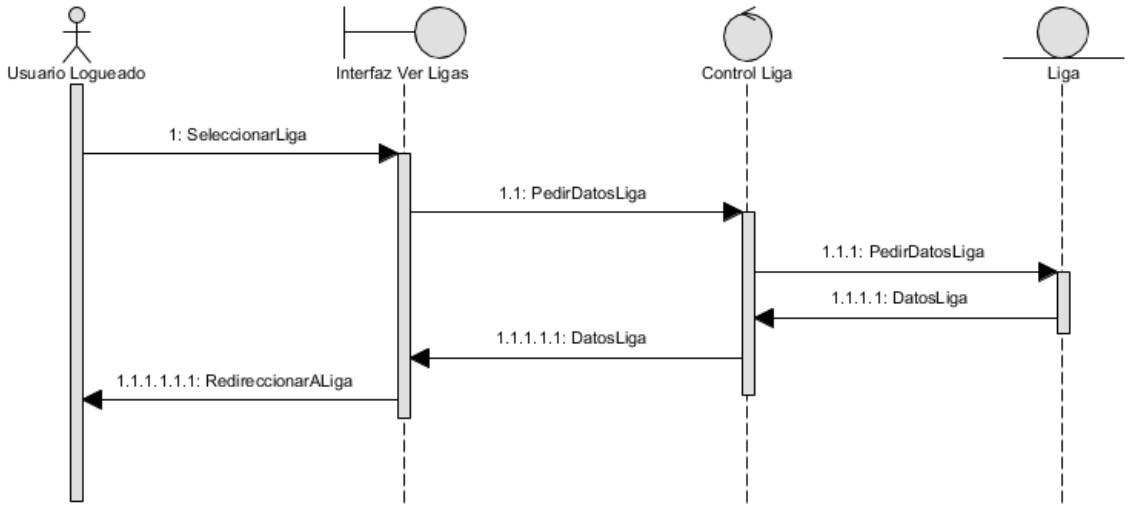


Ilustración 12: UC-0011 Entrar a liga que participa

Caso de uso 0012 **Ver clasificación**

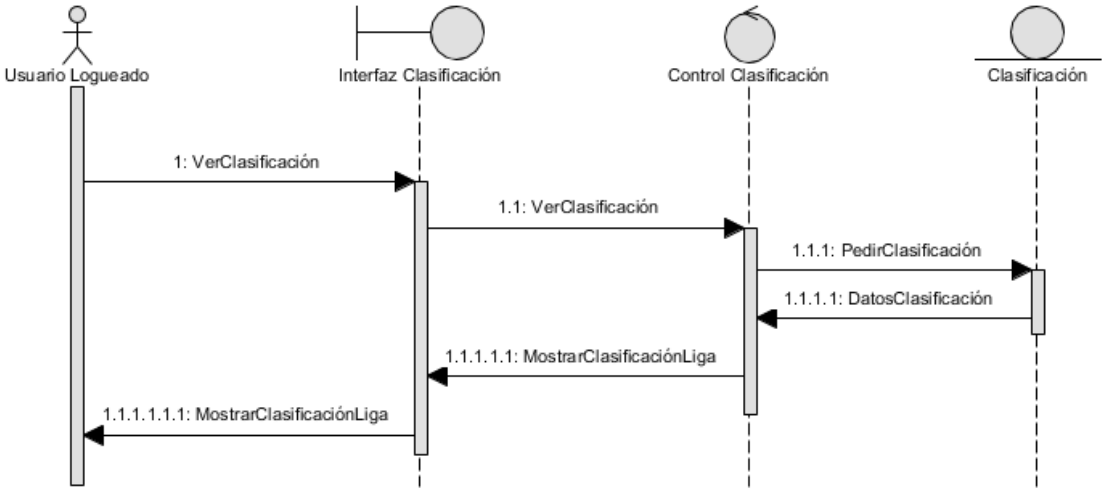


Ilustración 13: UC-0012 Ver clasificación

Caso de uso 0013 **Ver clasificación por jornada**

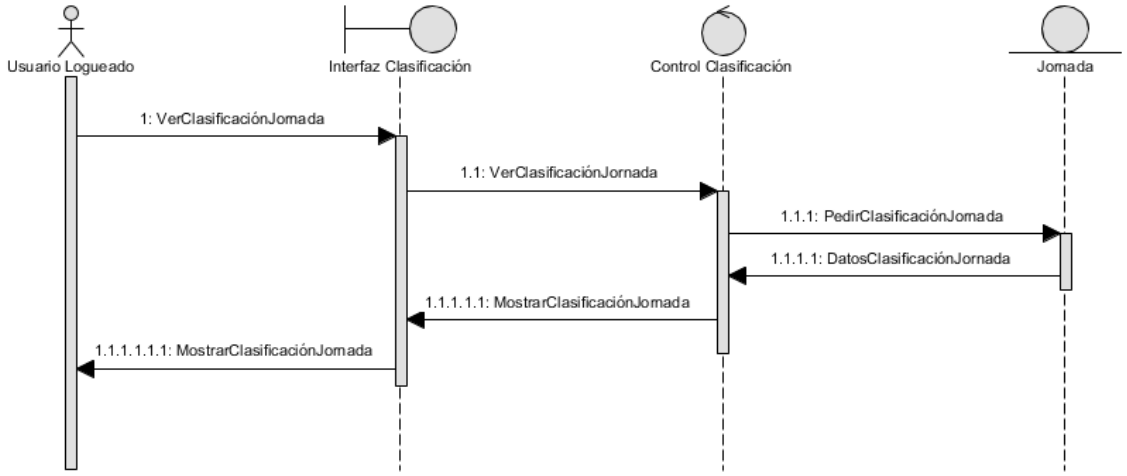


Ilustración 14: UC-0013 Ver clasificación por jornada

Caso de uso 0014 **Ver calendario**

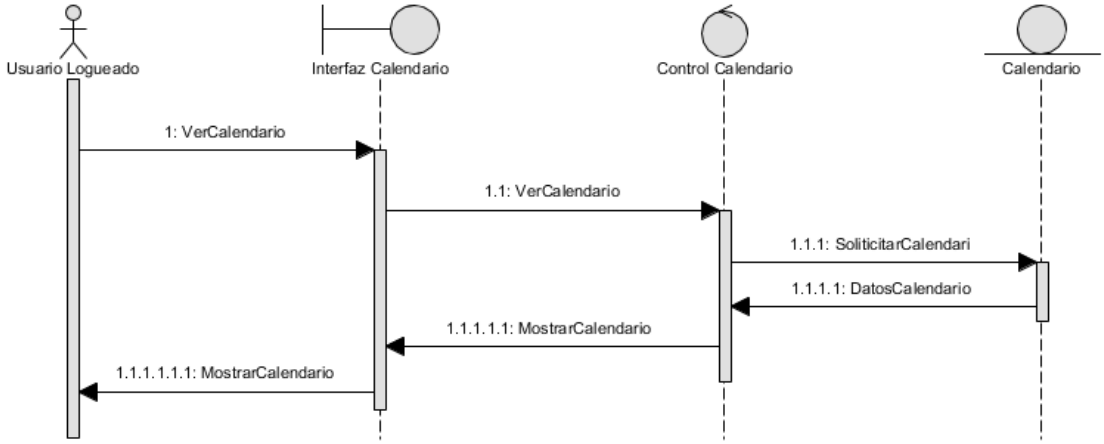


Ilustración 15: UC-0014 Ver calendario

Caso de uso 0015 **Ver estadísticas por partido**

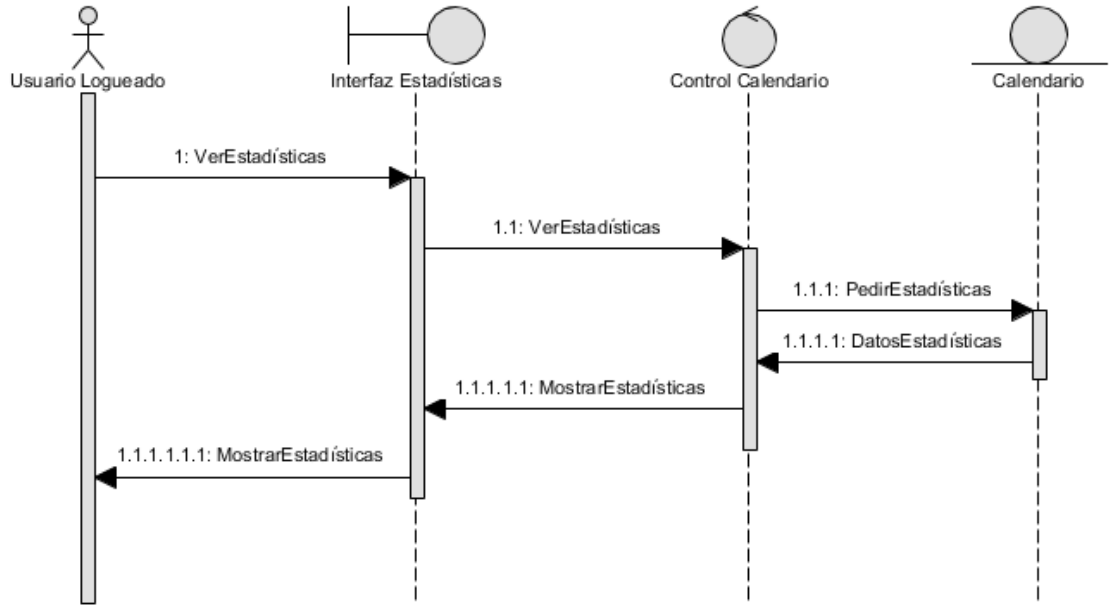


Ilustración 16: UC-0015 Ver estadísticas por partido

Caso de uso 0016 **Ver noticias**

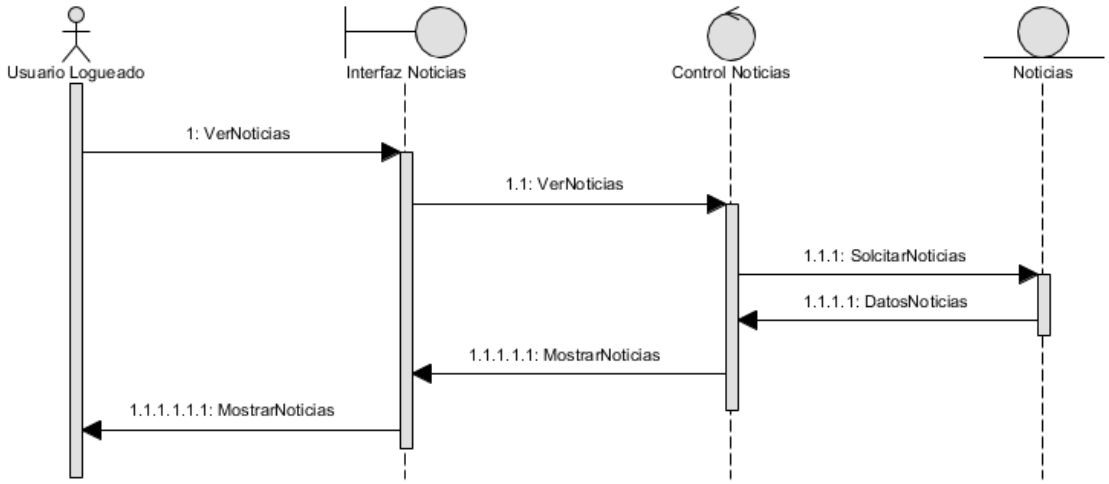


Ilustración 17: UC-0016 Ver noticias

Caso de uso 0017 **Calcular puntaje de jugador por jornada**

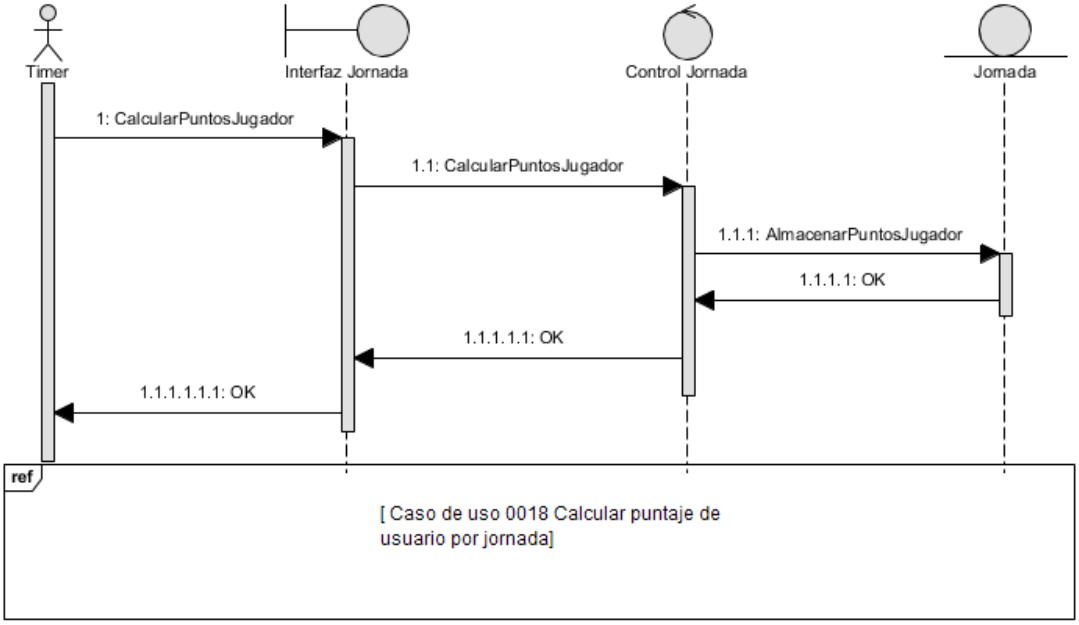


Ilustración 18: UC-0017 Calcular puntaje de jugador por jornada

Caso de uso 0018 **Calcular puntaje de usuario por jornada**

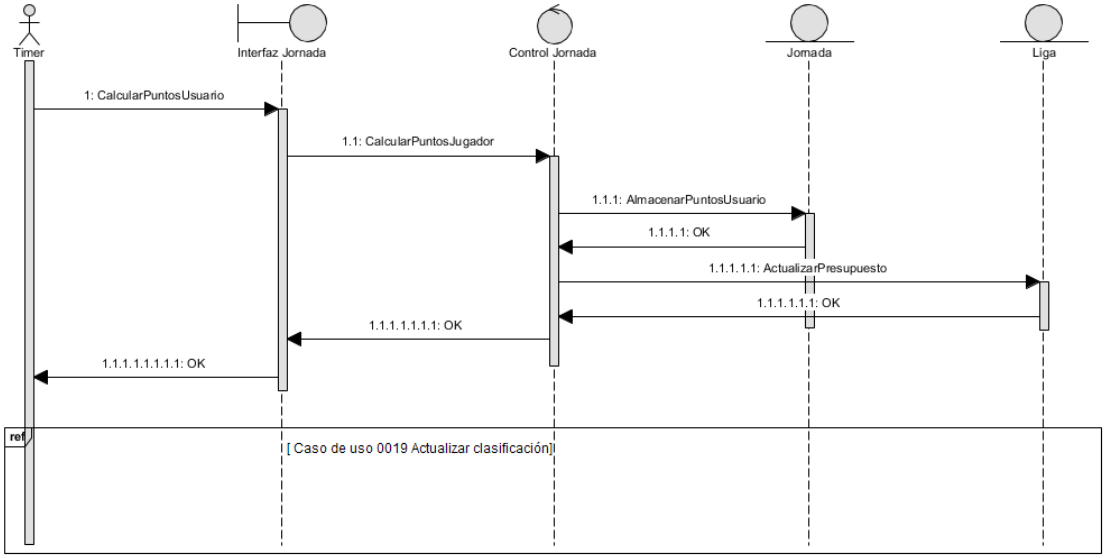


Ilustración 19: UC-0018 Calcular puntaje de usuario por jornada

Caso de uso 0019 **Actualizar clasificación**

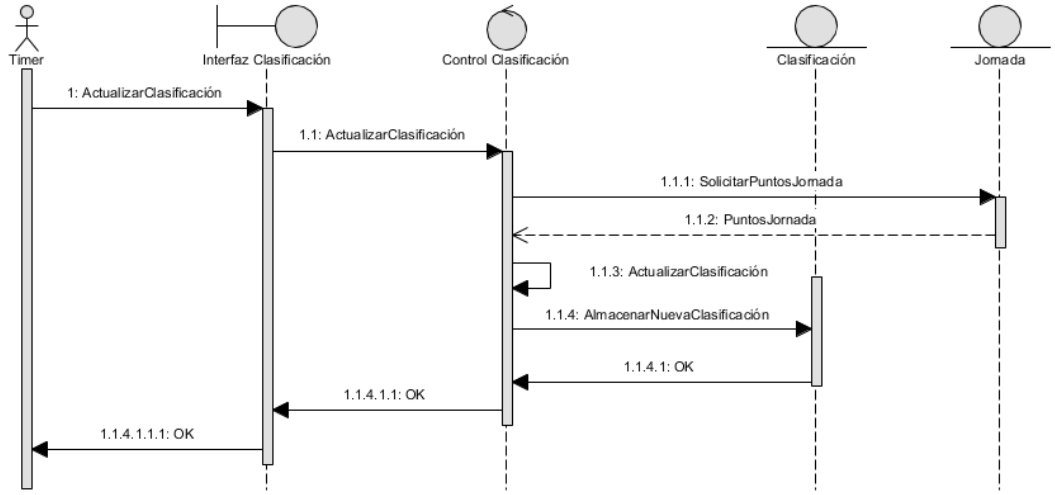


Ilustración 20: UC-0019 Actualizar clasificación

3.2 Diagramas de secuencia del Paquete Gestión de Mercado

Caso de uso 0020 **Vender jugador**

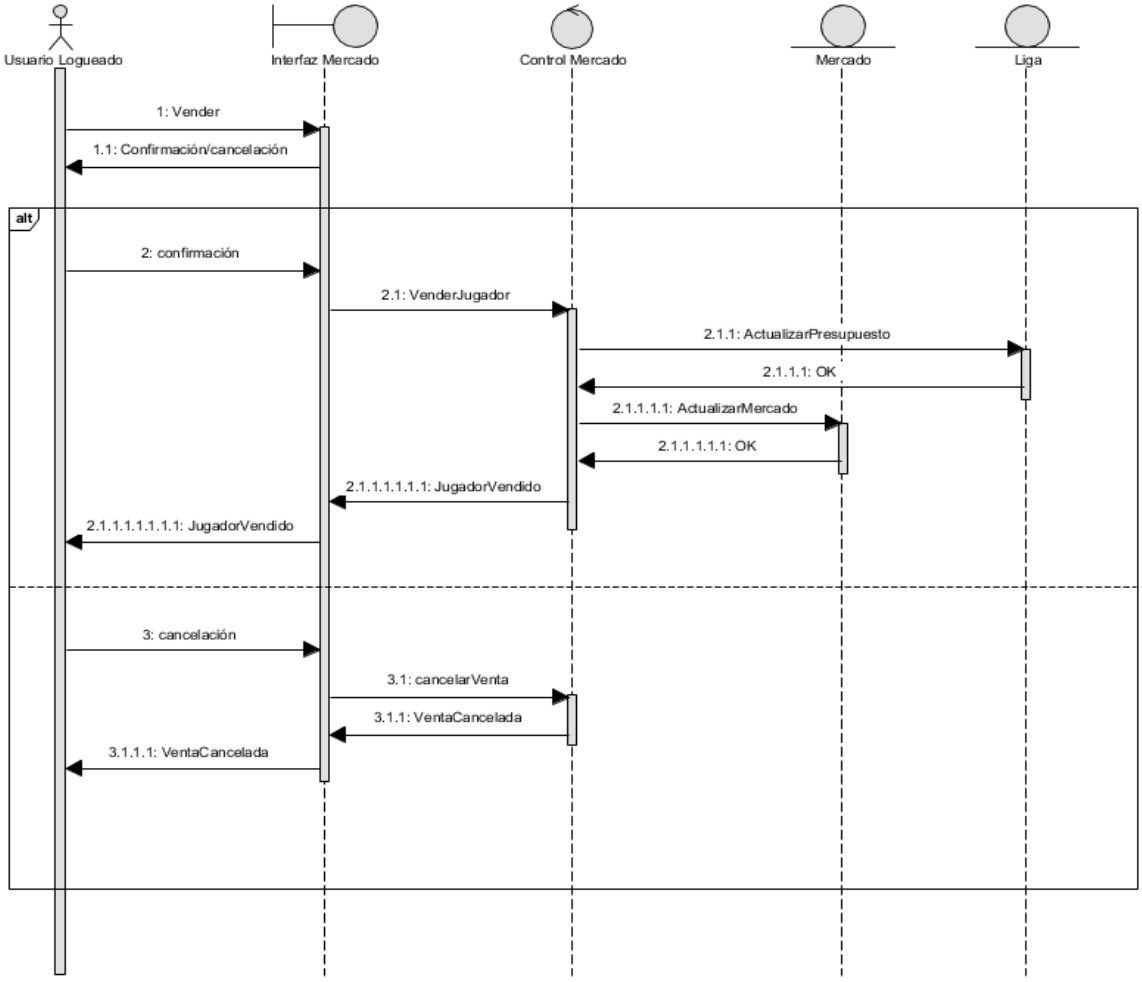


Ilustración 21: UC-0020 Vender jugador

Caso de uso 0021 **Ver mercado**

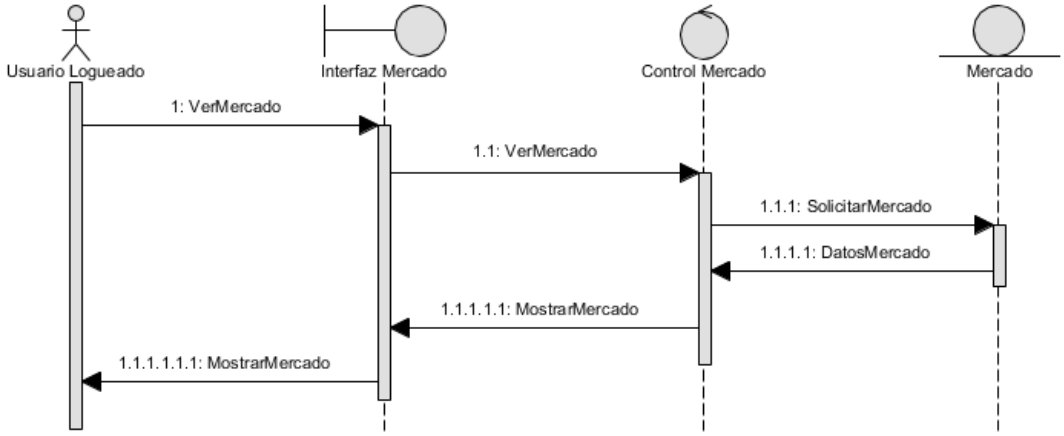


Ilustración 22: UC-0021 Ver mercado

Caso de uso 0022 **Poner jugador a la venta**

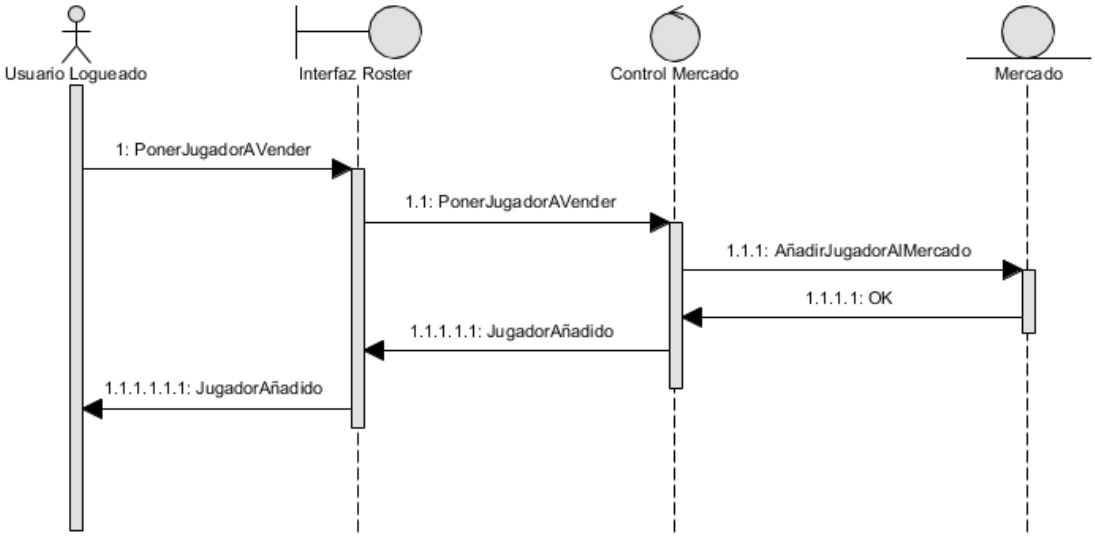


Ilustración 23: UC-0022 Poner jugador a la venta

Caso de uso 0023 **Hacer puja**

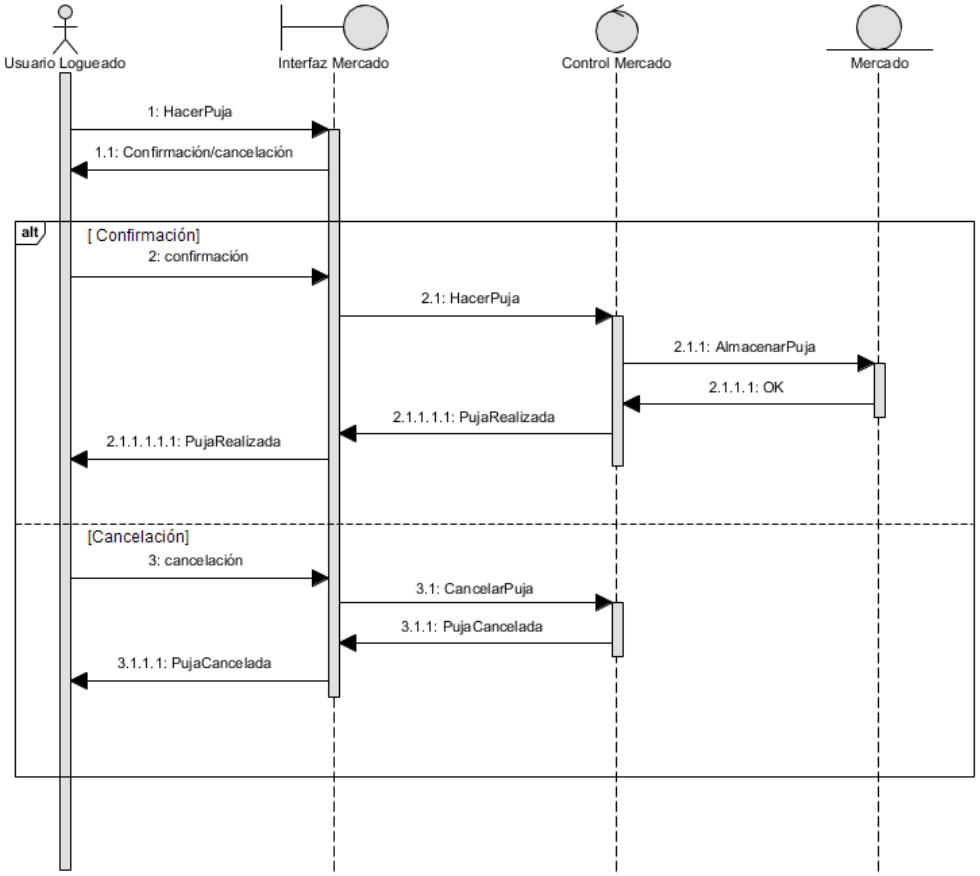


Ilustración 24: UC-0023 Hacer puja

Caso de uso 0024 **Modificar puja**

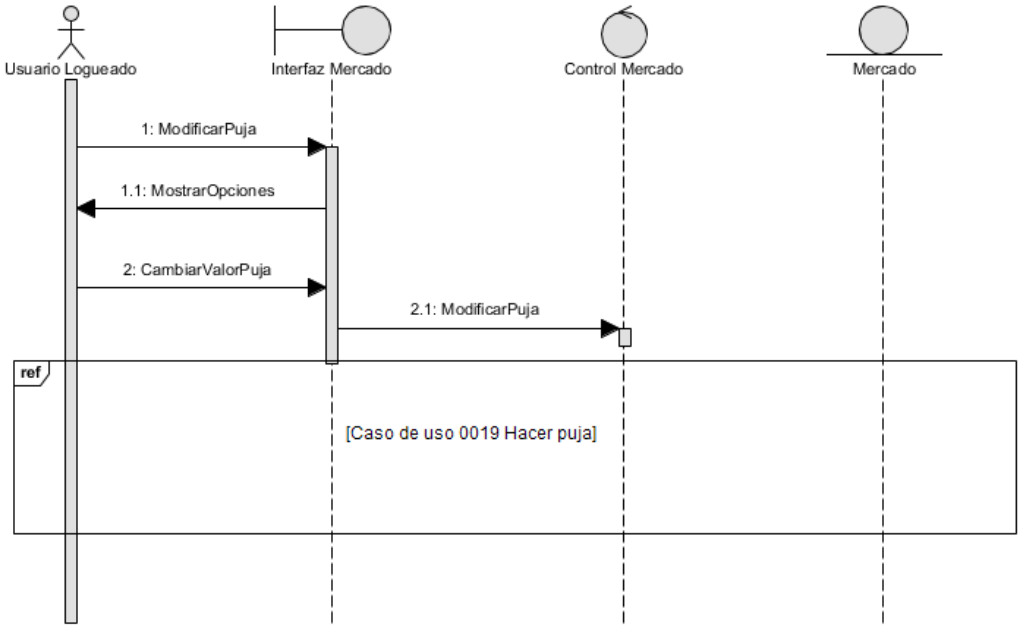


Ilustración 25: UC-0024 Modificar puja

Caso de uso 0025 **Eliminar puja**

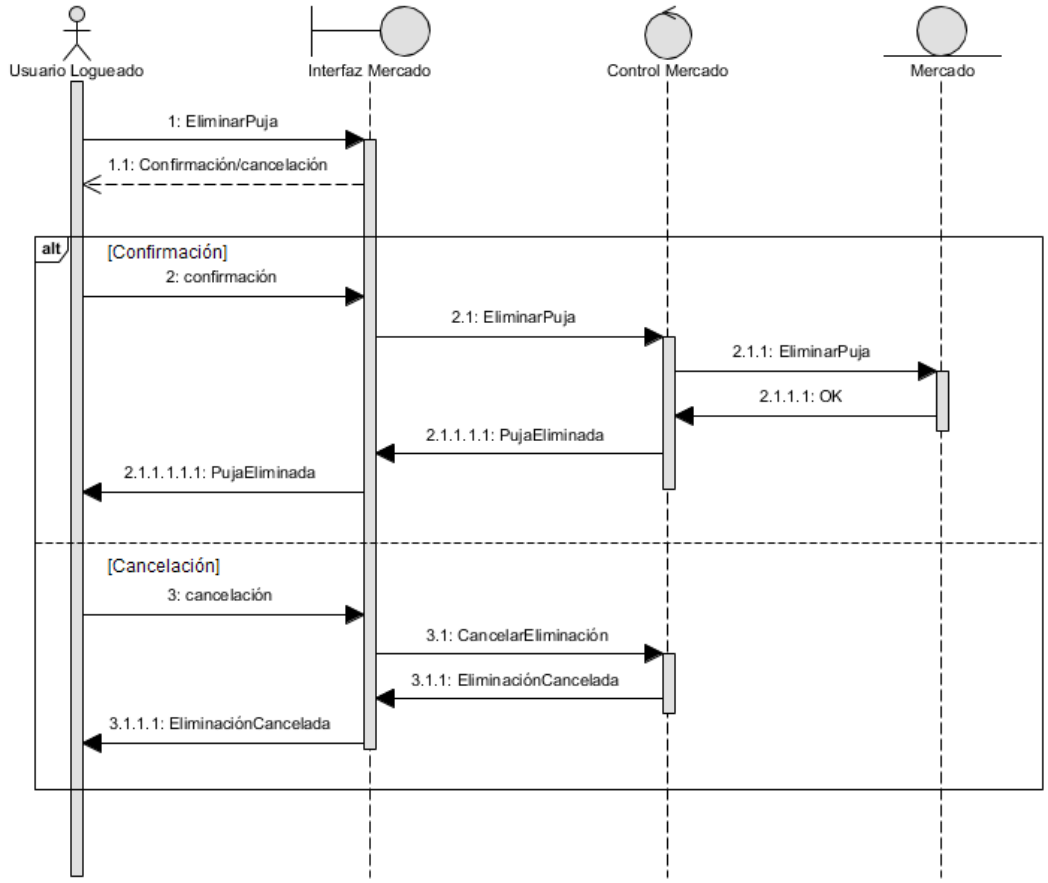


Ilustración 26: UC-0025 Eliminar puja

Caso de uso 0026 **Actualizar mercado**

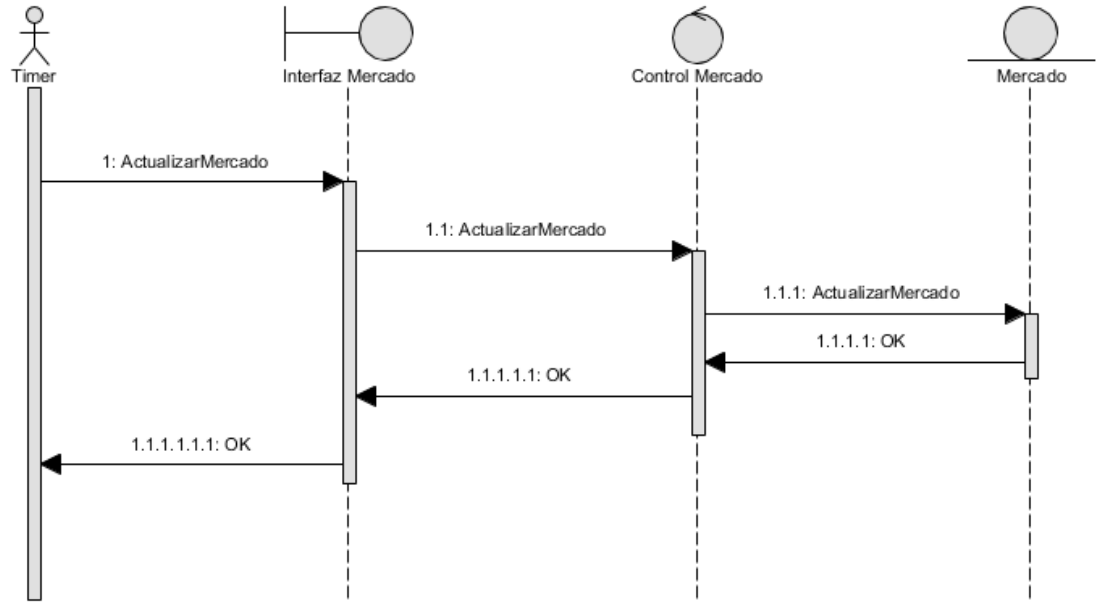


Ilustración 27: UC-0026 Actualizar mercado

3.2 Diagramas de secuencia del Paquete Gestión de Roster

Caso de uso 0027 **Ver equipo de amigo**

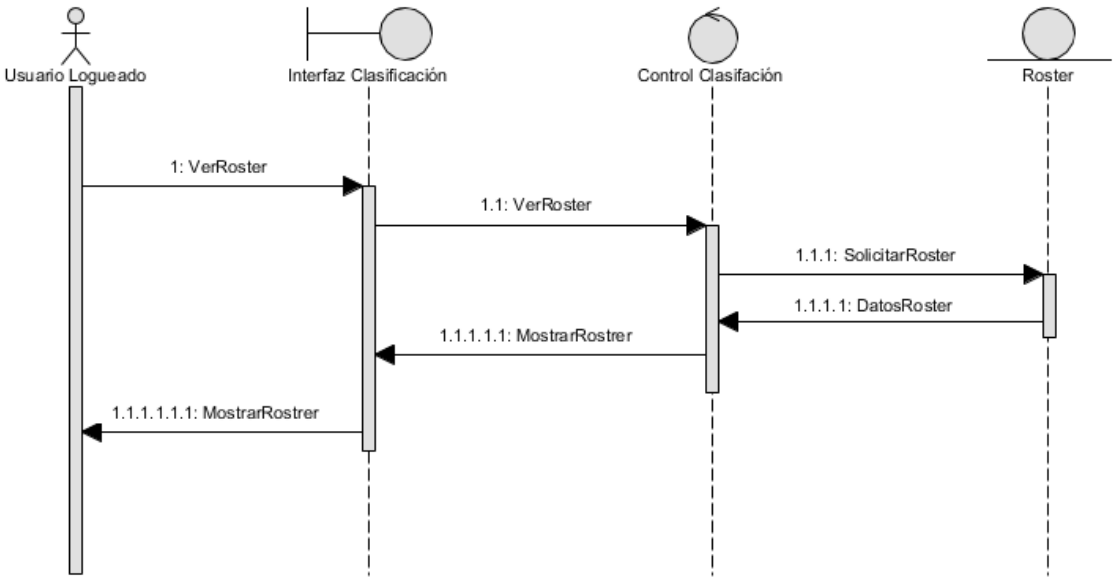


Ilustración 28: UC-0027 Ver equipo de amigo

Caso de uso 0028 **Ver equipo**

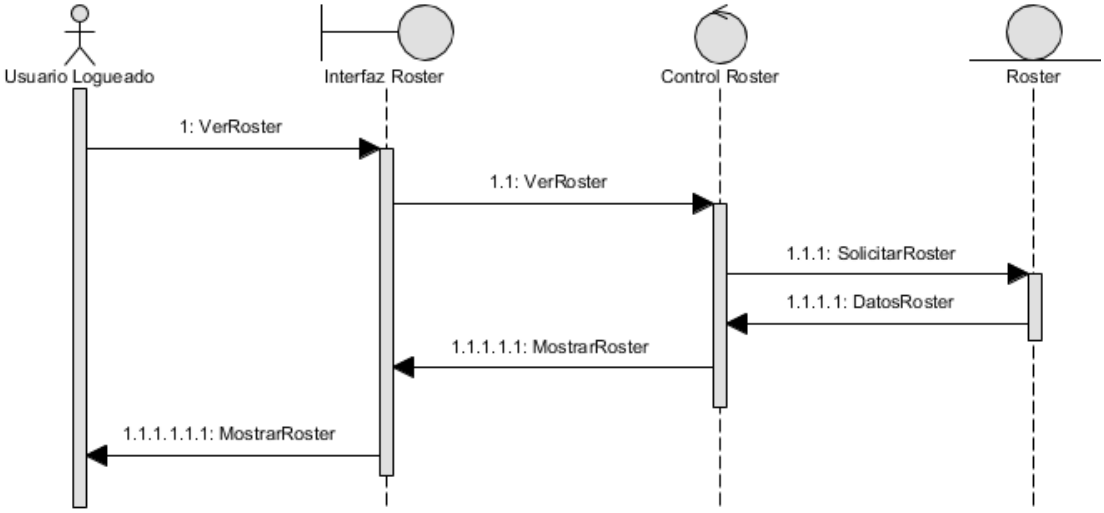


Ilustración 29: UC-0028 Ver equipo

Caso de uso 0029 **Modificar quinteto**

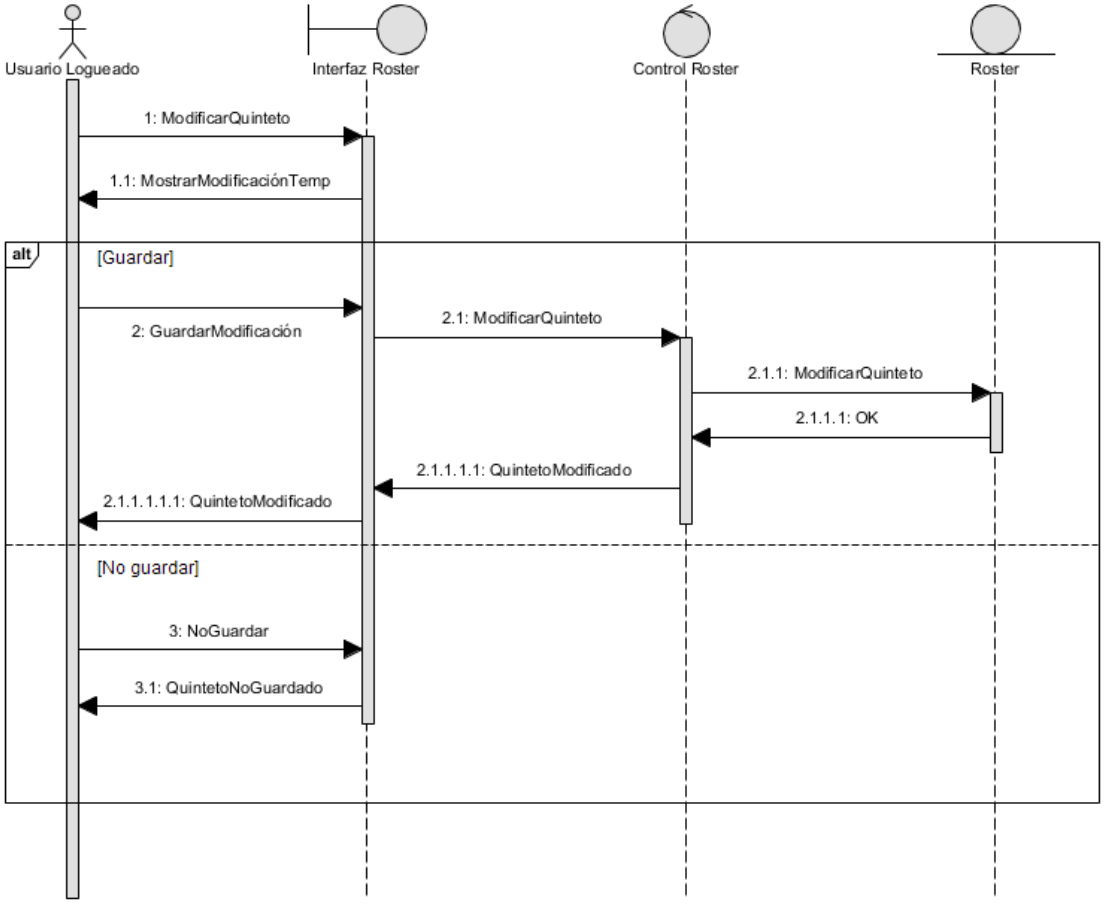


Ilustración 30: UC-0029 Modificar quinteto

Caso de uso 0030 **Actualizar roster**

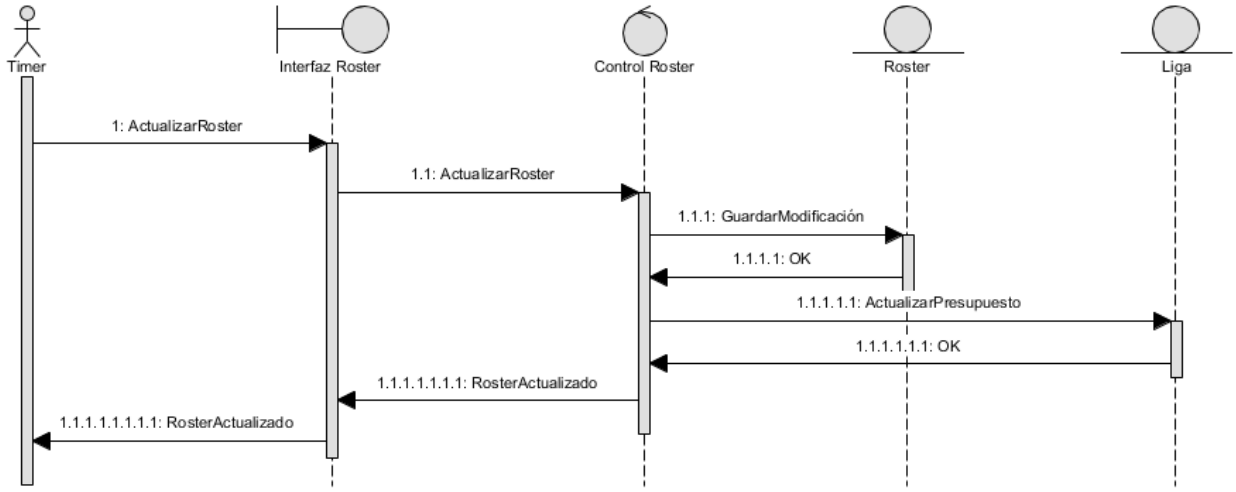


Ilustración 31: UC-0030 Actualizar roster

4. CLASES DE ANÁLISIS

En este apartado mediante los diagramas de comunicación se va a exponer cómo se relacionan y comunican entre sí las clases de análisis que forman parte de los diagramas de secuencia de la sección anterior. Estas clases están agrupadas en los paquetes que constituyen el sistema, las cuales debido a la relación entre paquetes aparecen en más de uno de estos. Cabe destacar que las relaciones entre estos son, el paquete "*Gestión de Ligas*" con los paquetes "*Gestión de Mercado*" y "*Gestión de Roster*".

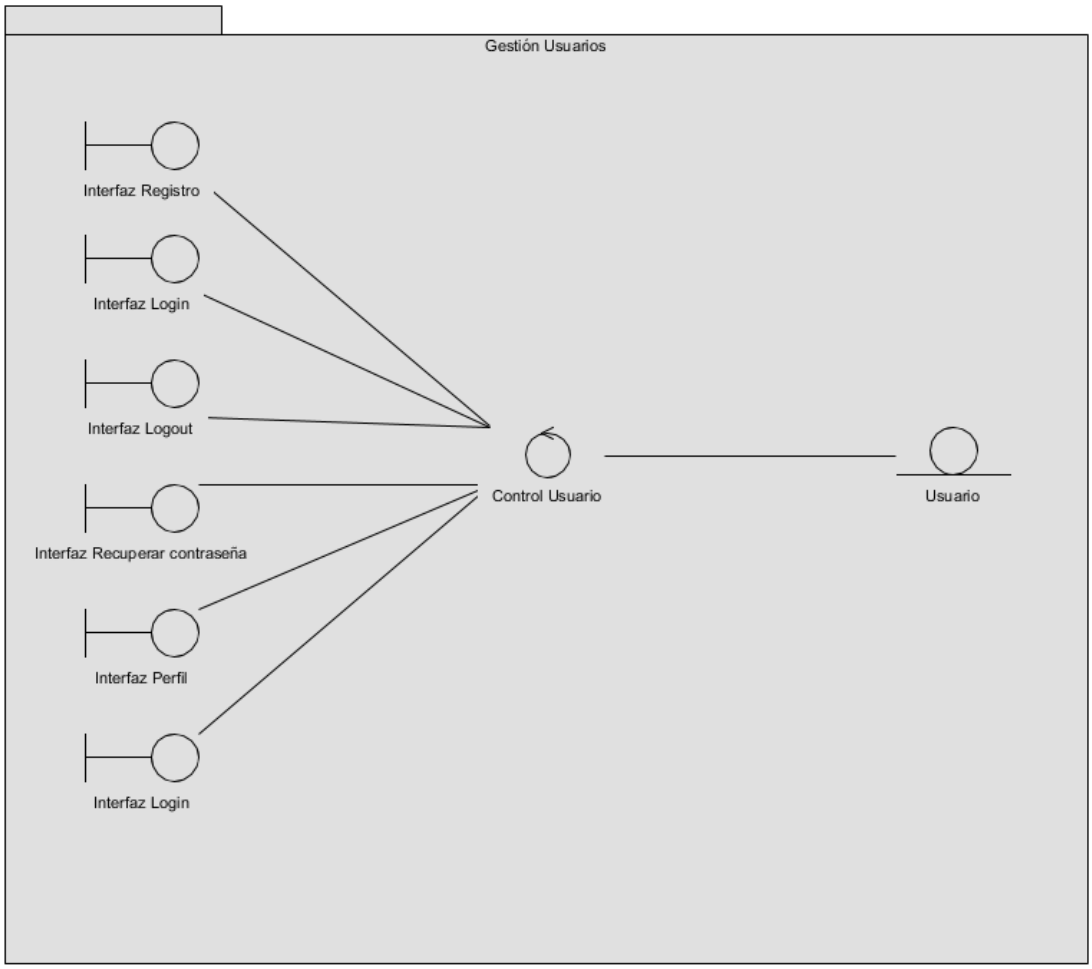


Ilustración 32: Diagrama de comunicación Paquete Gestión de Usuario

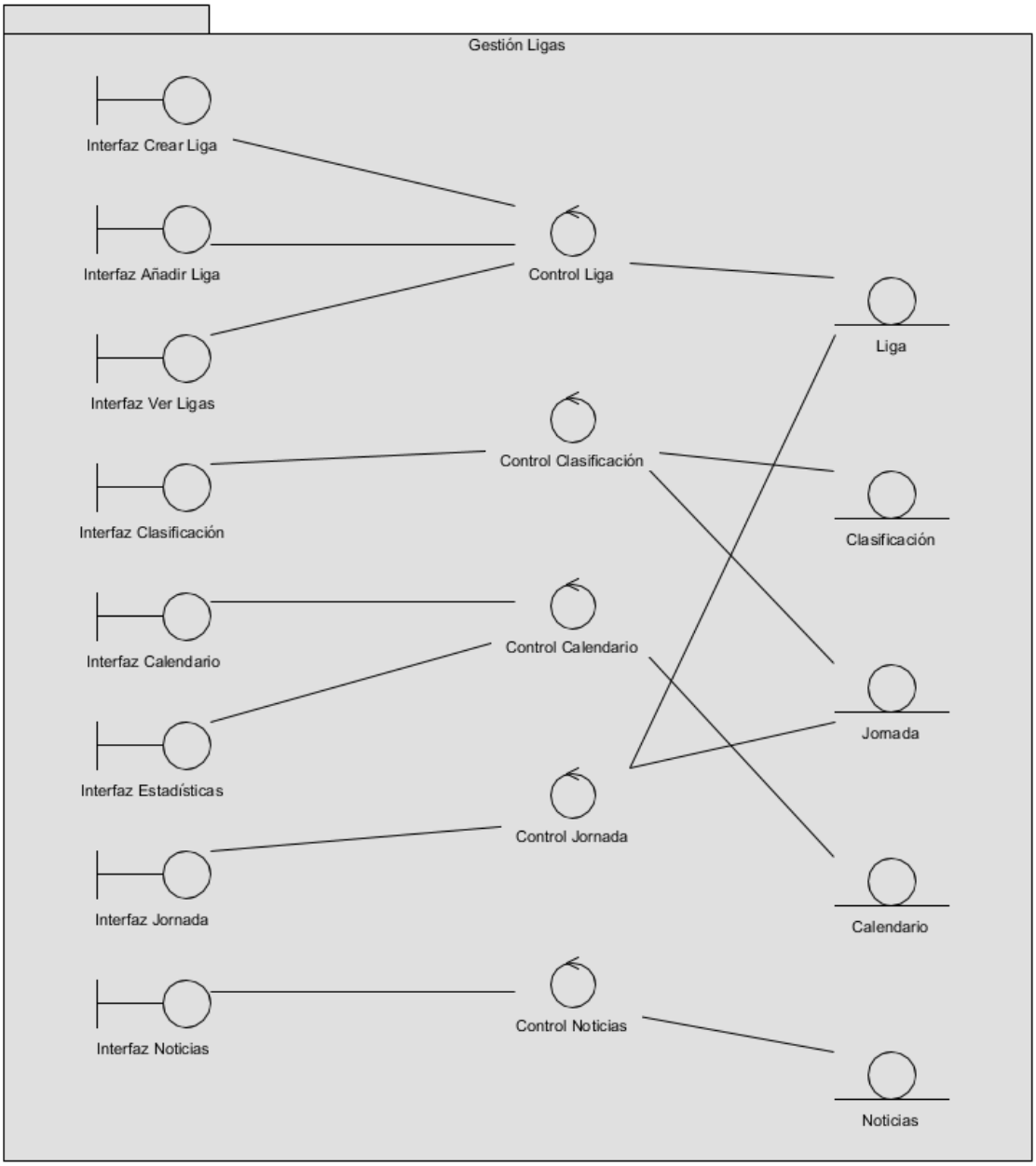


Ilustración 33: Diagrama de comunicación Paquete Gestión de Ligas

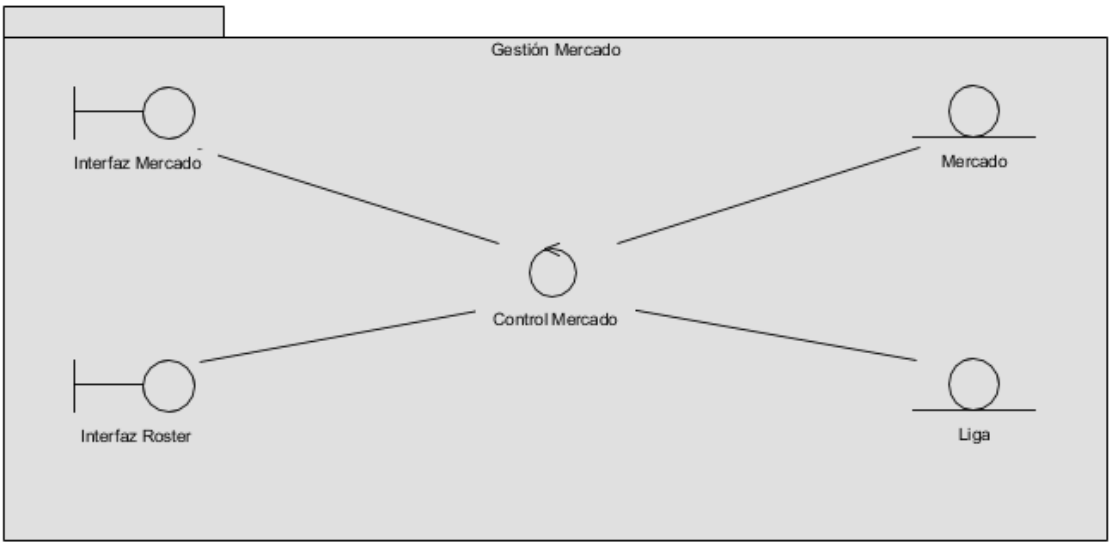


Ilustración 34: Diagrama de comunicación Paquete Gestión de Mercado

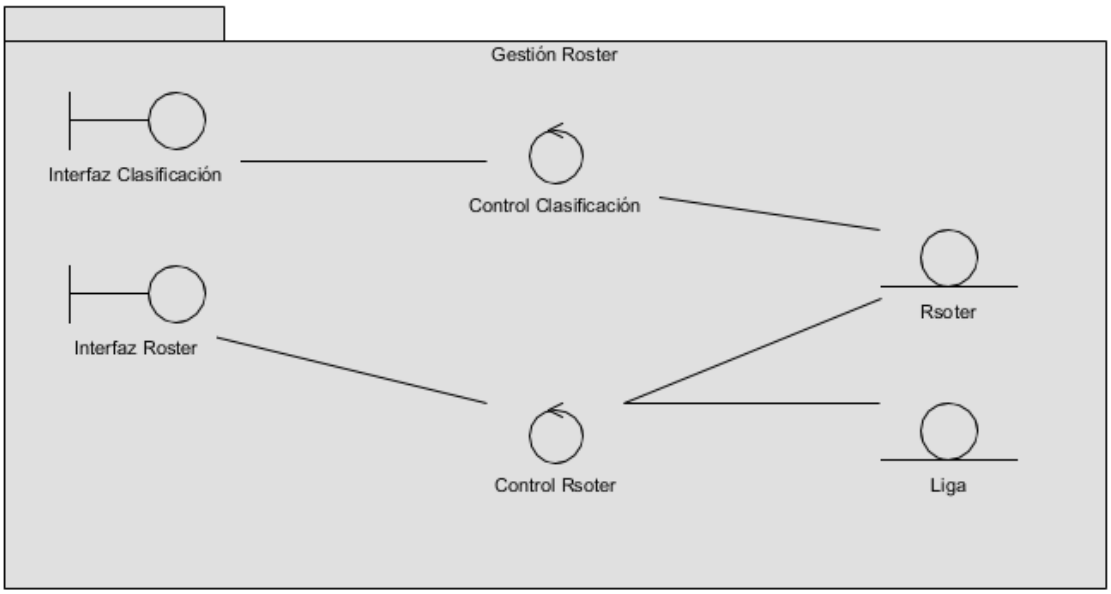


Ilustración 35: Diagrama de comunicación Paquete Gestión de Roster

5. VISTA DE ARQUITECTURA DEL MODELO DE ANÁLISIS

Siguiendo el modelo arquitectónico Modelo-Vista-Controlador, se permite separar la lógica de control de la interfaz de usuario y de los datos de la aplicación. Las clases de análisis mostradas en el punto anterior se agrupan en una vista de arquitectura del modelo de análisis.

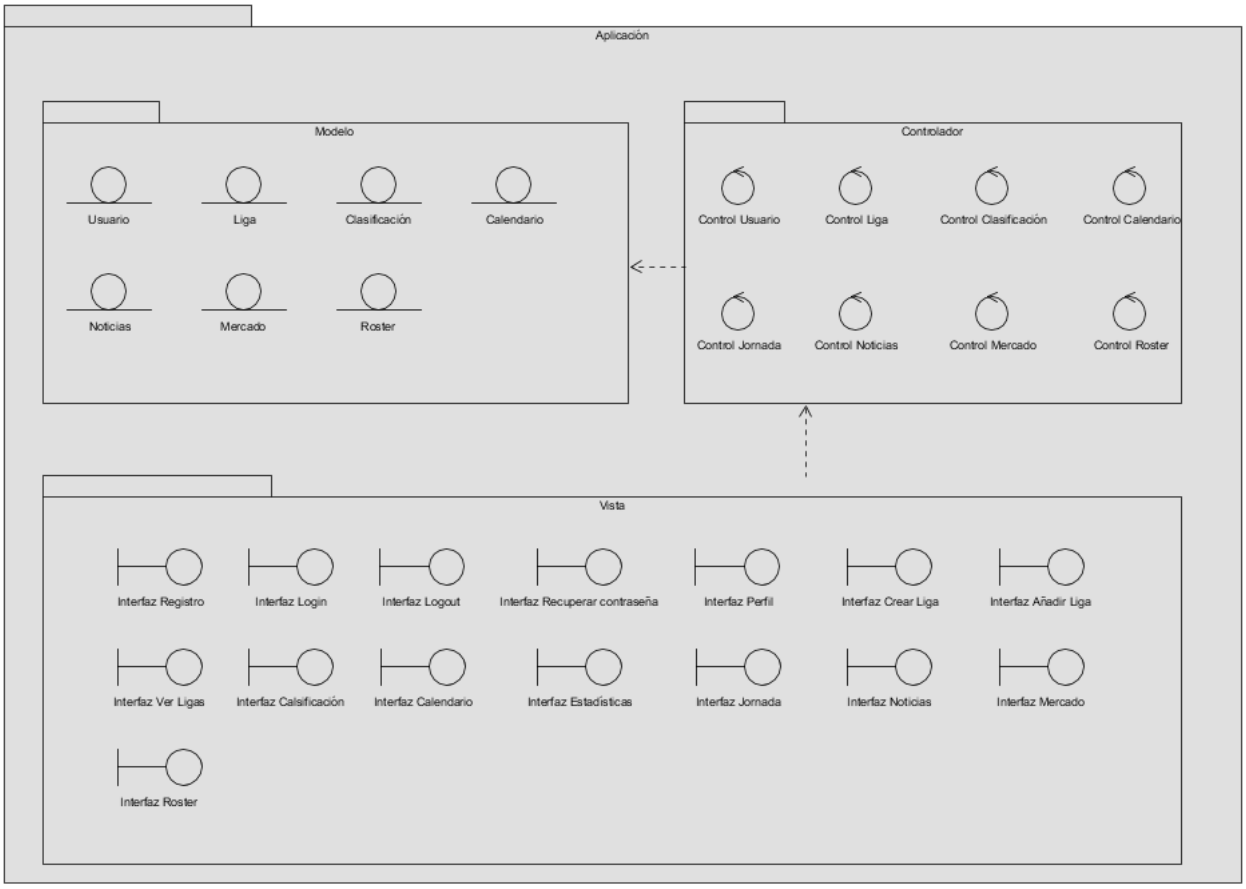


Ilustración 36: Vista de arquitectura

6. BIBLIOGRAFÍA

Berzal, F. *Relaciones entre clases: Diagramas de clases UML.* Obtenido de https://elvex.ugr.es/decsai/java/pdf/3C-Relaciones.pdf

García Peñalvo, F. J., & García Holgado, A. *Modelo de dominio.* Obtenido de https://repositorio.grial.eu/bitstream/grial/1153/1/8.%20Modelo%20de%20dominio.pdf

Moreno García, M., & García Peñalvo, F. J. *Transparencias Ingeniería del software, UML Vista de interacción.*

Servicio de Informática, Universidad de Alicante. Obtenido de Modelo vista controlador: https://si.ua.es/es/documentacion/asp-net-mvc-3/1-dia/modelo-vista-controlador-mvc.html#:~:text=Modelo%20Vista%20Controlador%20(MVC)%20es,control%20en%20tres%20componentes%20distintos.

ANEXO IV: Diseño del Sistema Software

Dream NBA: aplicación para la creación de ligas virtuales y gestión de un equipo propio

Trabajo de Fin de Grado

INGENIERÍA INFORMÁTICA

VNiVERSiDAD D SALAMANCA

CAMPUS DE EXCELENCIA INTERNACIONAL

Julio 2022

Autor

Jairo Sánchez García

Tutores

Gabriel Villarrubia González

Juan Francisco de Paz Santana

Héctor Sánchez San Blas

TABLA CONTENIDO

ÍNDICE DE ILUSTRACIONES

1. INTRODUCCIÓN

En este anexo se van a realizar las tareas propias a la fase de diseño software del Proceso Unificado. Para ello se procede a hacer un refinamiento de las especificaciones llevadas a cabo en la etapa previa desarrollada en el *Anexo III Análisis de requisitos*.

El documento contará con los siguientes apartados:

- **Modelo de diseño:** Se detallarán las tareas propias al modelo de diseño en distintos subapartados.
    - **Patrones arquitectónicos:** Se definirá el patrón arquitectónico que se debe seguir para estructurar el sistema. También se detallarán los distintos patrones de diseño que se utilizarán para su desarrollo.
    - **Subsistemas de diseño:** Se especificarán los paquetes que conforman al sistema y la comunicación entre ellos.
    - **Clases de diseño:** Se puntualizarán los métodos y atributos que dan forma a cada clase de diseño.
    - **Vista arquitectónica:** Se realizará la estructuración global del sistema.
    - **Realización de casos de uso:** Se refinarán los diagramas de secuencia establecidos en la anterior etapa.
- **Modelo de despliegue:** Se explicarán los distintos nodos que participarán en la ejecución, así como la comunicación entre estos.

2. MODELO DE DISEÑO

Permite modelar el sistema hacia un nivel más bajo cercano a la programación del sistema. A partir de esta fase se tendrá una especificación detallada de los paquetes, clases de diseño y casos de uso.

La arquitectura general que tomará el sistema será la relacionada con el patrón arquitectónico por capas, se explicará posteriormente.

El sistema estará desarrollado por las siguientes tecnologías:

- **TypeScript:** Lenguaje de programación elegido tanto para la parte del servidor como para la parte del cliente debido a la posibilidad de trabajar con clases de manera más fácil que JavaScript.
- **Vue.js e Ionic:** Frameworks de desarrollo web utilizados en la parte del cliente para el desarrollo de la capa de presentación. Estos frameworks utilizan el patrón MVVM(Model-View-ViewModel).
- **CSS:** Lenguaje de programación que sirve como apoyo a los marcos de trabajo superior para la creación de interfaces de usuario.
- **Firebase:** Se utilizarán los servicios que proporciona para el almacenamiento de datos y la autenticación de usuarios. Las prestaciones que ofrece para la base de datos siguen el patrón Publisher-Subscriber.

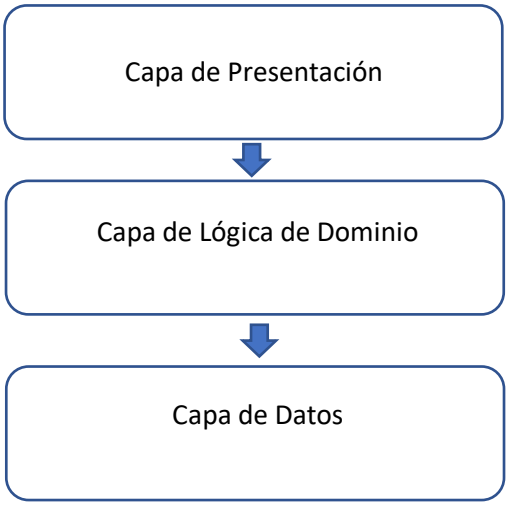
2.1 Patrones arquitectónicos

2.1.1 Patrón por capas

El patrón por capas permite organizar los componentes de un sistema en diferentes niveles con responsabilidades distintas. Las capas superiores colaboran con las inferiores, en el caso contrario se necesita de interfaces bien definidas para llevar a cabo la comunicación, de modo que las capas superiores sólo conocen las interfaces de comunicación de las inferiores, lo cual resulta muy útil a la hora de realizar cambios en alguna capa.

El sistema estará formado por las siguientes capas:

- **Capa de Presentación:** Representa la interacción entre el usuario y el sistema a través de interfaces de usuario. No debe de tener mucha funcionalidad, solamente mostrar y recoger datos del usuario y validaciones simples.
- **Capa de Lógica de Dominio:** Funcionalidad de la aplicación. Contiene los cálculos basados en la información dada por el usuario y datos almacenados.
- **Capa de Datos:** Lógica de comunicación con otros sistemas, como una API externa, o con la base de datos.



2.1.2 Patrón MVVM

Como se comentaba previamente esta arquitectura es la que siguen los marcos de trabajo Vue.js e Ionic. La estructura y modo de funcionamiento es similar a la que se presenta en Modelo-Vista-Controlador.

Siguiendo este patrón se separa la lógica de negocios con la presentación de la aplicación. Se tienen tres componentes principales cada uno con responsabilidades distintas, estos componentes son:

- **Modelo:** Contiene clases no visuales en donde se encapsulan los datos que utiliza la aplicación. Se comunica con servicios para proporcionar el acceso a datos.
- **Vista:** Define la estructura y apariencia que ve el usuario, no debe contener código relacionado con la lógica de negocio.
- **Modelo de vista:** Implementa propiedades a las que la vista puede enlazar datos y notifica a esta los cambios de estado. Las acciones que ofrece el modelo de vista definen la funcionalidad que ofrece la interfaz de usuario. Separa el modelo de la vista.

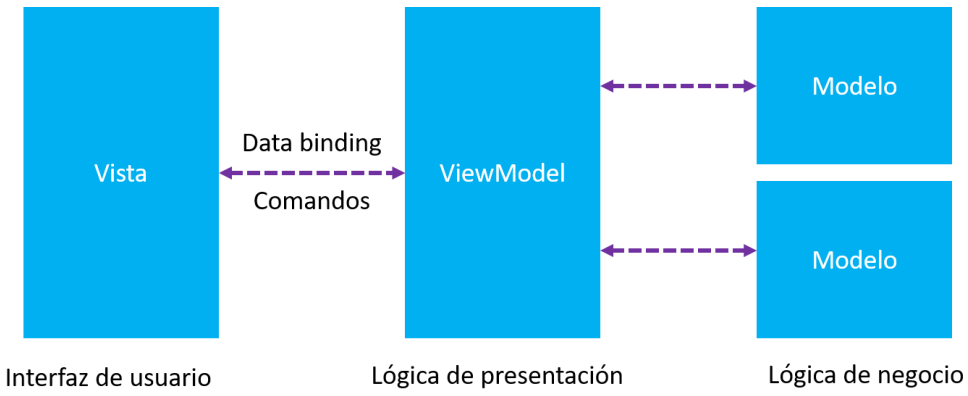


Ilustración 1: Esquema Patrón MVVM

2.1.3 Patrón Publisher-Subscriber

Es un patrón de intercambio de mensajes en el que participan emisores, publishers, y receptores, subscribers.

El funcionamiento del patrón consiste en que el publicador envía mensajes asíncronos hacia algún servidor, ya que desconoce el paradero y existencia de los subscriptores. Estos mensajes publicados se diferencias por clases, los subscriptores expresan interés en uno o varios tipos de clases, y sólo reciben los mensajes que corresponden a las clases con interés sin conocer al publicador. Esta relación anónima permite tener una alta escalabilidad.

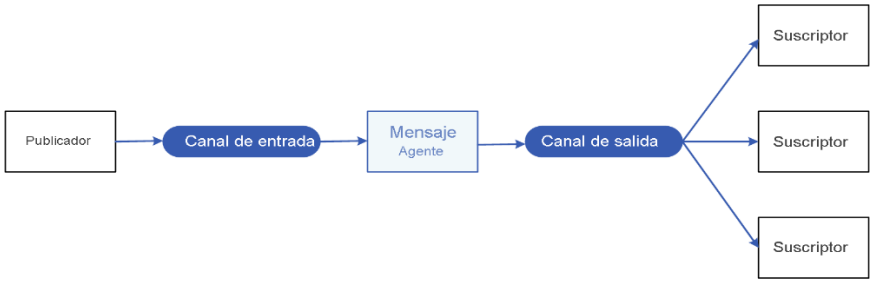


Ilustración 2: Esquema Patrón Publisher-Subscriber

2.2 Subsistemas de diseño

En este subapartado se realizará una descomposición del sistema en paquetes y subpaquetes, se podrá observar la disposición de estos paquetes en la *Ilustración 3*:

- **Aplicación Web:** Contiene la parte con la que el usuario interactuará. Se tiene un subpaquete llamado "*Components"* que tomará el papel de modelo de vista del patrón MVVM compuesto por varios archivos .vue, por otra parte se tiene otro subpaquete llamado "Views" el cual contiene los archivos .vue con las interfaces de usuario. El modelo en este paquete no existe como tal debido a la propiedad data que se dispone en Vue.
- **Servidor:** Está formado por tres subpaquetes, controlador, scripts y modelo. El subpaquete controlador es el middleware encargado de construir los endpoints, a través de los cuales los clientes realizarán las llamadas al servidor. Este subpaquete además, recoge las peticiones por parte del cliente, las procesa y responde al cliente, por otra parte tendrá comunicación con el modelo. El subpaquete scripts contendrá las clases que se deben ejecutar de manera automática cada cierto tiempo. El subpaquete modelo hará la comunicación con la base de datos y API externas y mantendrá la información de los datos.

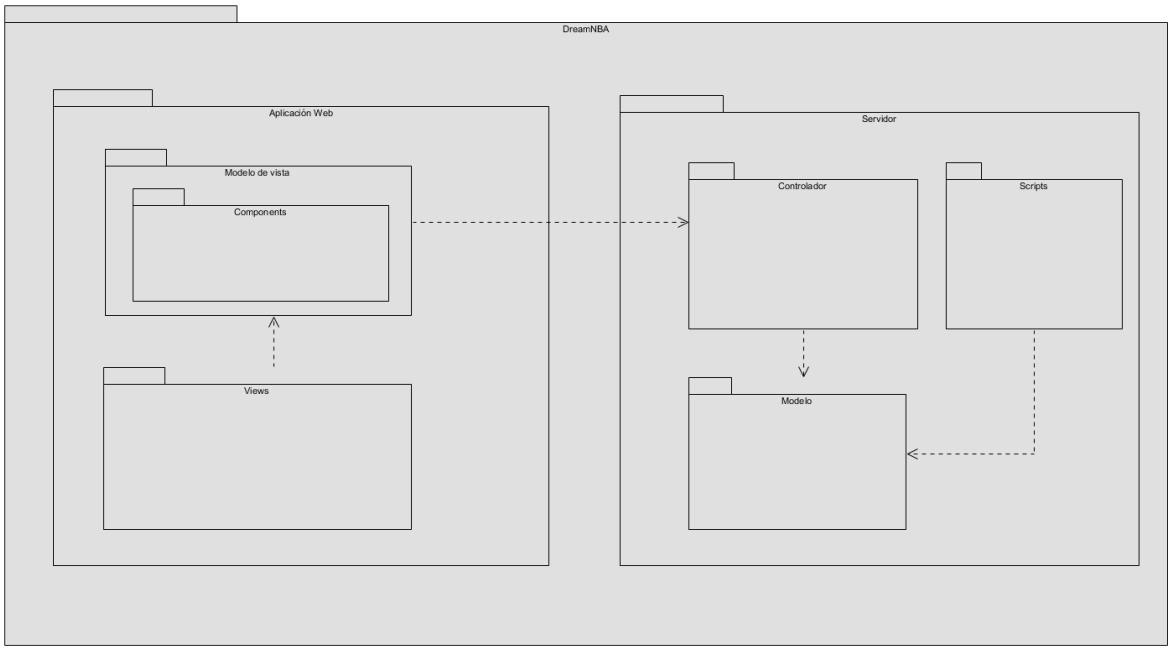


Ilustración 3: Modelo de diseño

2.3 Clases de diseño

Cada paquete/subpaquete del modelo de diseño está conformado por una serie de clases de diseño, que indican los métodos y atributos que tiene cada componente del paquete. Siguiendo con las diferentes estructuras se va a presentar primero el paquete "*Aplicación Web"* que sigue el patrón Modelo-Vista-Modelo de vista pero con la modificación de la ausencia del modelo representado con atributos de la propiedad data en los componentes del modelo de vista. Por último, se exhibe el paquete *"Servidor"*, el cual ocupa las capas de negocio y datos de la arquitectura global del sistema, la capa de presentación la formará los subpaquetes de la "*Aplicación Web".*

2.3.1 Paquete Aplicación Web

2.3.1.1 Modelo de vista

El modelo de vista está formado por los diferentes archivos .vue dentro del paquete llamado "*Components"*, estos archivos a su vez mantendrán una serie de atributos que actuarán de modelo.

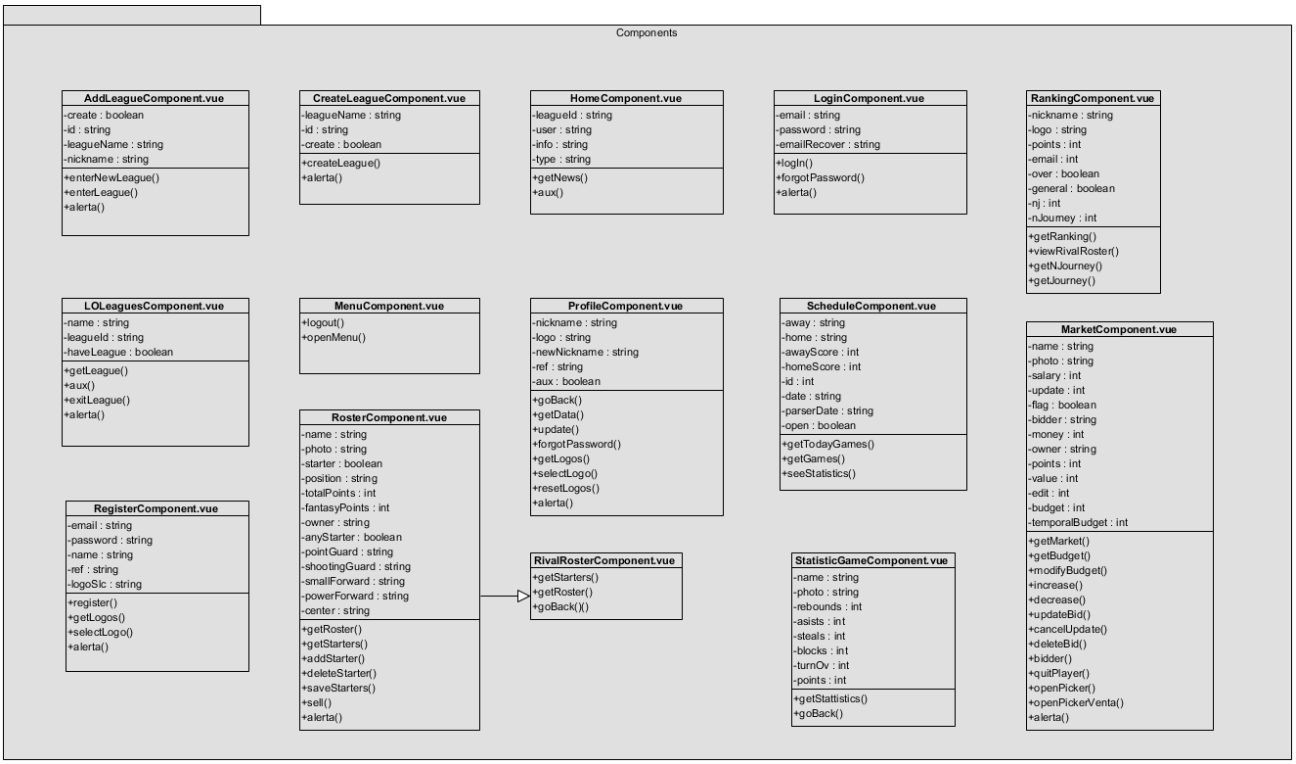


Ilustración 4: Clases de diseño Modelo de Vista

2.3.1.2 Vista

La vista representa las páginas a través las cuales el usuario interaccionará con el sistema, su funcionalidad está limitada a pequeñas comprobaciones y carecen de atributos.

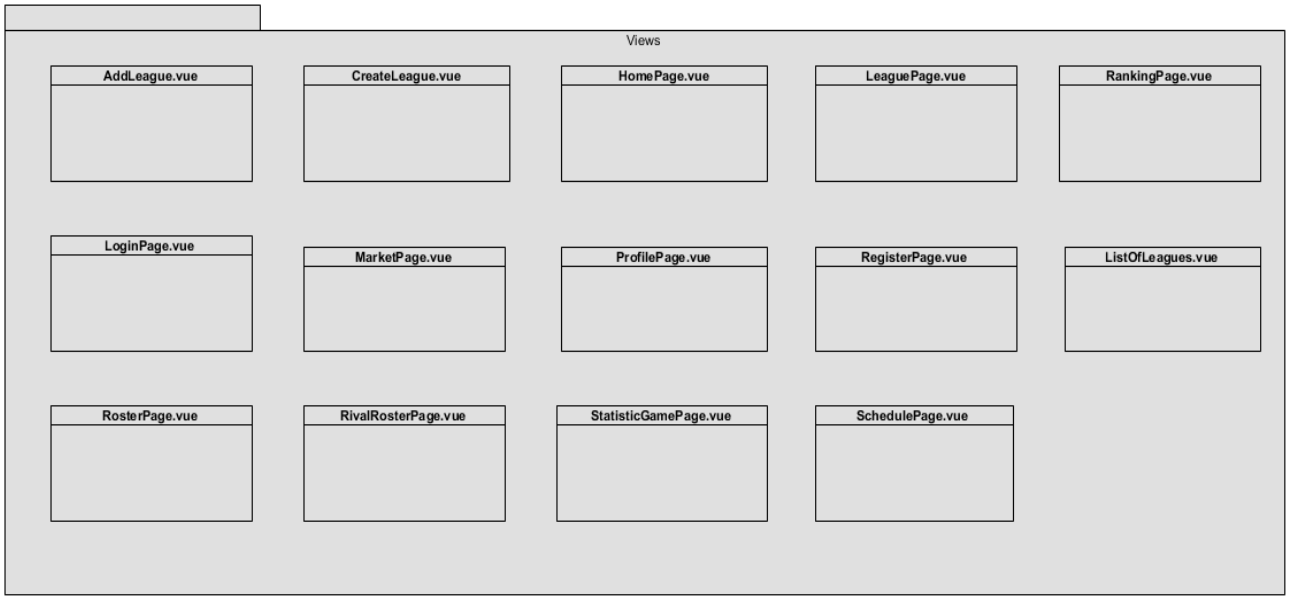


Ilustración 5: Clases de diseño Vista

2.3.2 Paquete Servidor

2.3.2.1 Capa Lógica de Dominio

Esta capa estará formada por los subpaquetes "*Controlador"* y "*Scripts".* La diferencia entre ambos reside en que el controlador proporciona los puntos de comunicación con el cliente, mientras que scripts ofrece cálculos automatizados en el tiempo.

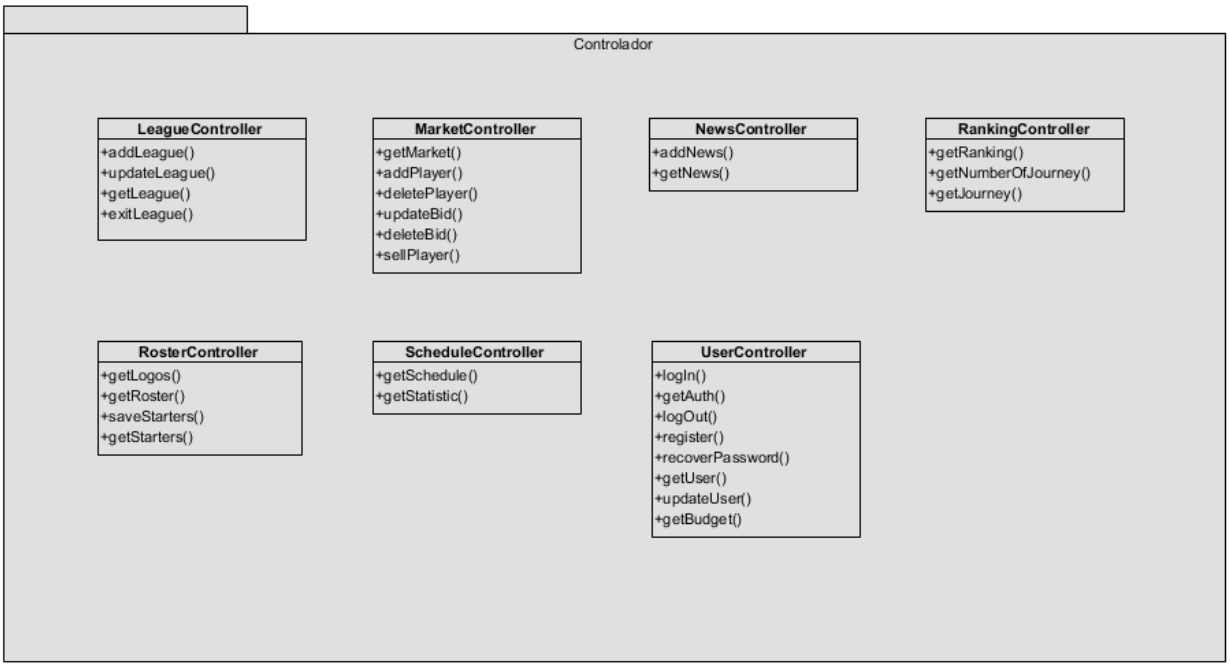


Ilustración 6: Clases de diseño Controlador

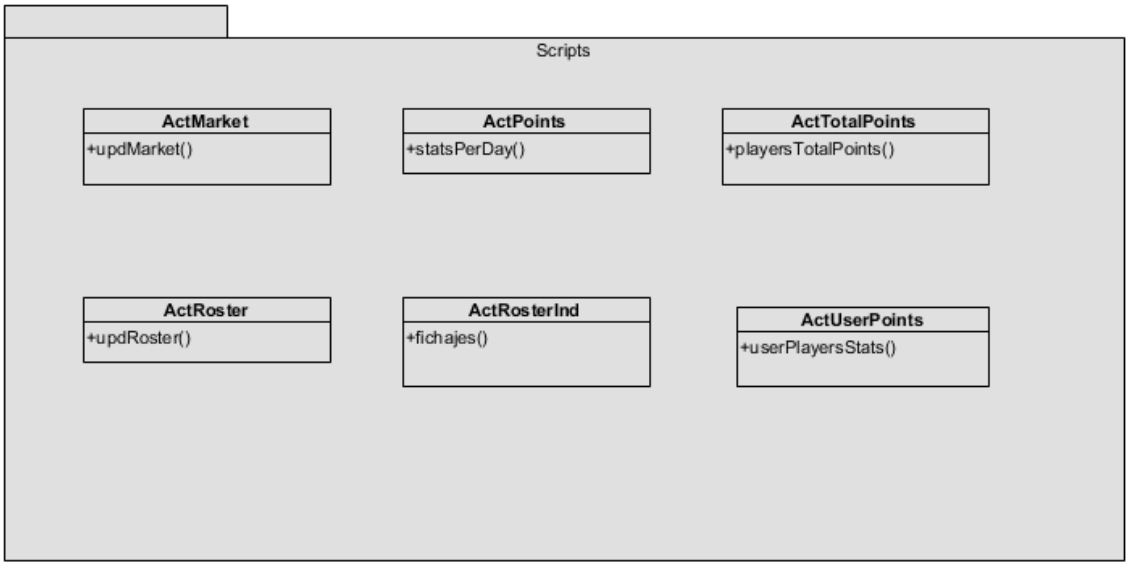


Ilustración 7: Clases de diseño Scripts

2.3.2.2 Capa de Datos

En esta capa se encuentra el paquete "*Modelo"*, el cual mantiene la información de los datos y realiza las comunicaciones con la base de datos y aplicaciones externas que proveen al sistema con datos, como en este caso una API de NBA.

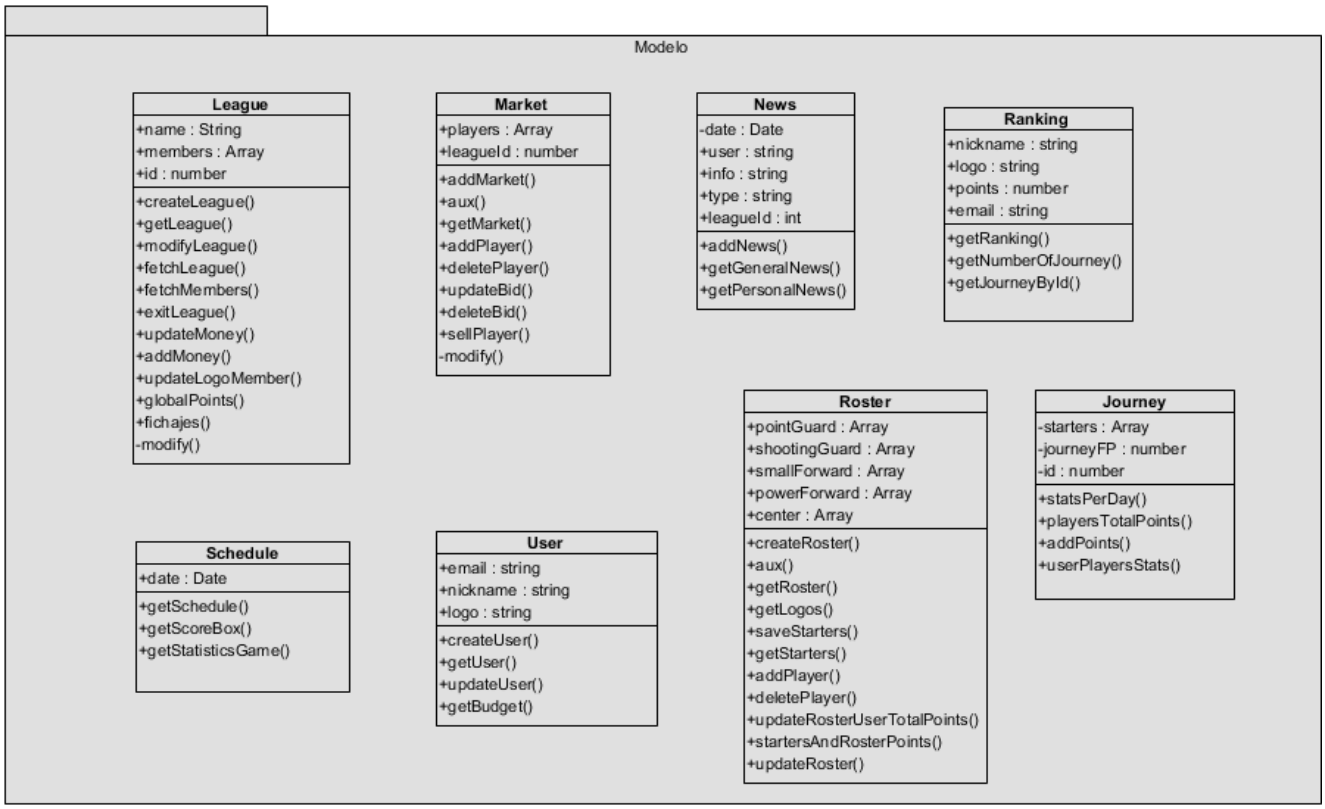


Ilustración 8: Clases de diseño Modelo

2.4 Vista arquitectónica

Siguiendo los paquetes desglosados en el punto anterior, se puede modelar la vista arquitectónica del sistema. Como ya se mencionó el paquete "*Aplicación Web"* sigue el patrón MVVM con la modificación del modelo, pero la arquitectura general sigue el patrón por capas. En la *Ilustración 9* se puede observar esta arquitectura.

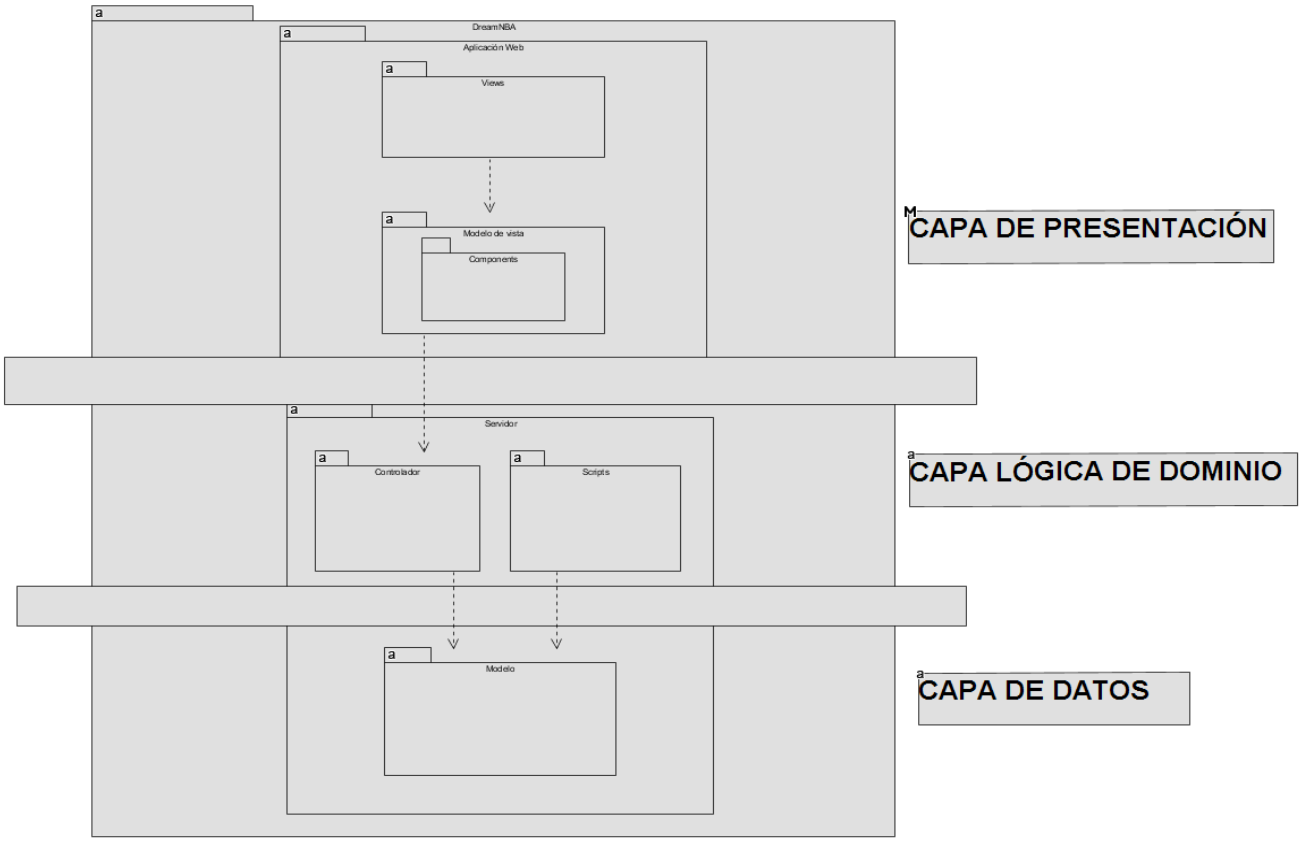


Ilustración 9: Vista arquitectónica

2.5 Realización de Casos de Uso

A partir de los diagramas de secuencia detallados en el *Anexo III Análisis de requisitos*, se va a llevar a cabo su refinamiento con el objetivo de obtener los diagramas de casos de uso diseño.

2.5.1 Diagramas de secuencia del Paquete Gestión de Usuarios

Caso de uso 0001 **Crear cuenta**

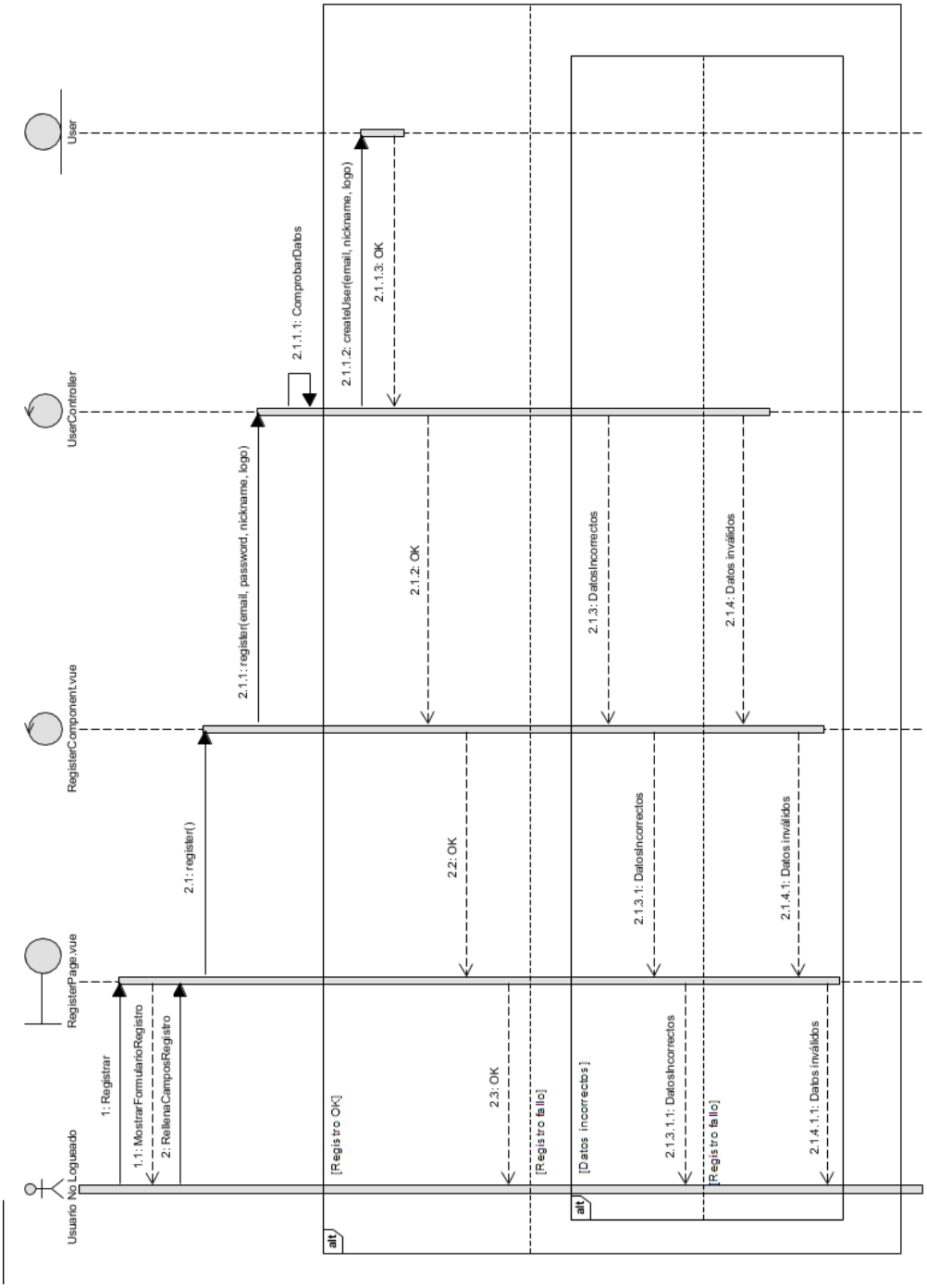


Ilustración 10: UC-0001 Crear cuenta

Caso de uso 0002 **Iniciar sesión**

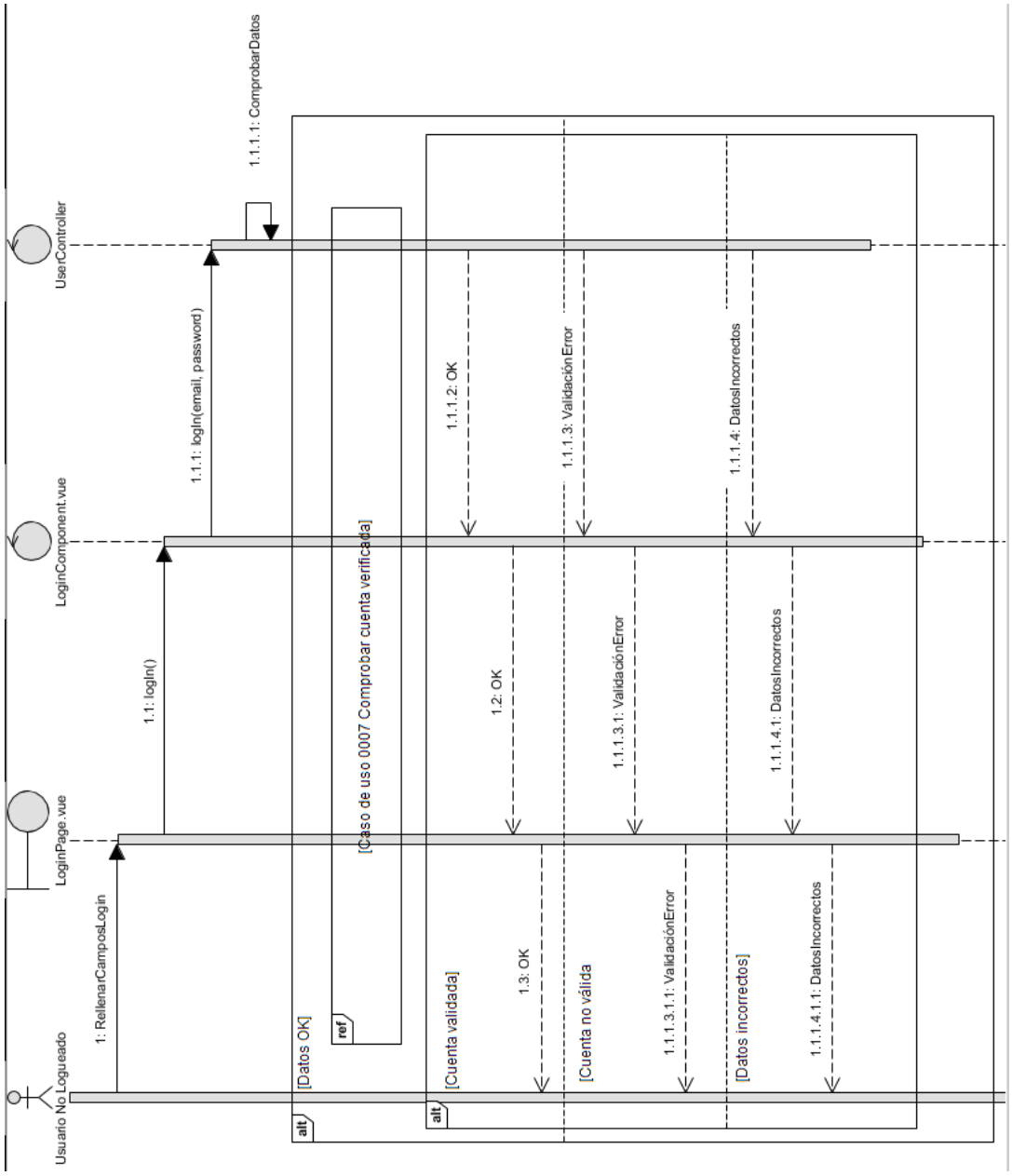


Ilustración 11: UC-0002 Iniciar sesión

Caso de uso 0003 **Salir**

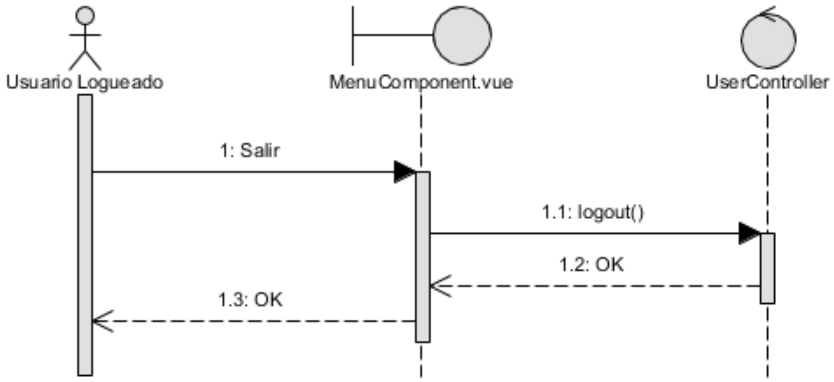


Ilustración 12: UC-0003 Salir

Caso de uso 0004 **Recuperar contraseña**

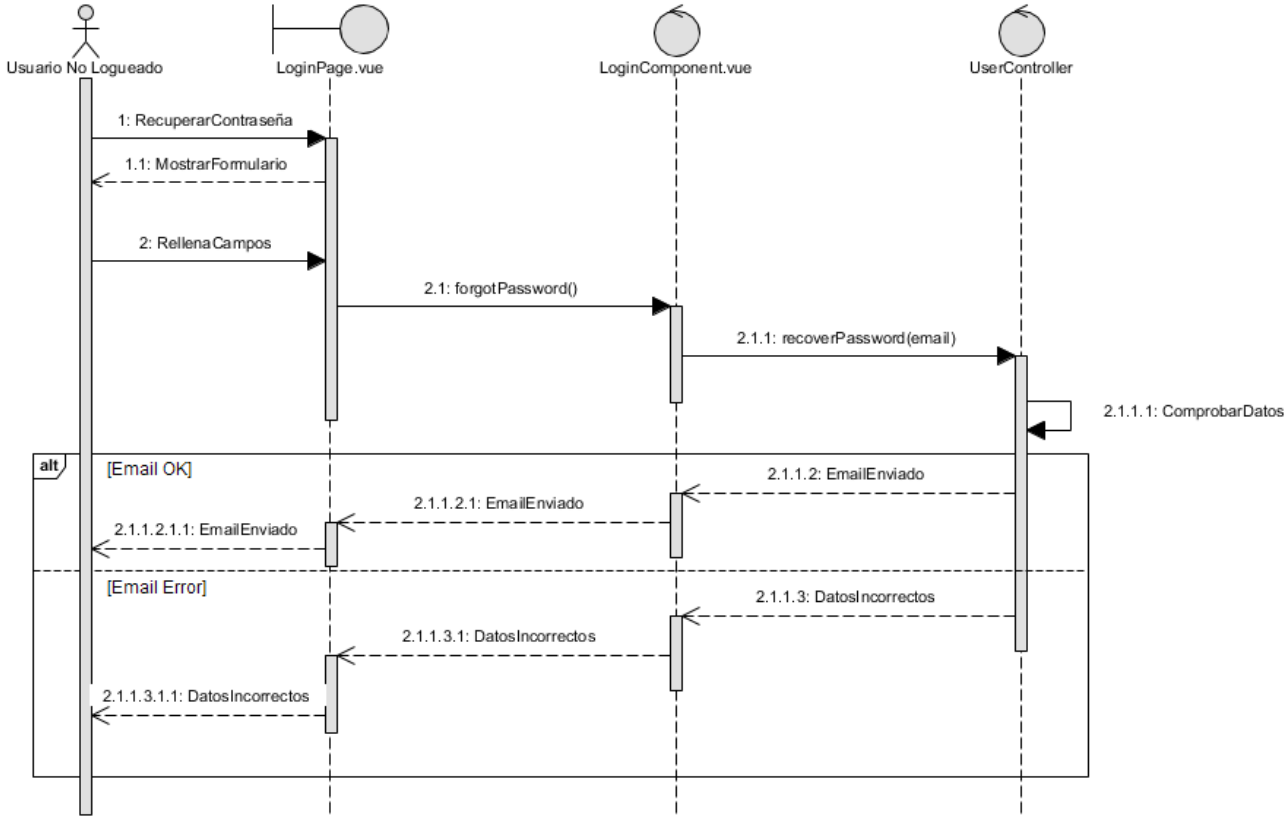


Ilustración 13: UC-0004 Recuperar contraseña

Caso de uso 0005 **Ver perfil**

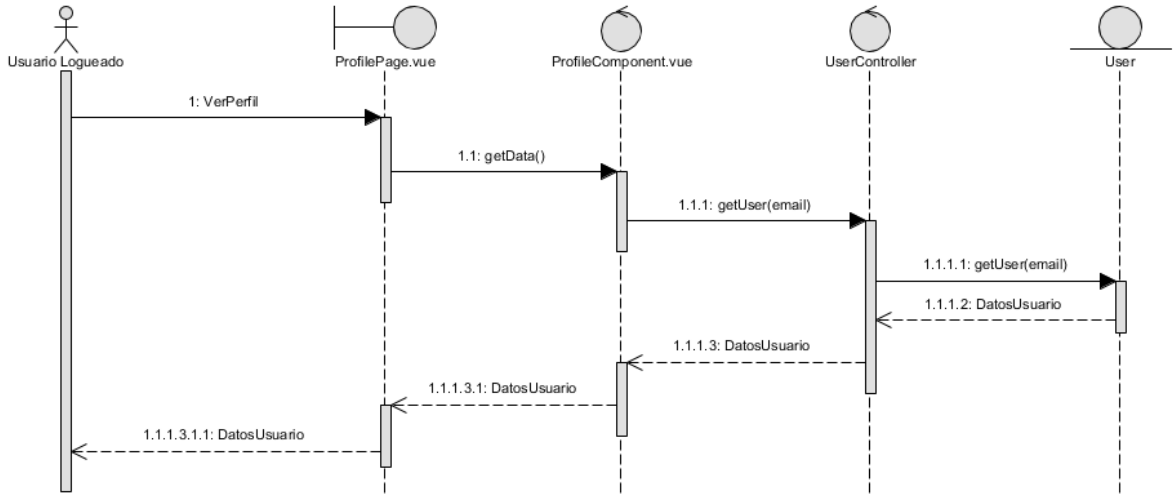


Ilustración 14: UC-0005 Ver perfil

Caso de uso 0006 **Modificar datos cuenta**

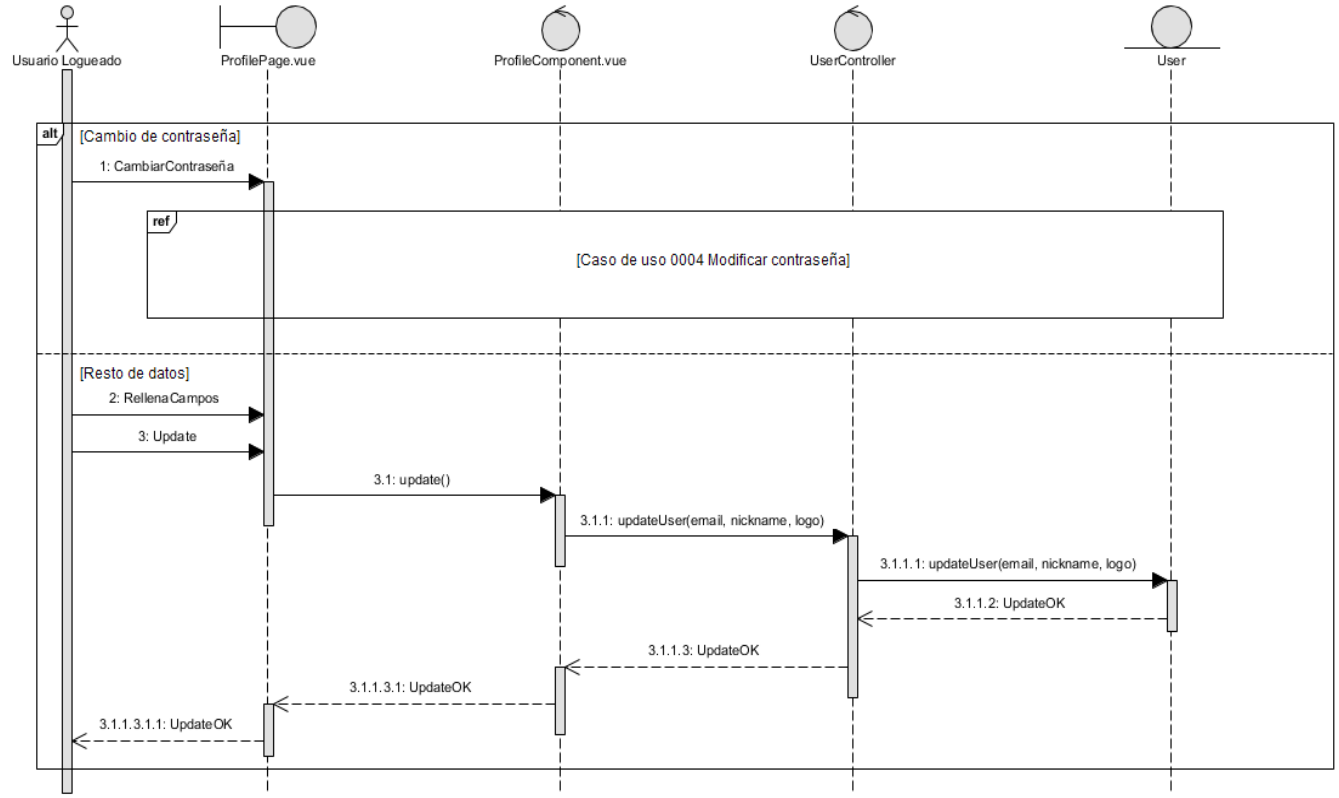


Ilustración 15: UC-0006 Modificar datos cuenta

Caso de uso 0007 **Comprobar cuenta verificada**

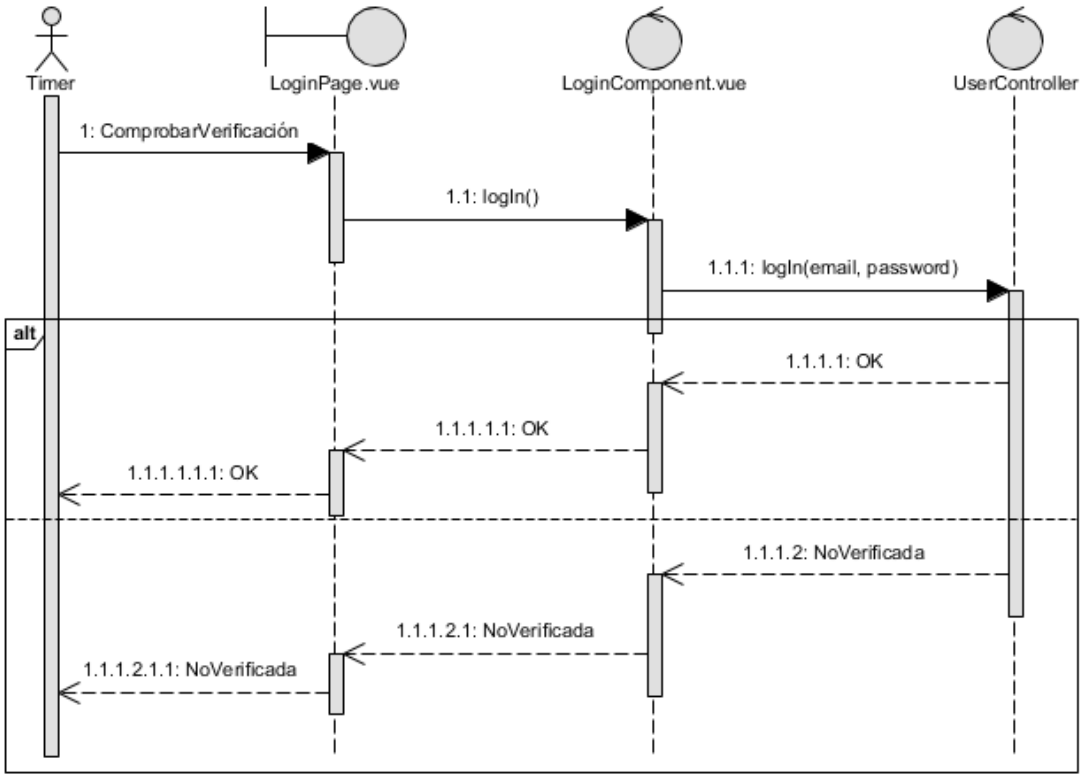


Ilustración 16: UC-0007 Comprobar cuenta verificada

2.5.2 Diagramas de secuencia del Paquete Gestión de Ligas

Caso de uso 0008 **Crear Liga**

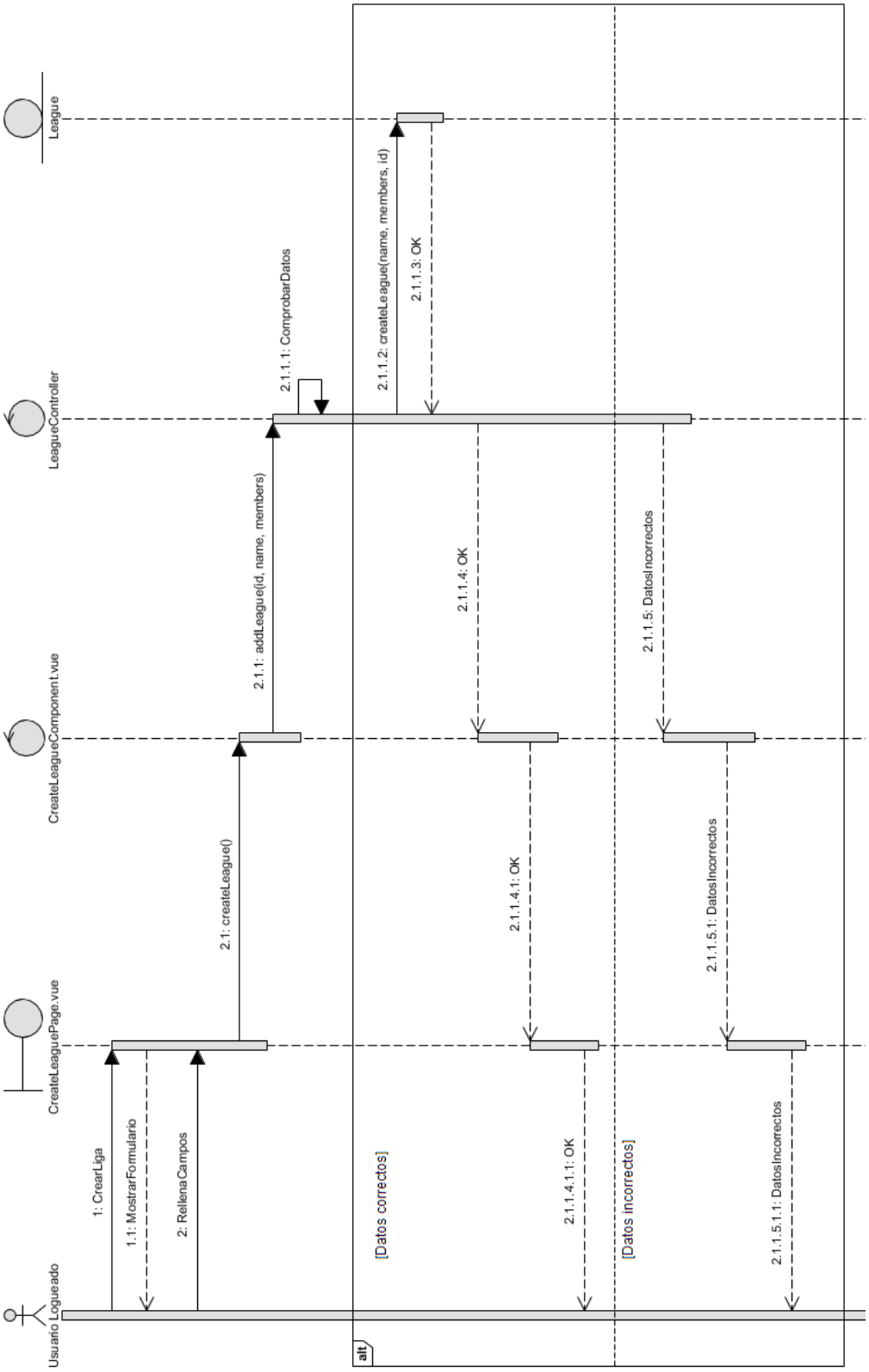


Ilustración 17: UC-0008 Crear liga

Caso de uso 0009 **Unirse a liga a través de código**

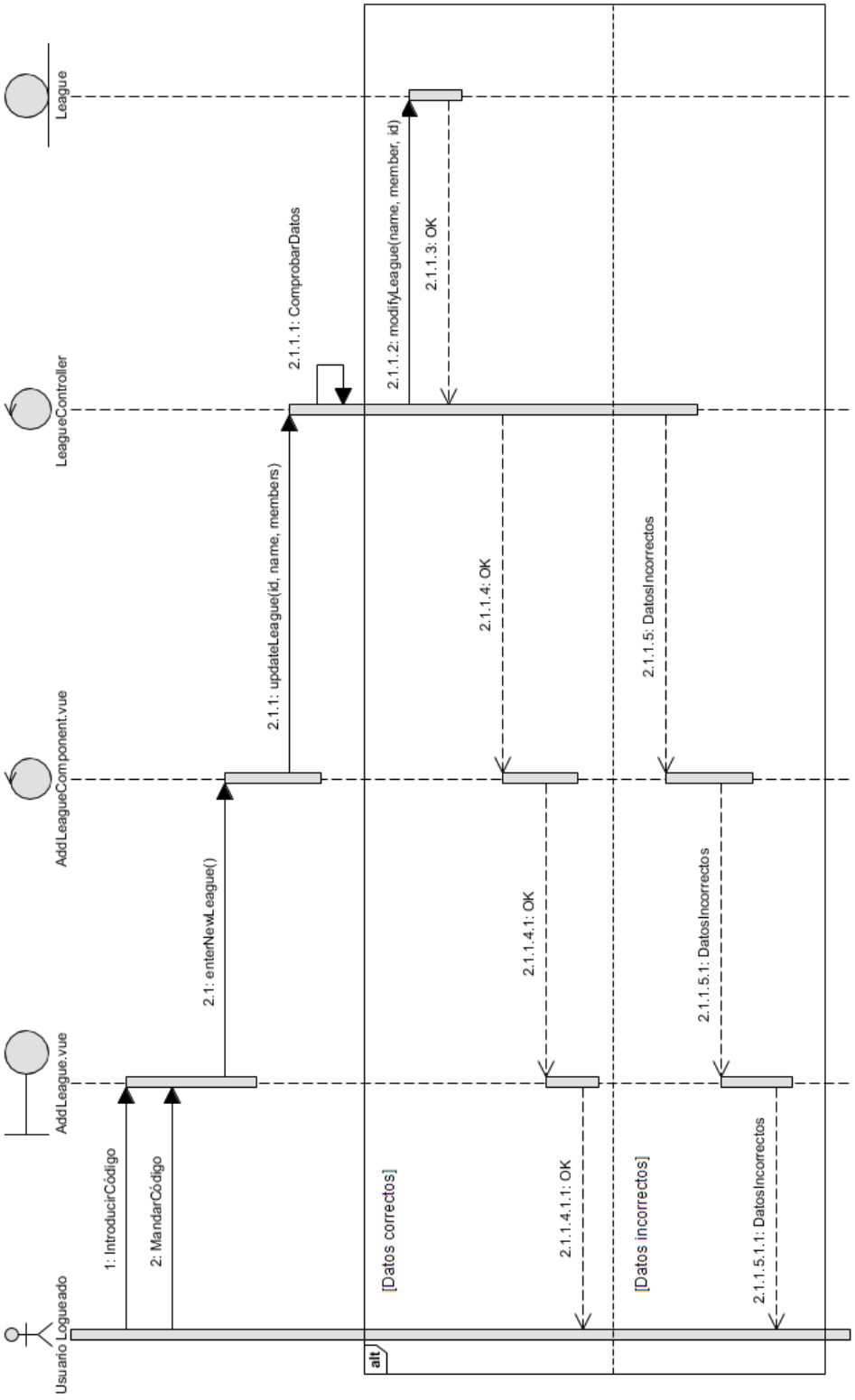


Ilustración 18: UC-0009 Unirse a liga a través de código

Caso de uso 0010 **Ver ligas que participa**

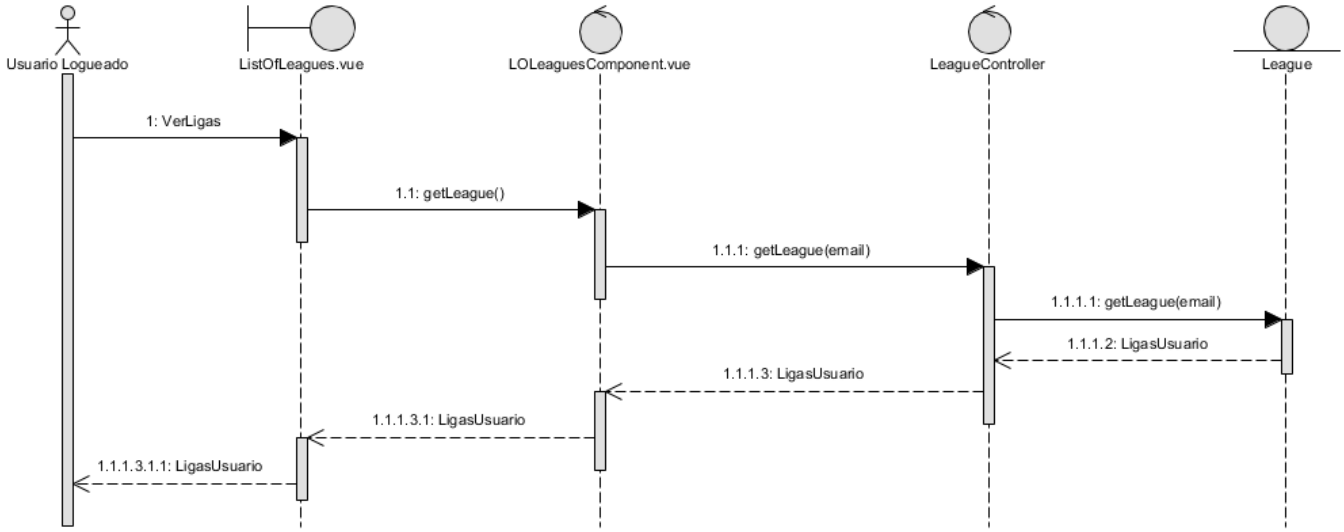


Ilustración 19: UC-0010 Ver ligas que participa

Caso de uso 0011 **Entrar a liga que participa**

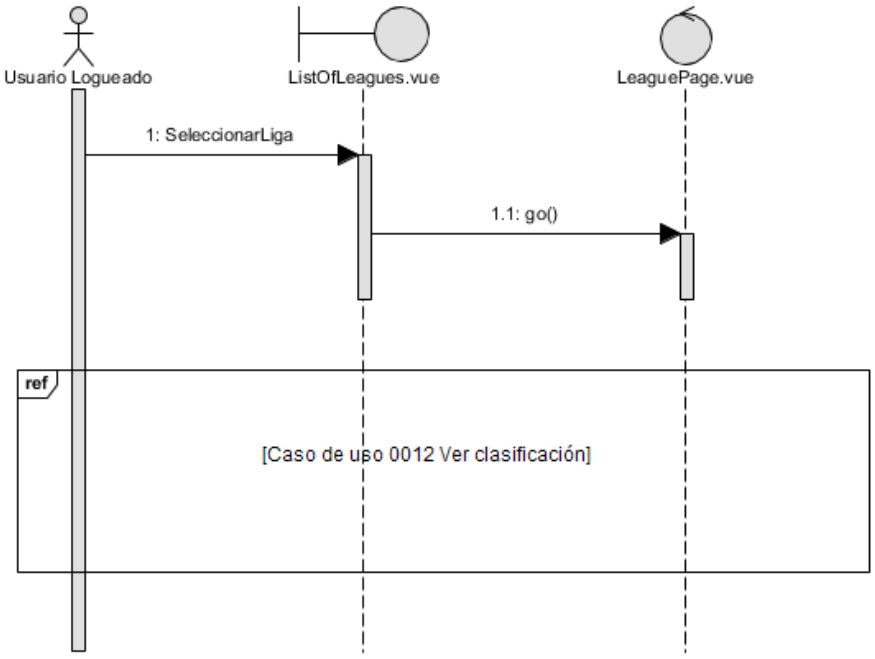


Ilustración 20: UC-0011 Entrar a liga que participa

Caso de uso 0012 **Ver clasificación**

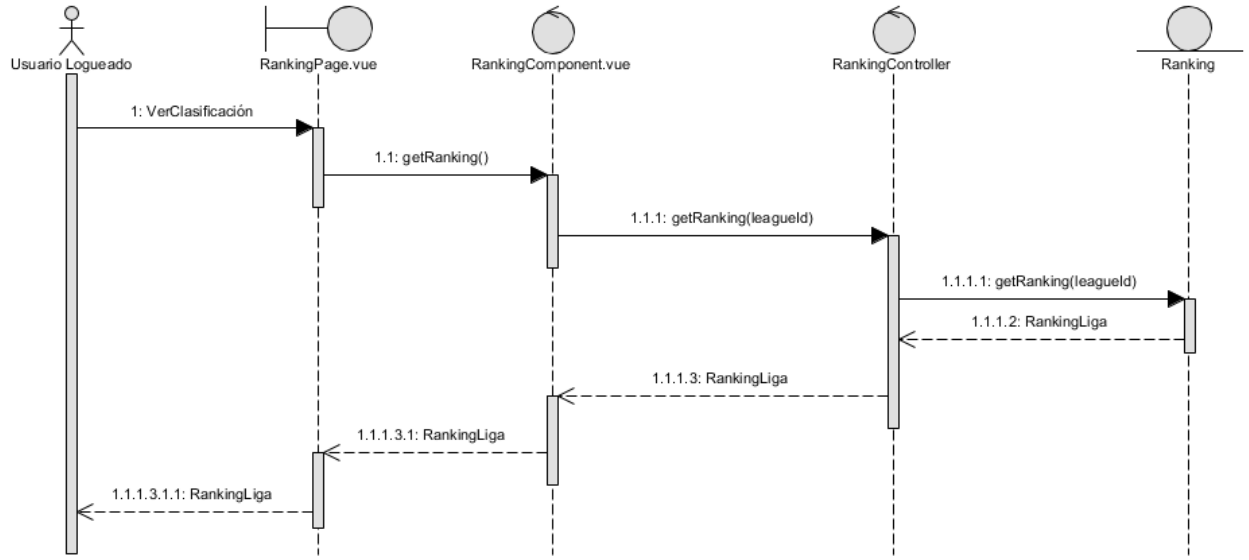


Ilustración 21: UC-0012 Ver clasificación

Caso de uso 0013 **Ver clasificación por jornada**

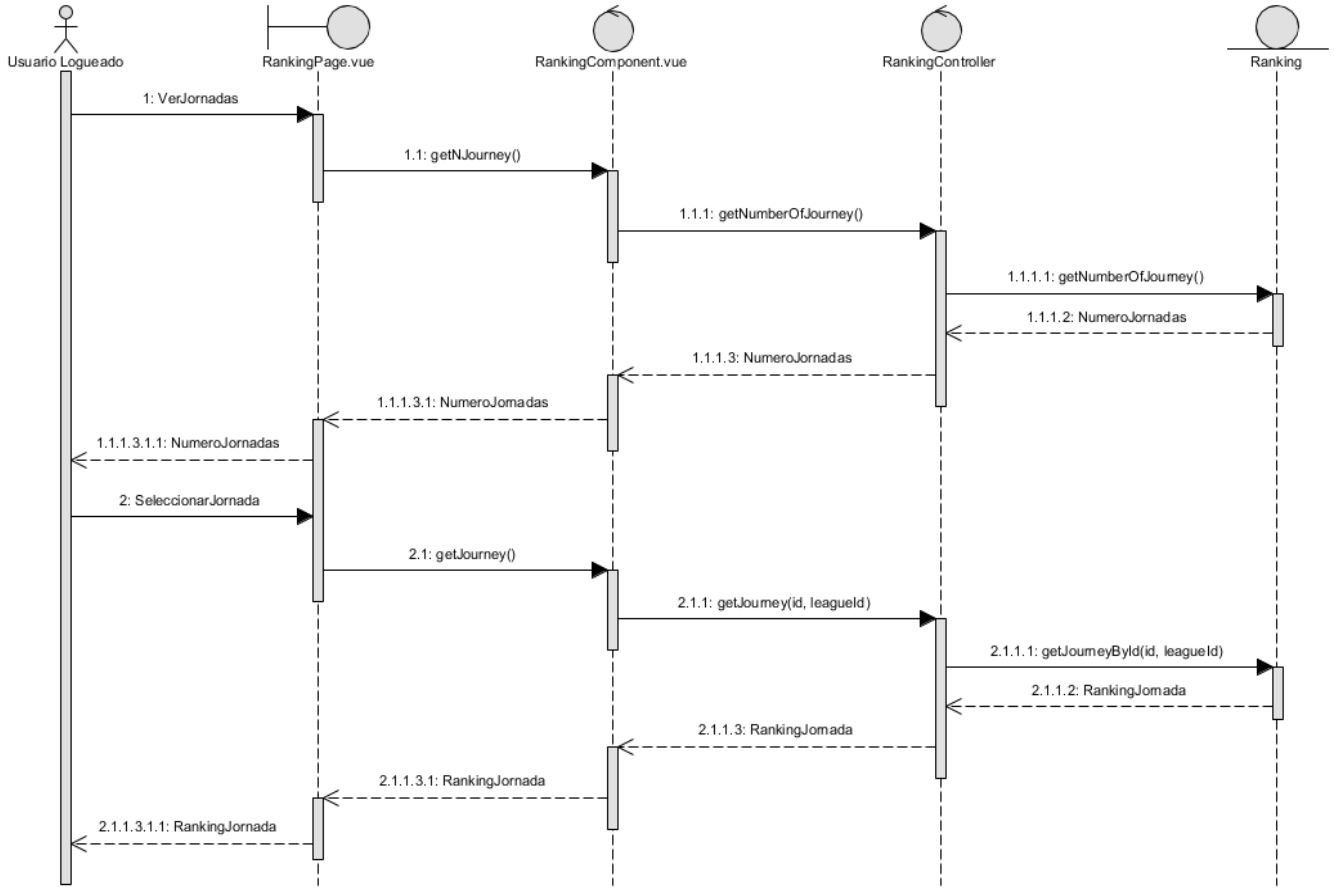


Ilustración 22: UC-0013 Ver clasificación por jornada

Caso de uso 0014 **Ver calendario**

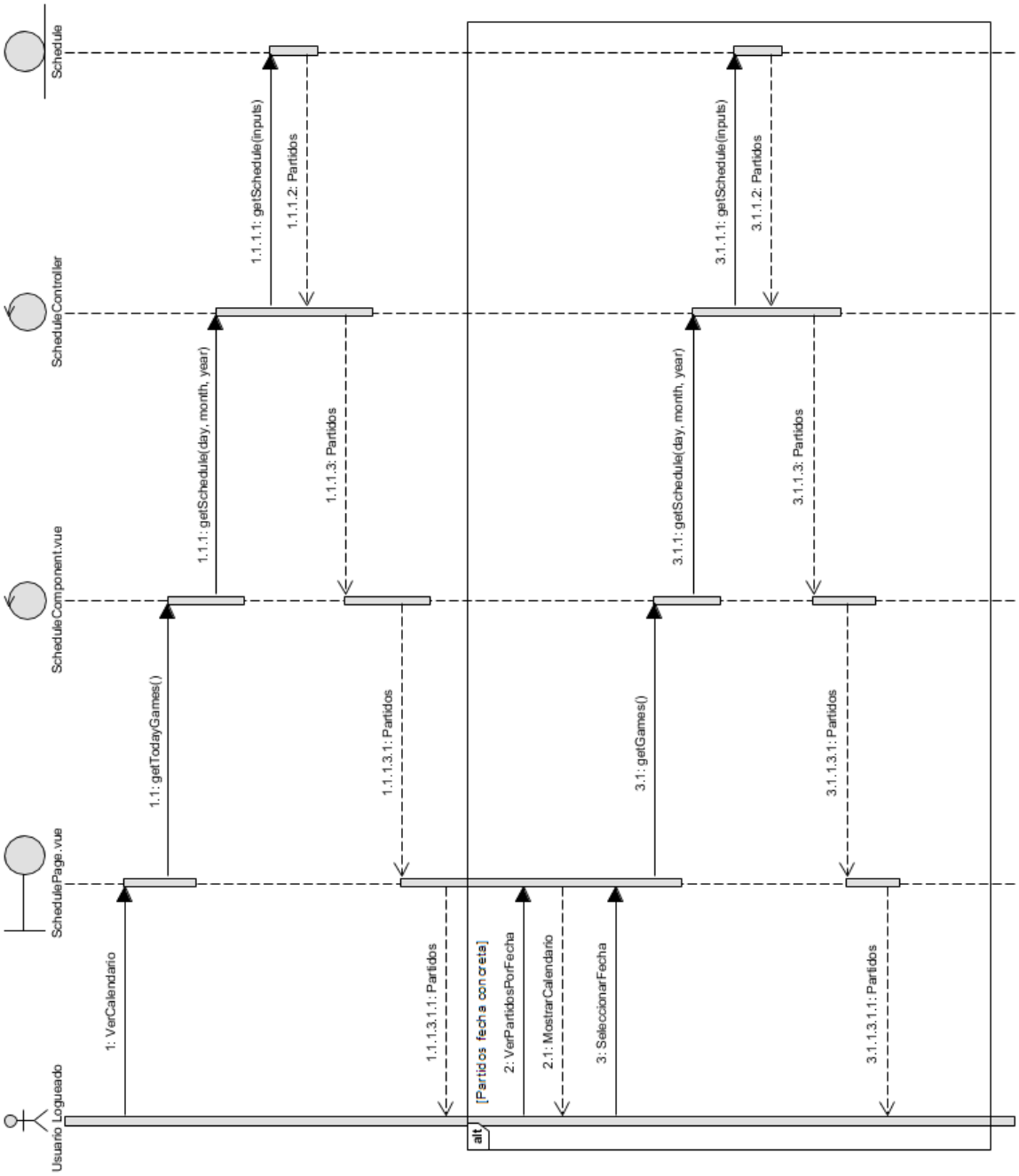


Ilustración 23: UC-0014 Ver calendario

Caso de uso 0015 **Ver estadísticas por partido**

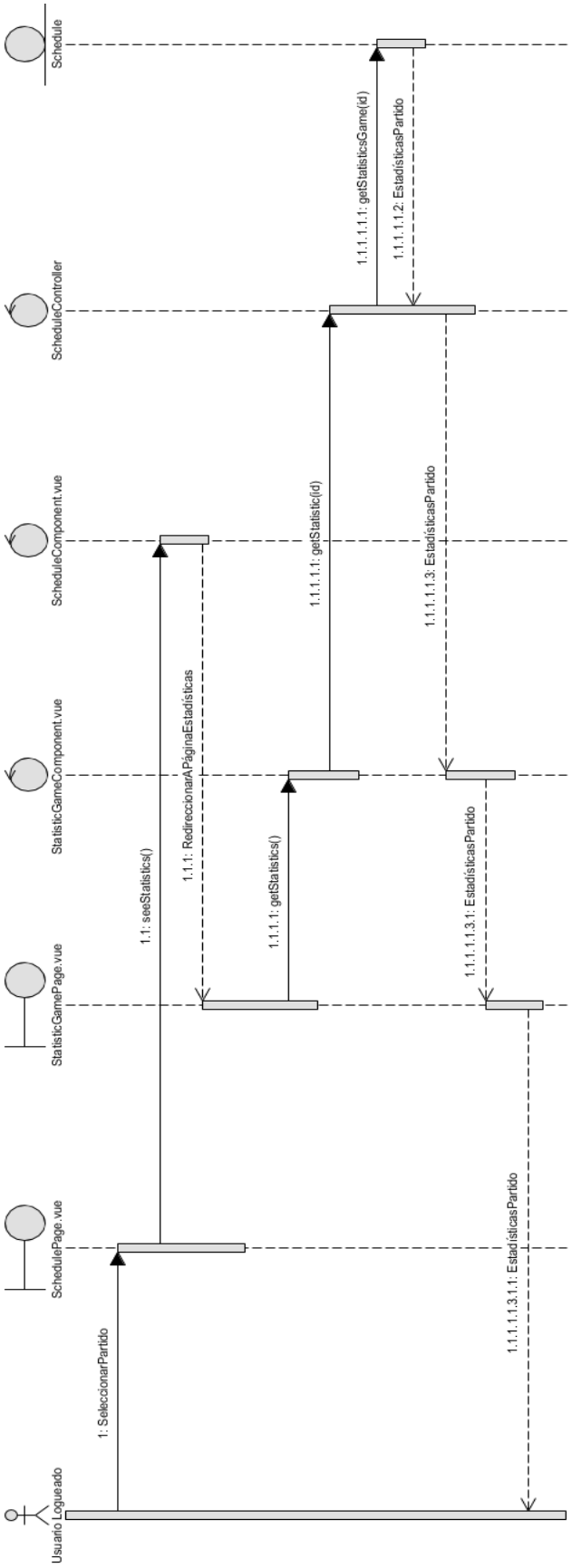


Ilustración 24: UC-0015 Ver estadísticas por partido

Caso de uso 0016 **Ver noticias**

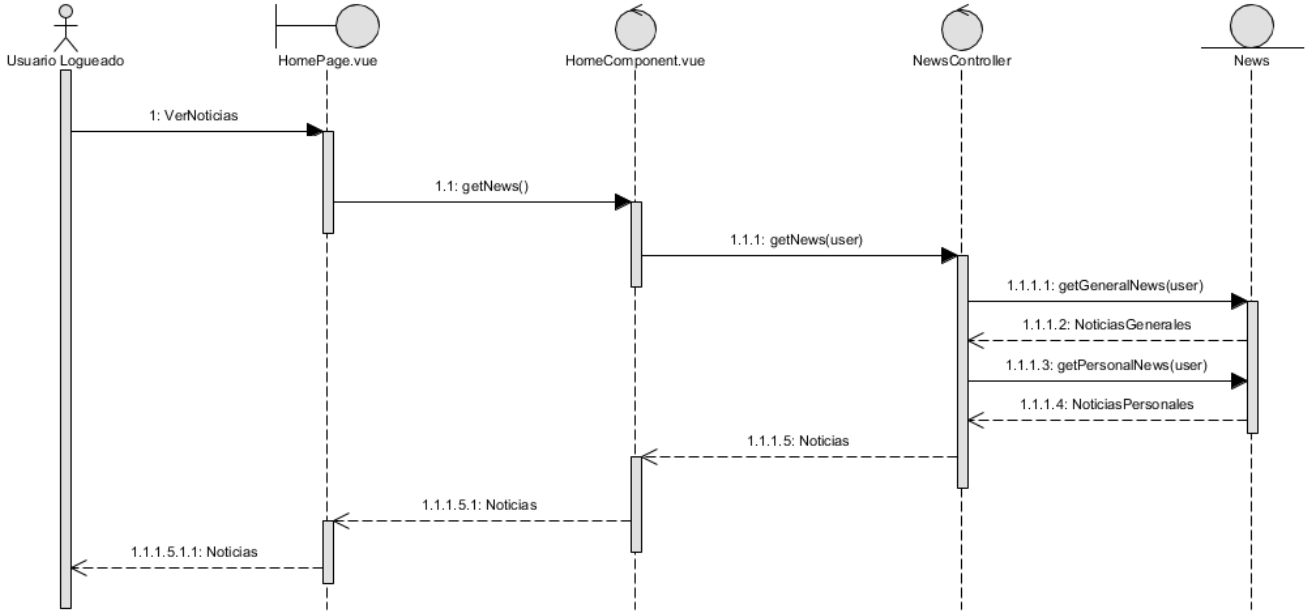


Ilustración 25: UC-0016 Ver noticias

Caso de uso 0017 **Calcular puntaje jugador por jornada**

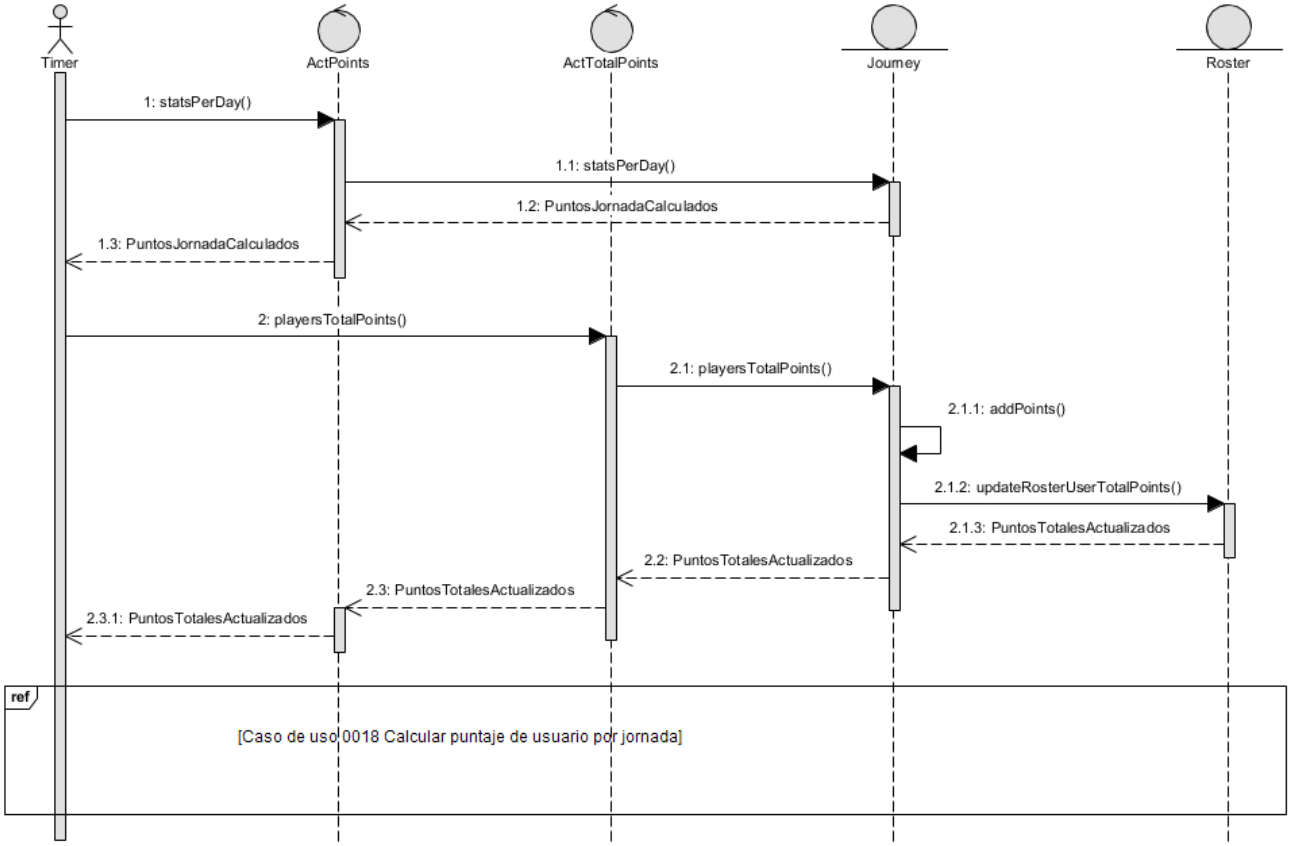


Ilustración 26: UC-0017 Calcular puntaje jugador por jornada

Caso de uso 0018 **Calcular puntaje usuario por jornada**

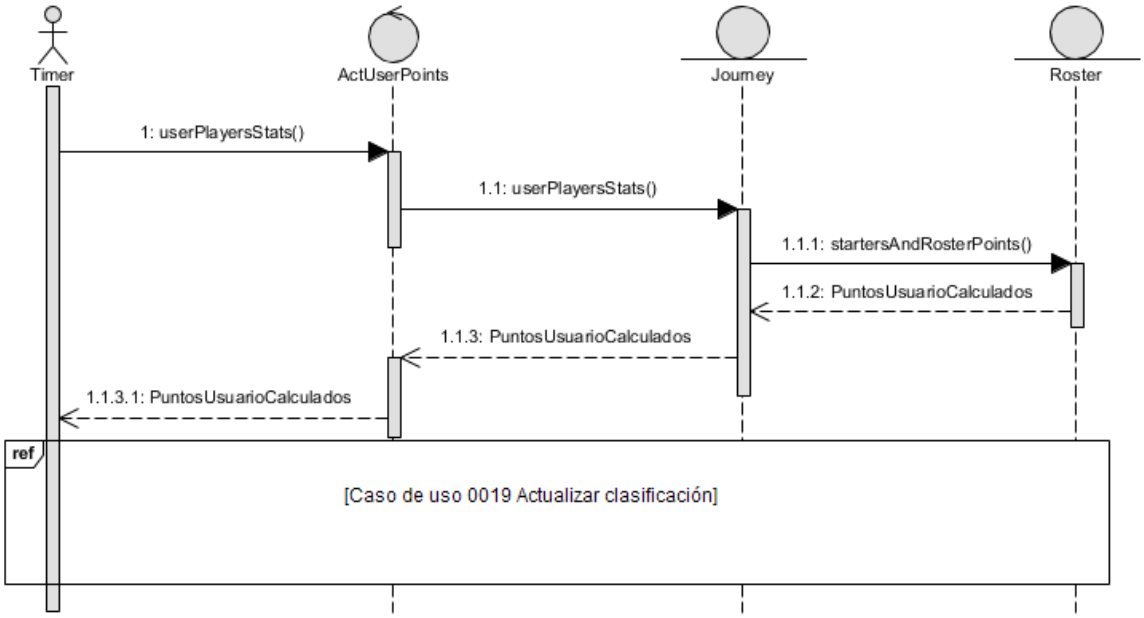


Ilustración 27: UC-0018 Calcular puntaje usuario por jornada

Caso de uso 0019 **Actualizar clasificación**

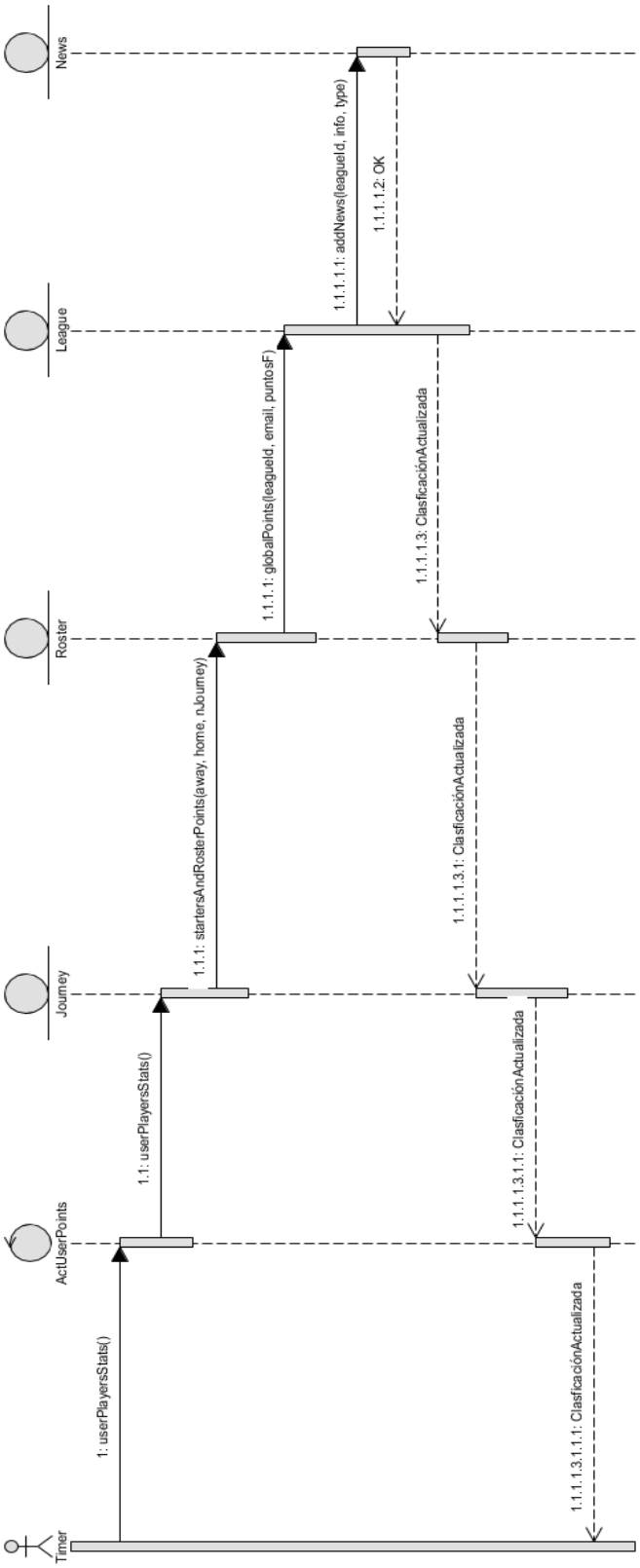


Ilustración 28: UC-0019 Actualizar clasificación

2.5.3 Diagramas de secuencia del Paquete Gestión de Mercado

Caso de uso 0020 **Vender jugador**

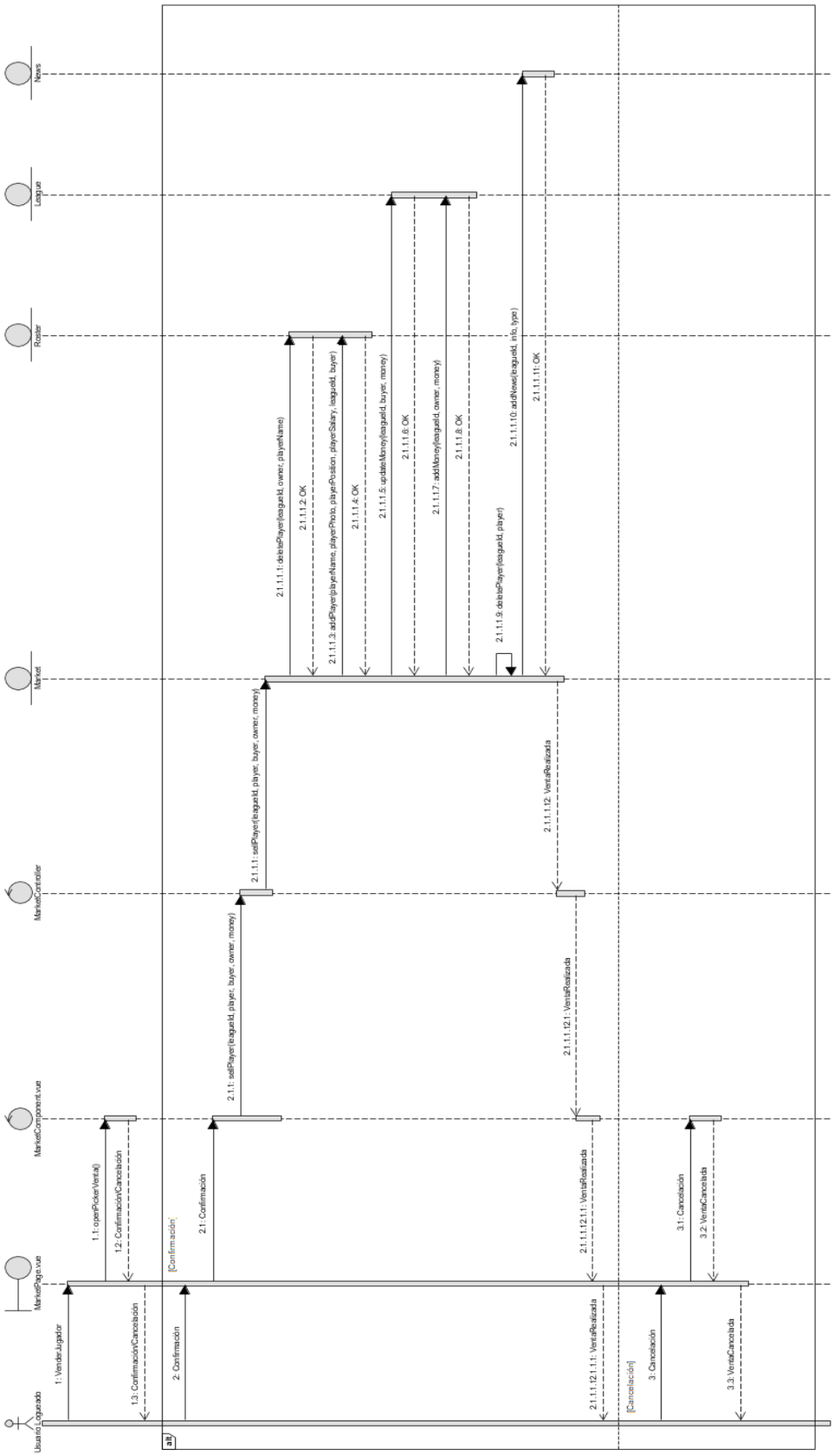


Ilustración 29: LUC-0020 Vender jugador

Caso de uso 0021 **Ver mercado**

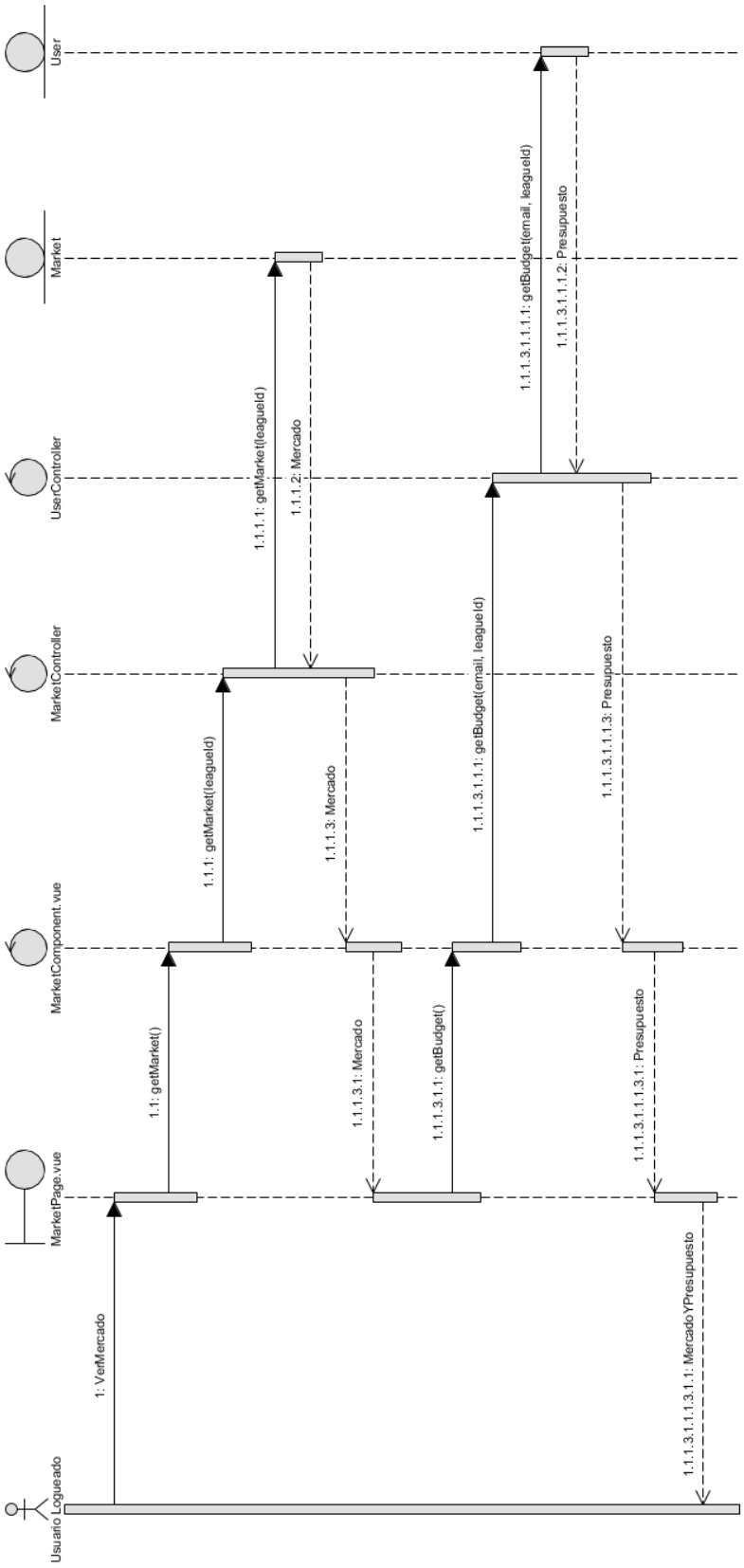


Ilustración 30: UC-0021 Ver mercado

Caso de uso 0022 **Poner jugador a la venta**

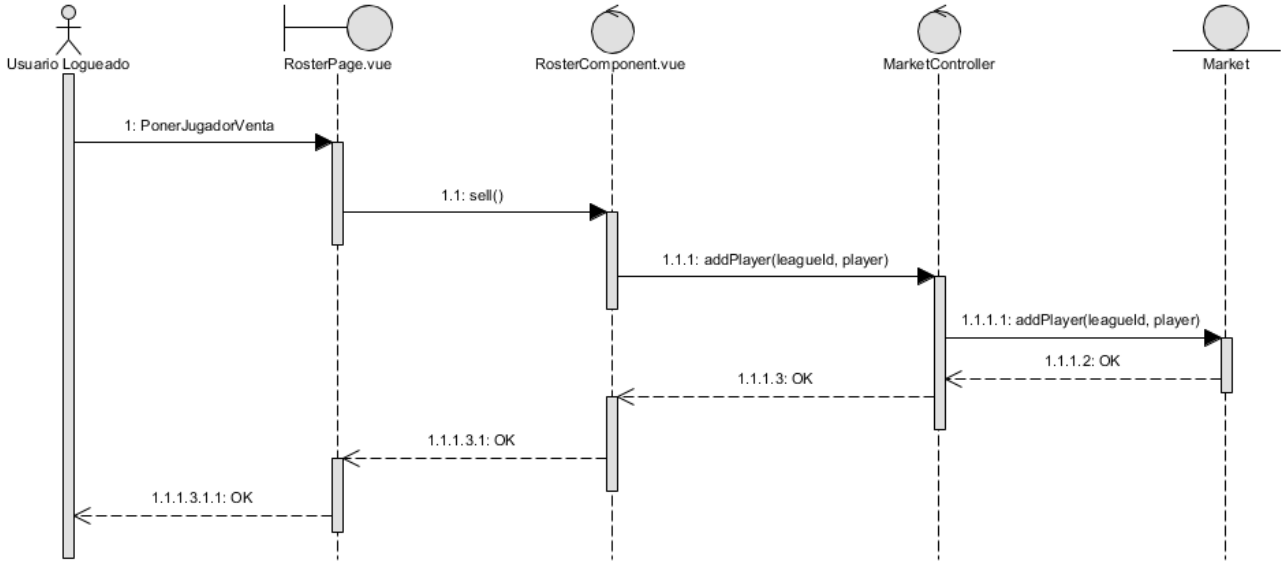


Ilustración 31: UC-0022 Poner jugador a la venta

Caso de uso 0023 **Hacer puja**

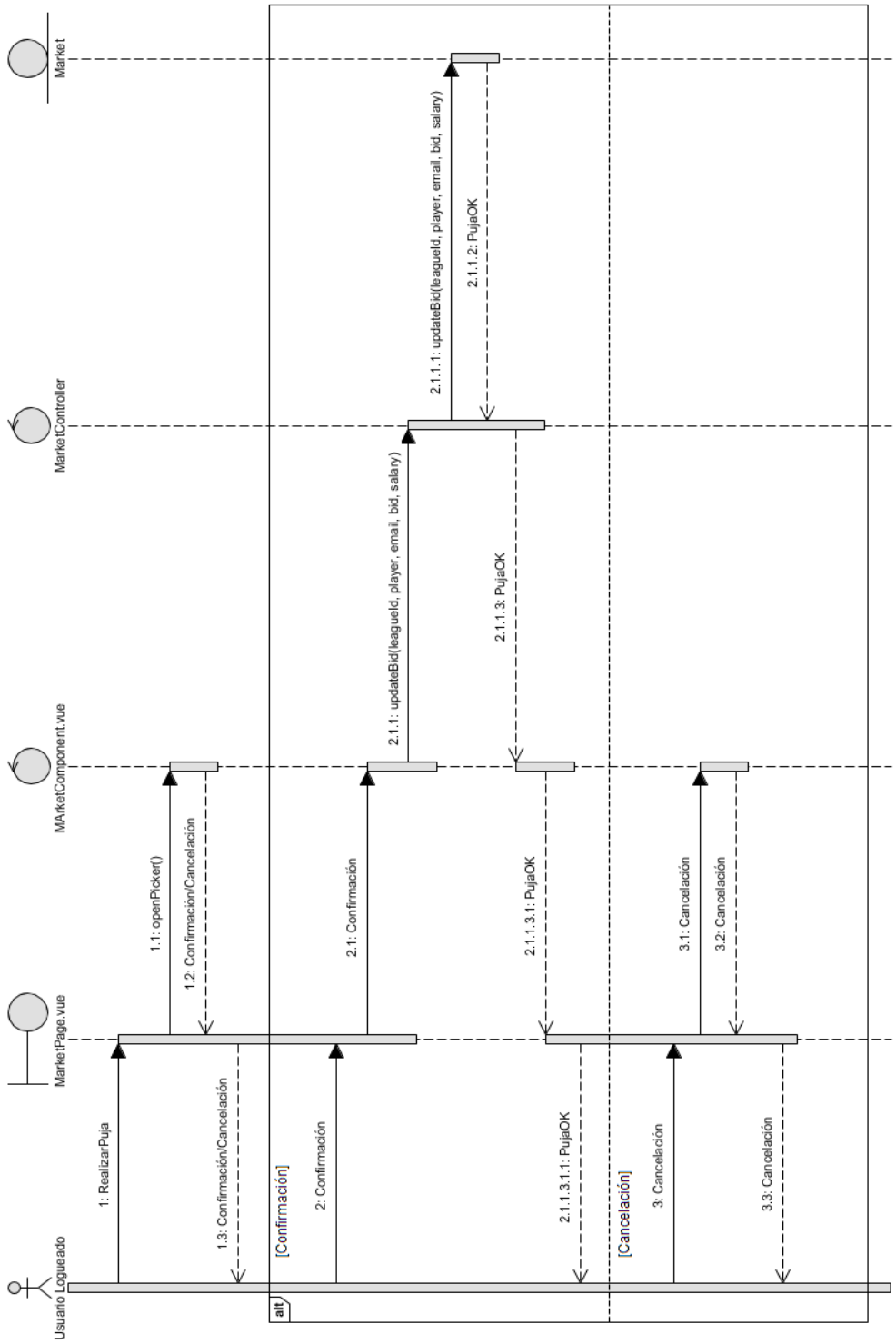


Ilustración 32: UC-0023 Hacer puja

Caso de uso 0024 **Modificar puja**

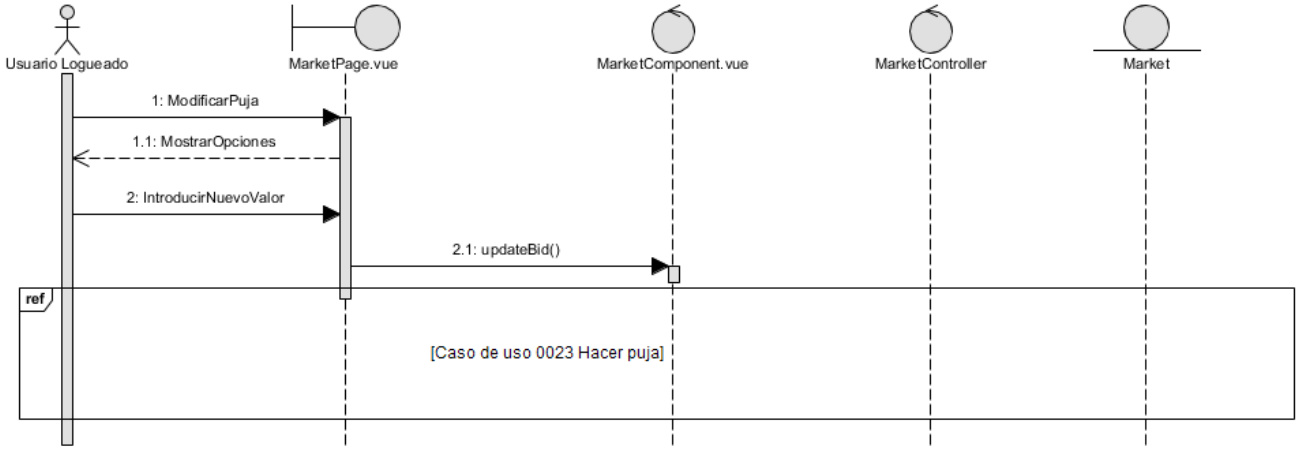


Ilustración 33: UC-0024 Modificar puja

Caso de uso 0025 **Eliminar puja**

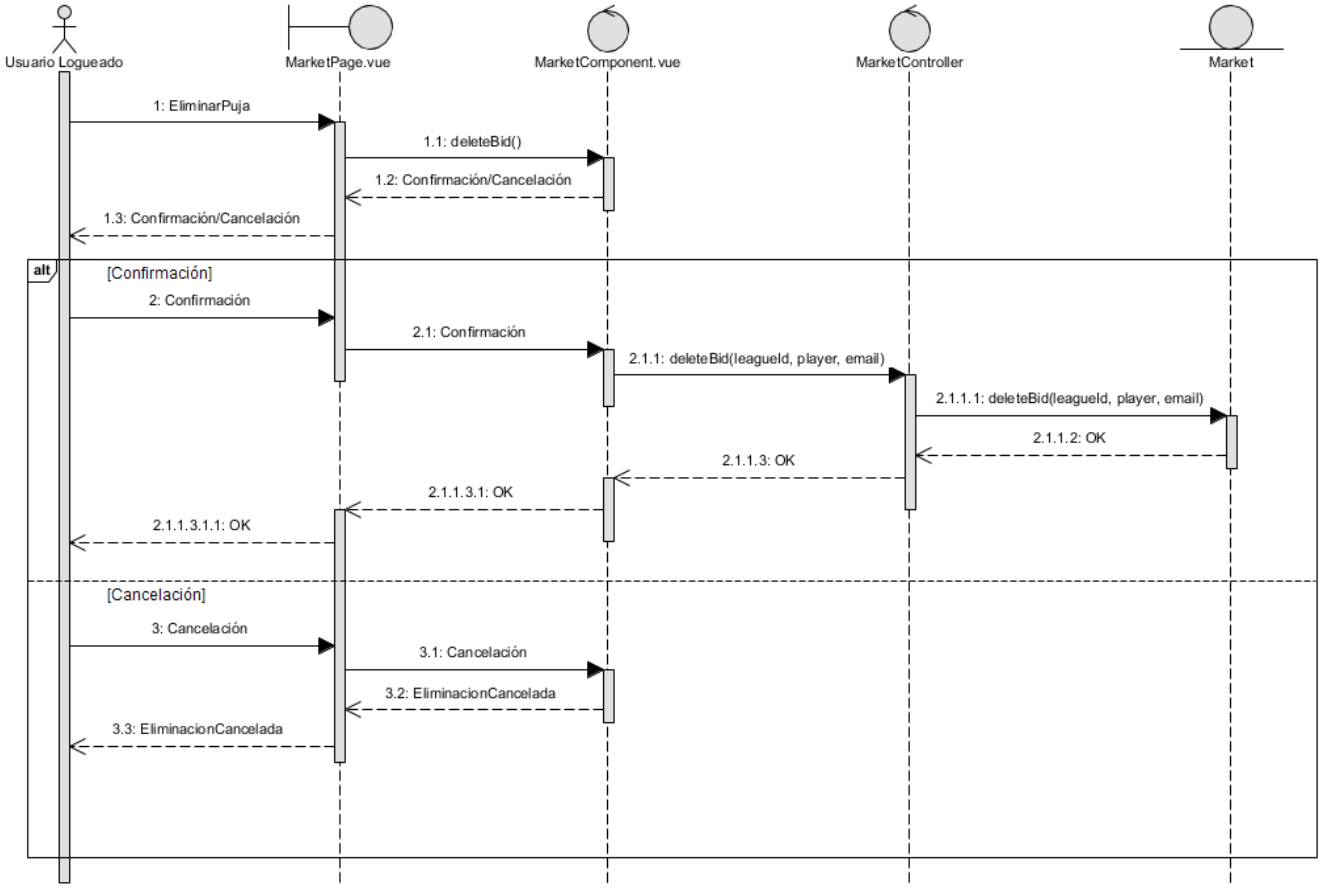


Ilustración 34: UC-0025 Eliminar puja

Caso de uso 0026 **Ver mercado**

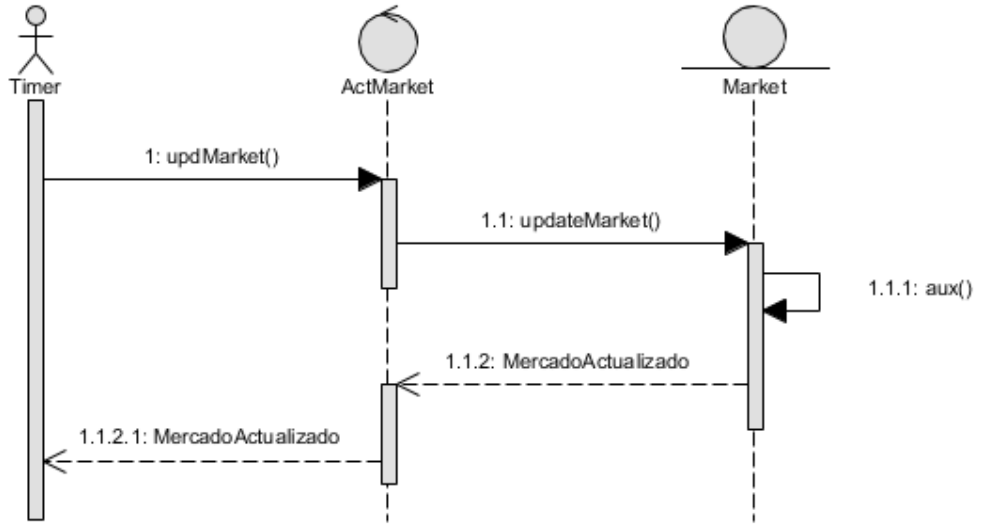


Ilustración 35: UC-0026 Ver mercado

2.5.4 Diagramas de secuencia del Paquete Gestión de Roster

Caso de uso 0027 **Ver equipo de amigo**

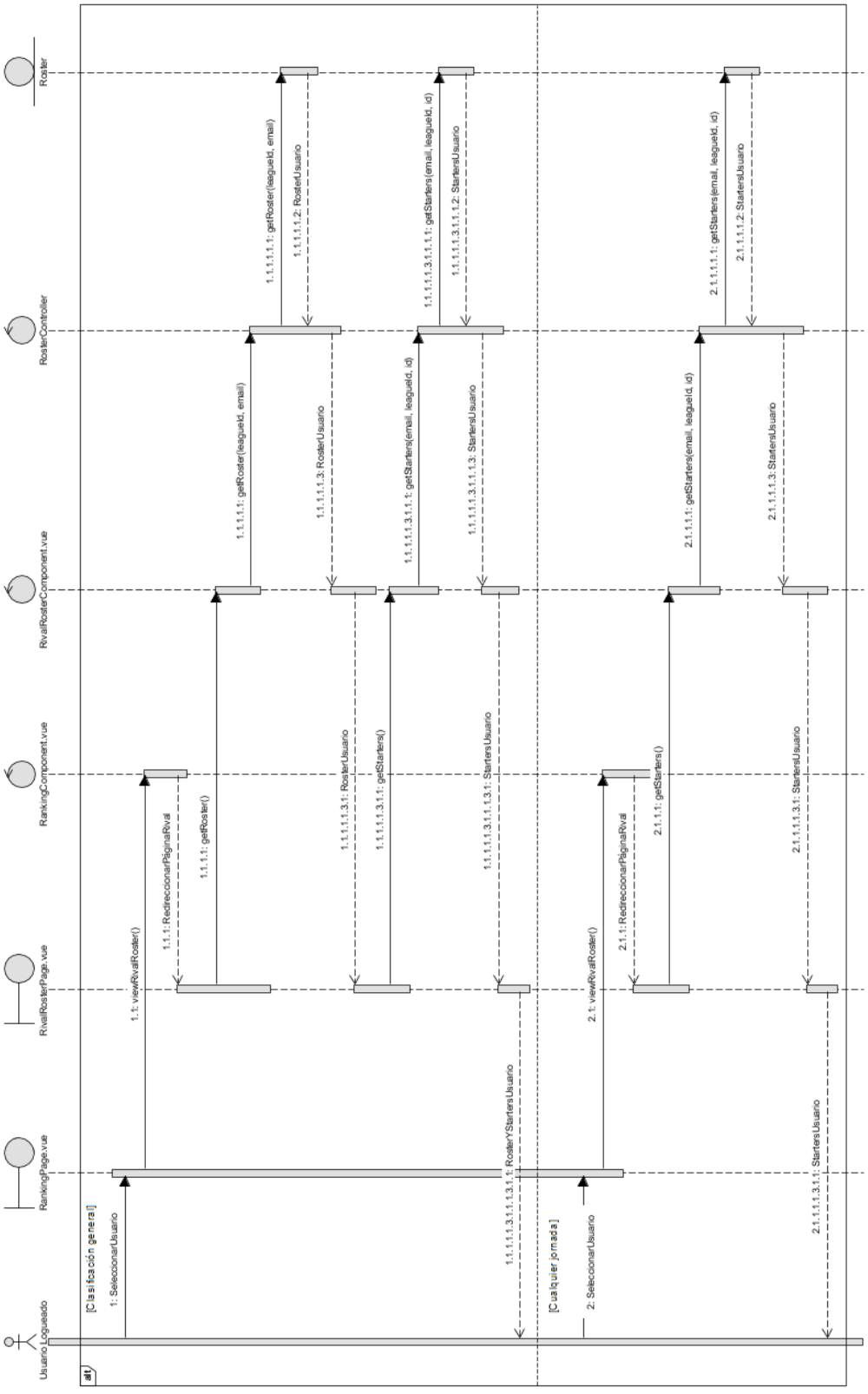


Ilustración 36: UC-0027 Ver equipo de amigo

Caso de uso 0028 **Ver equipo**

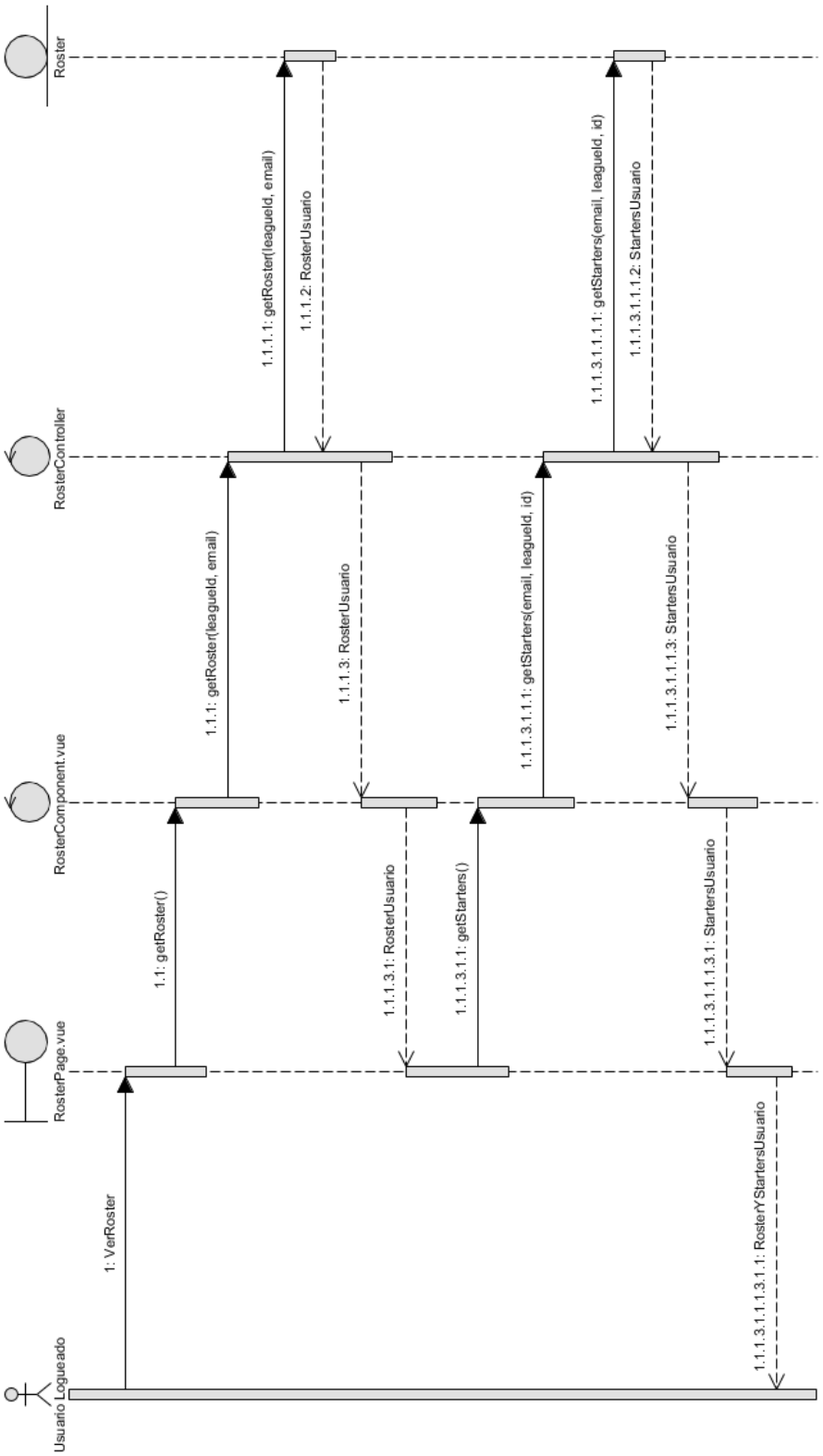


Ilustración 37: UC-0028 Ver equipo

Caso de uso 0029 **Modificar quinteto**

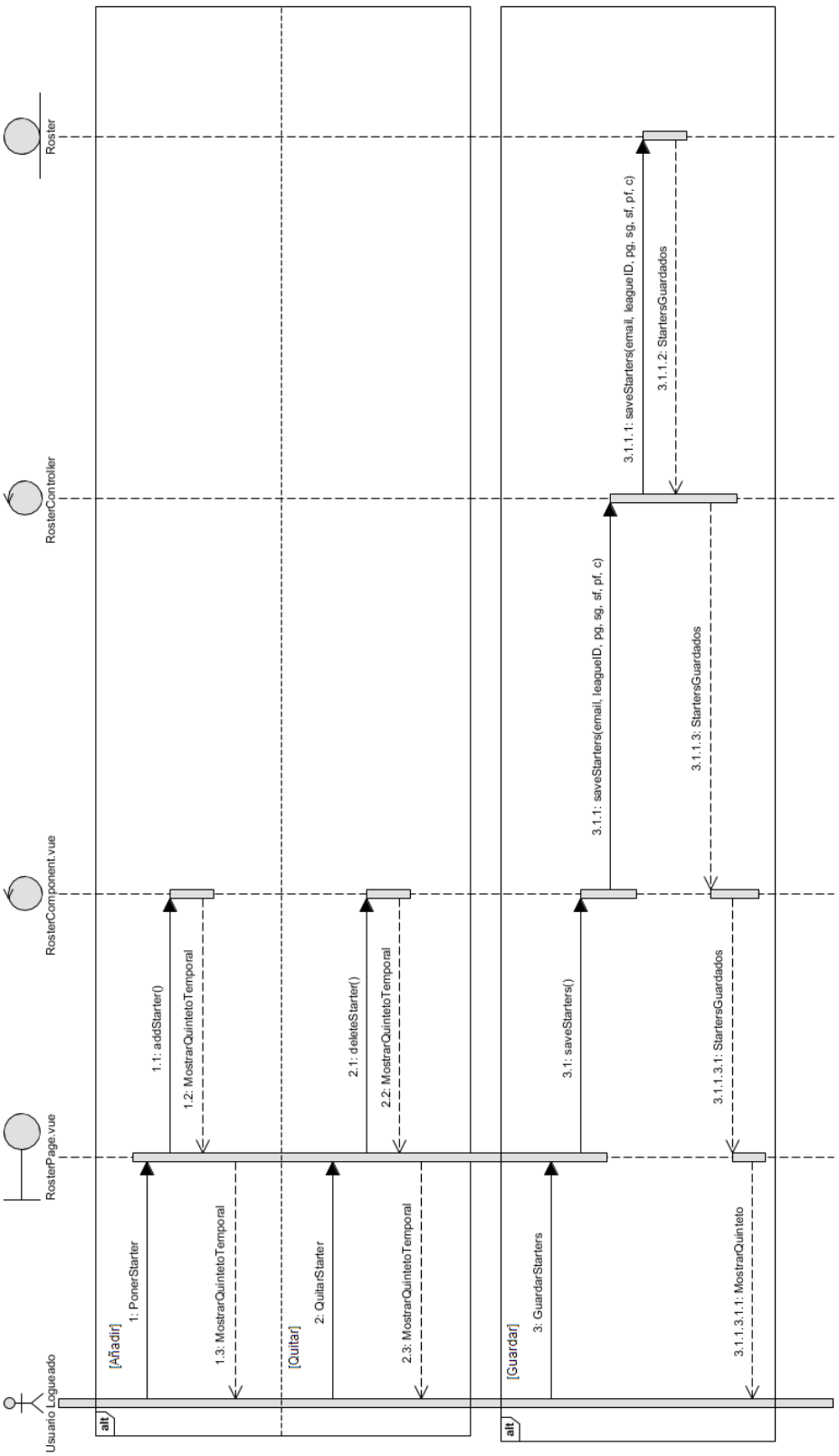


Ilustración 38: UC-0029 Modificar quinteto

Caso de uso 0030 **Actualizar roster**

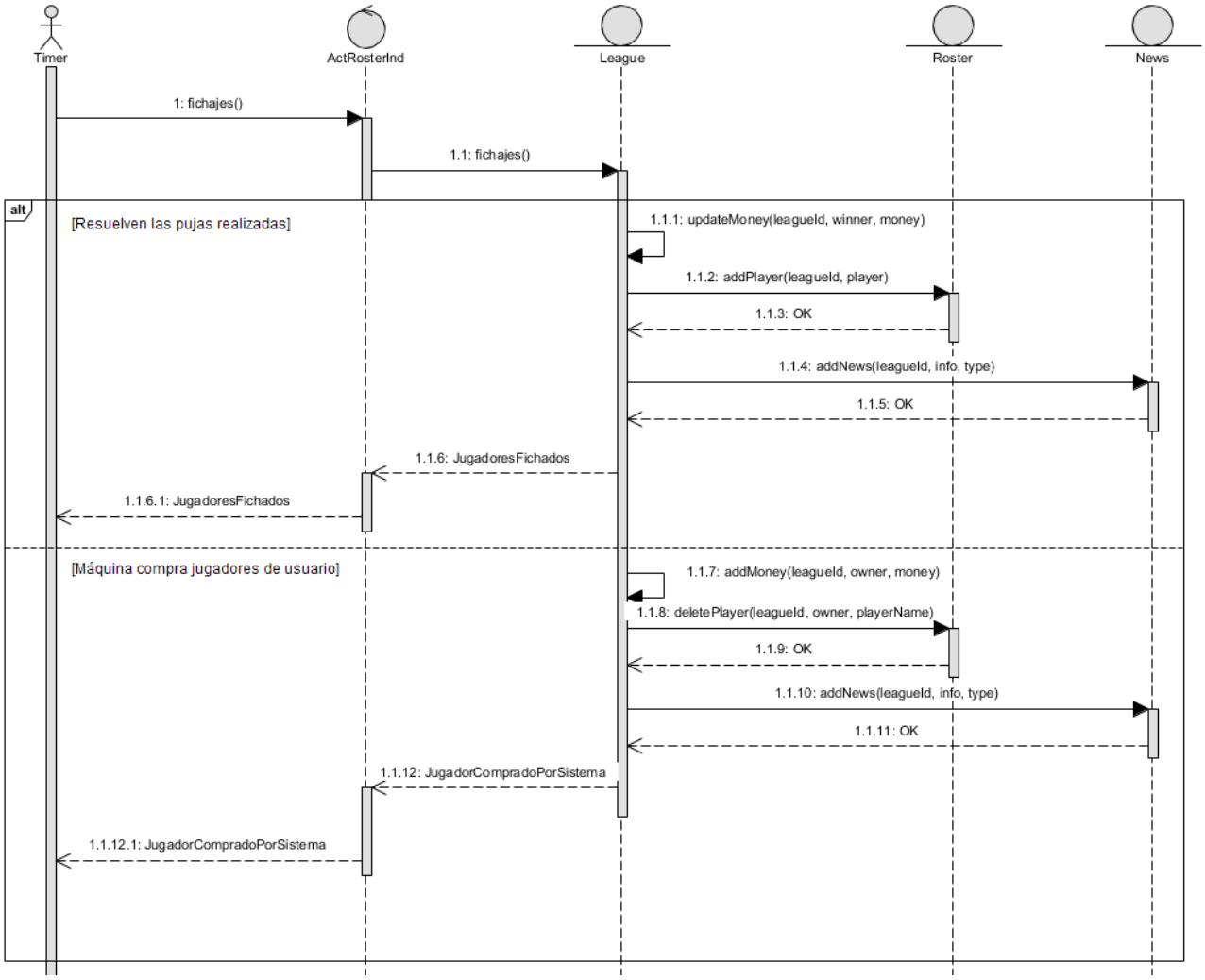


Ilustración 39: UC-0030 Actualizar roster

3. DISEÑO BASE DE DATOS

En este sistema se va a utilizar una variación de base de datos NoSQL, base de datos documental utilizada por Firebase, la cual estará formada por colecciones que contendrán documentos o subcolecciones. Los datos se conceptualizan como un árbol JSON dentro de estas colecciones y documentos.

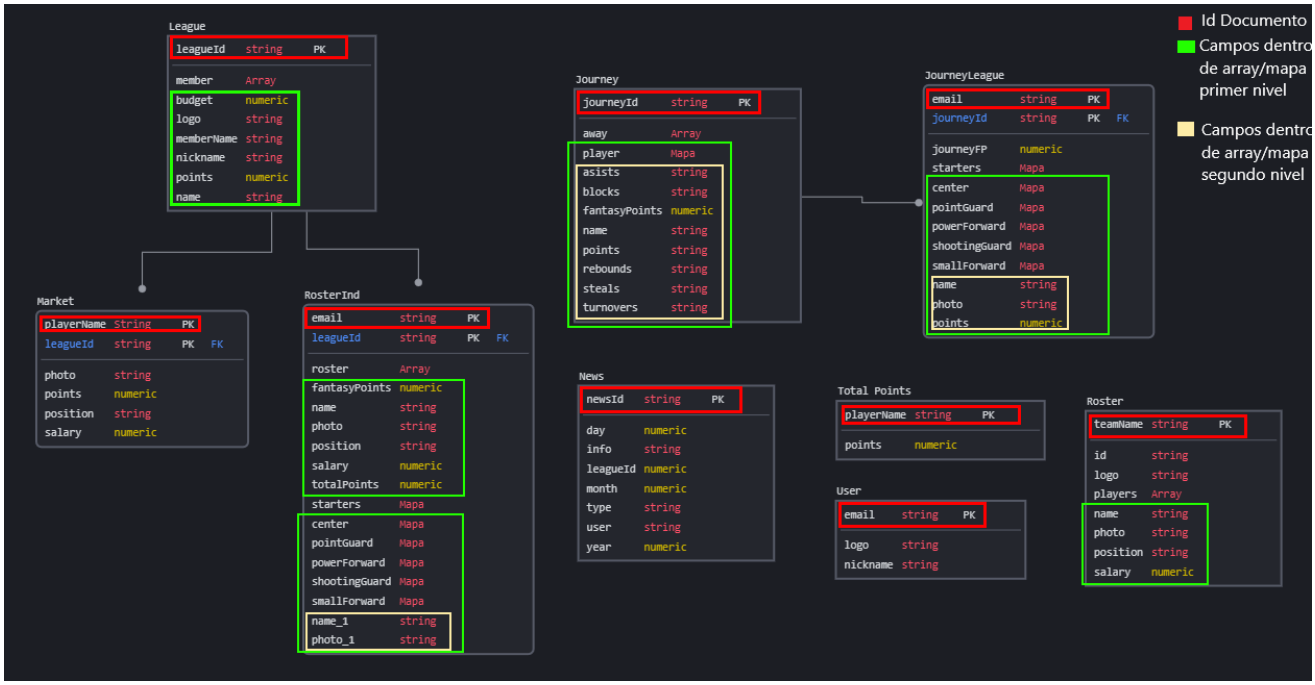


Ilustración 40: Diagrama de base de datos

Como la leyenda indica en la *Ilustración 40*, los campos resaltados en rojo hacen referencia al identificador del documento dentro de la colección. Así mismo, los campos destacados en verde y blanco indican los campos que se encuentran dentro de un mapa o un array, esto es debido a la estructura de árbol JSON comentada previamente.

Por último, la colección "*League*" tiene dos subcolecciones dentro, "*Market*" y "*RosterInd*". Como es a su vez "*JourneyLeague*" subcolección de la colección "*Journey*".

4. MODELO DE DESPLIEGUE

El modelo de despliegue permite observar la arquitectura de ejecución del sistema, esta arquitectura incluye los distintos nodos hardware y software, así como el middleware que conecta estos nodos. Para ello se realiza un diagrama de despliegue, *Ilustración 40*, que incluye los siguientes nodos:

- **Ordenador/Dispositivo móvil:** Será utilizado por los usuarios, es decir, es el nodo cliente. Pueden existir varios de estos nodos en función del número de usuarios que estén utilizando la aplicación. El entorno de ejecución donde se desplegará el artefacto aplicación será el navegador web. Mantiene la comunicación con el nodo "*Servidor*" explicado a continuación.
- **Servidor:** Da soporte de datos y funcionalidad al nodo anterior. Solamente existe un nodo de este tipo, el cual se ejecutará sobre node.js. Además de la comunicación con el

nodo "*Ordenador/Dispositivo móvil*", debe tener a su vez comunicación con el servidor de Firebase, representado en otro nodo, para obtener ciertos servicios.

- **Servidor Firebase:** Ofrece servicios de autenticación y base de datos al servidor de la aplicación, con el cual mantiene comunicación como se explicó anteriormente.

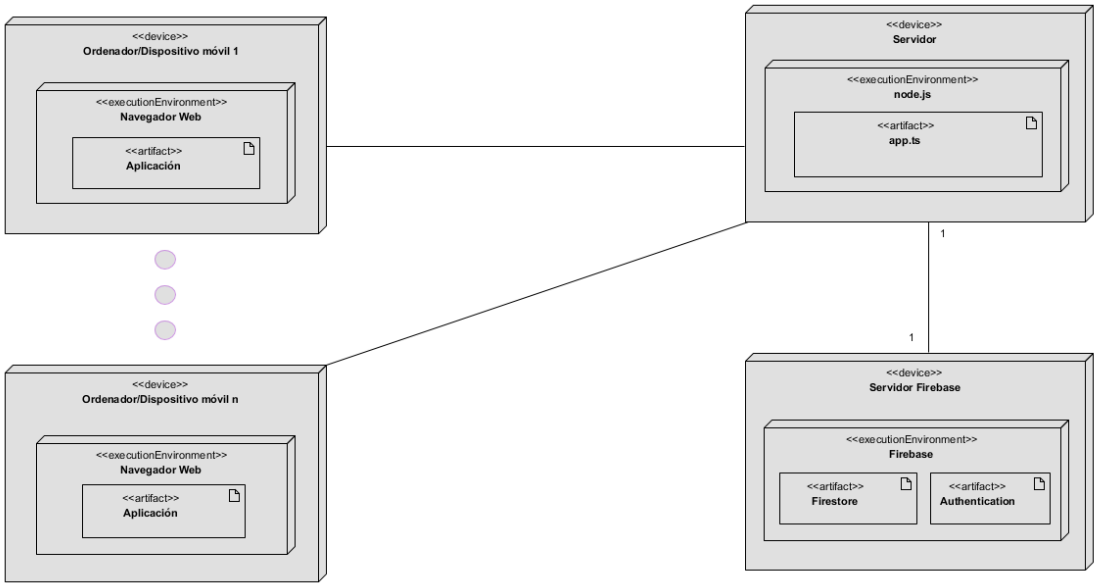


Ilustración 41: Diagrama de despliegue

La comunicación entre los diferentes nodos se hace de manera distinta en función de los nodos que participen en esta. Entre los nodos "*Ordenador/Dispositivo móvil"* y el nodo "*Servidor"*, la comunicación se establece mediante peticiones HTTP desde el cliente hacia el servidor, debido a que este proporciona una API REST.

Por otro lado, la comunicación entre servidor y servidor de Firebase se realiza mediante las interfaces que proporciona el propio Firebase.

5. BIBLIOGRAFÍA

Delgado, D. (3 de 12 de 2014). *Arquitectura de software*. Obtenido de http://arquitecturasomos4.blogspot.com/

IBM. (05 de 03 de 2021). Obtenido de El modelo de diseño: https://www.ibm.com/docs/es/rsm/7.5.0?topic=model-design

Microsoft. (18 de 03 de 2022). Obtenido de https://docs.microsoft.com/es-es/projectonline/project-online

Moreno García, M. N., & J., G. P. (2020). *Trasparencias Ingeniería del Software II Tema 3, Patrones.*

Visual Paradigm. (s.f.). Obtenido de https://www.visual-paradigm.com/

Wikipedia. (08 de 06 de 2022). Obtenido de https://en.wikipedia.org/wiki/Software_design_pattern

ANEXO V: Documentación técnica

Dream NBA: aplicación para la creación de ligas virtuales y gestión de un equipo propio

Trabajo de Fin de Grado

INGENIERÍA INFORMÁTICA

VNiVERSiDAD D SALAMANCA

CAMPUS DE EXCELENCIA INTERNACIONAL

Julio 2022

Autor

Jairo Sánchez García

Tutores

Gabriel Villarrubia González

Juan Francisco de Paz Santana

Héctor Sánchez San Blas

TABLA DE CONTENIDO

1. INTRODUCCIÓN

El propósito principal de este documento es el de ayudar a personas externas al proyecto a la comprensión del código fuente y de la estructura de la aplicación.

El sistema se compone de dos partes diferenciadas:

- Aplicación web.
- Servidor.

A continuación, se explicarán los componentes que dan forma a cada una de las partes del sistema describiendo la función que cumplen dentro de este.

2. ESCTRUCTURA DEL SISTEMA

En este apartado se realizará un desglose de las carpetas que organizan la estructura de los dos subsistemas. Dentro de las carpetas se encuentran los ficheros de código fuente correspondientes a cada tarea.

2.1 Servidor

La carpeta raíz de la parte del servidor es "*./server*", a partir de esta se pueden encontrar varias más que se van a explicar seguidamente:

- **./docs**: contiene los archivos correspondientes a la documentación de los ficheros del código generados con la herramienta Typedoc.
- **./node_modules:** se encuentran los módulos utilizados para el desarrollo de la aplicación.
- **./src:** dentro de este directorio se encuentran todos los ficheros de código, subdivididos en directorios según su funcionalidad.
    - **./src/controller:** contiene todos los archivos controlador. En estos están implementadas las acciones a realizar por el servidor según la petición que recibe, es decir, se implementan las rutas para las peticiones. Están divididos según la funcionalidad y datos que se quiere aportar.
    - **./src/models:** abarca los archivos modelo, donde se definen que datos utilizará el sistema y la comunicación con la base de datos.
    - **./src/scripts:** se encuentran los scripts que implementan una tarea periódica a través de Cron, es decir, estos scripts se ejecutarán de manera semanal o diaria según sea su funcionalidad.
    - **./src/app.ts:** es el archivo principal del servidor en donde se inicializa la aplicación y se montan los middlewares necesarios.
    - **./src/firebaseAdmin.ts** y **./src/firebaseConf.ts:** son los ficheros de configuración de Firebase para poder iniciar sus aplicaciones.

Dentro del servidor se encuentran, además de los expuestos, los archivos de configuración de la aplicación y archivos que contienen las variables de entorno.

2.2 Aplicación web

El grueso de esta parte se encuentra en el directorio "*./src*", el cual contiene diversos subdirectorios agrupados por funcionalidad. Además de esta carpeta, se tienen los archivos de configuración correspondientes a este subsistema, como el archivo "*package.json*" que contiene metadatos del proyecto, los paquetes node instalados y la lista de dependencias. O el fichero "*tsconfig.json*" donde se especifican los ficheros raíz y las opciones de compilación requeridas, estos dos ficheros son, junto algún otro, los más importantes ya que proporcionan la correcta configuración para el funcionamiento y ejecución de la aplicación.

- **./src/components:** este directorio comprende los componentes Vue que se van a utilizar en la aplicación.
- **./src/router:** el cual contiene el archivo "*index.ts"* , en donde se definen las rutas de la aplicación y las vistas asociadas a cada ruta.
- **./src/theme:** contiene algunos archivos multimedia utilizados y el fichero css con la paleta de color de la aplicación.
- **./src/views:** corresponden a las vistas de la aplicación, dependiendo de la necesidad utilizarán los componentes explicados previamente.

- **./src/App.vue:** es la ruta raíz de la aplicación, a partir de esta se formarán las demás rutas y vistas del sistema.
- **./src/main.ts:** es el fichero en cual se crea y se monta la aplicación Vue. Se monta en la ruta raíz y se inicia desde este archivo el enrutado.

3. ESPECIFICACIÓN DE LA DOCUMENTACIÓN

Para realizar la documentación técnica del sistema se ha utilizado *Typedoc* en la parte del servidor ya que el lenguaje utilizado ha sido *TypeScript*. En la aplicación web, hecha con el framework *Ionic* y *Vue*, se ha realizado un trabajo de comentario sobre el código.

La herramienta *Typedoc* proporciona una manera clara y organizada de documentar el código generando un archivo HTML, donde se especifica el nombre del archivo, clases, métodos y atributos que conforman esta clase.

3.1 Servidor

Como se explica en los párrafos superiores, la documentación técnica ha sido generada a través de la herramienta *Typedoc.* Esta documentación se encuentra en *server/docs/index.html.*

3.2 Aplicación web

La documentación técnica para los archivos .vue que forman este subsistema se ha realizado mediante comentarios en el código. Por lo que para ver dicha documentación se tienen que visitar los ficheros .vue de los directorios *src/components* y *src/views.*

4. BIBLIOGRAFÍA

Ionic Framework. Obtenido de https://ionicframework.com/docs/components

Typedoc. Obtenido de http://typedoc.org/

TypeScript. Obtenido de https://www.typescriptlang.org/

Vue Styleguidist. Obtenido de https://vue-styleguidist.github.io/docs/Documenting.html#code comments

ANEXO VI: Manual de usuario

Dream NBA: aplicación para la creación de ligas virtuales y gestión de un equipo propio

Trabajo de Fin de Grado

INGENIERÍA INFORMÁTICA

VNiVERSiDAD D SALAMANCA

CAMPUS DE EXCELENCIA INTERNACIONAL

Julio 2022

Autor

Jairo Sánchez García

Tutores

Gabriel Villarrubia González

Juan Francisco de Paz Santana

Héctor Sánchez San Blas

TABLA CONTENIDO

TABLA DE ILUSTRACIONES

1. INTRODUCCIÓN

El objetivo principal de este anexo es ilustrar a los usuarios sobre el funcionamiento del sistema y la correcta interacción con sus elementos.

Estará compuesto por un único apartado y un subapartado por sección; enumeradas a continuación:

- Login.
- Registro.
- Página home.
- Menú lateral
    - Perfil.
    - Crear liga.
    - Añadir liga.
    - Ligas.
    - Logout.
- Clasificación.
    - Roster de usuarios.
- Roster.
- Mercado.
- Calendario.
    - Estadísticas.

2. MANUAL APLICACIÓN

A continuación, se detallará sección a sección el modo de funcionamiento del sistema y como el usuario debe interaccionar con él correctamente.

2.1 Pantalla de login

La pantalla de login, *Ilustración 1*, es la primera que se ve al acceder a la aplicación. Está compuesta por dos campos de entrada, uno para el correo y otro para la contraseña, respectivamente; un botón para iniciar sesión, otro para acceder a la pantalla de registro y un tercer campo para la recuperación de la contraseña, se puede apreciar en la *Ilustración 2*.

El tipo de datos que se debe introducir en los campos es el siguiente:

- Campo '*Email',* tanto para el campo de login como para el de recuperación de contraseña: email, con el formato email@compañía.extensión
- Campo '*Password*': texto o números, depende de la contraseña utilizada.

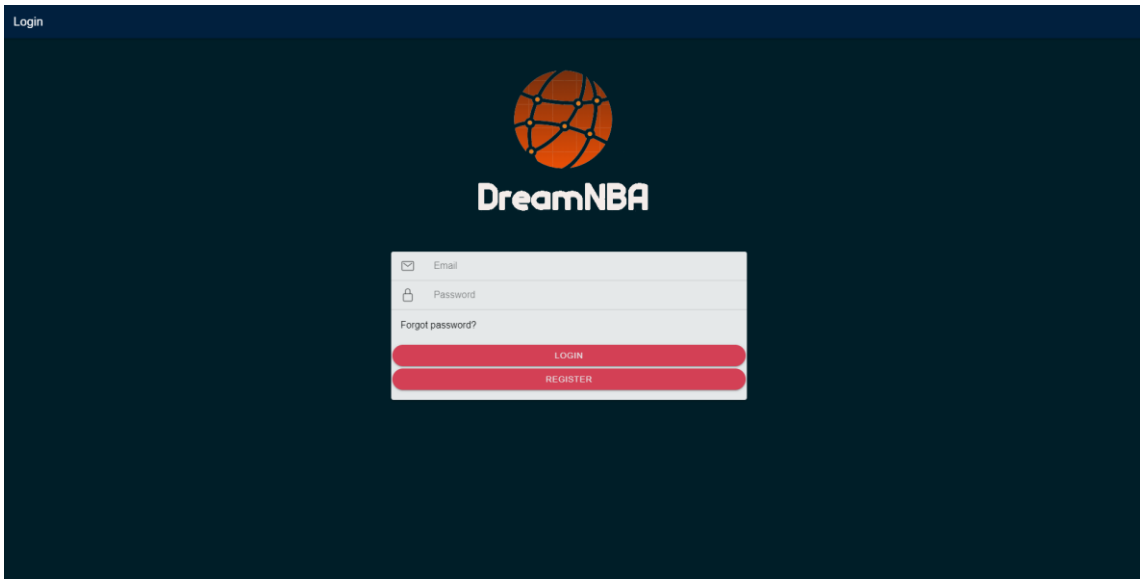


Ilustración 1: Pantalla Login

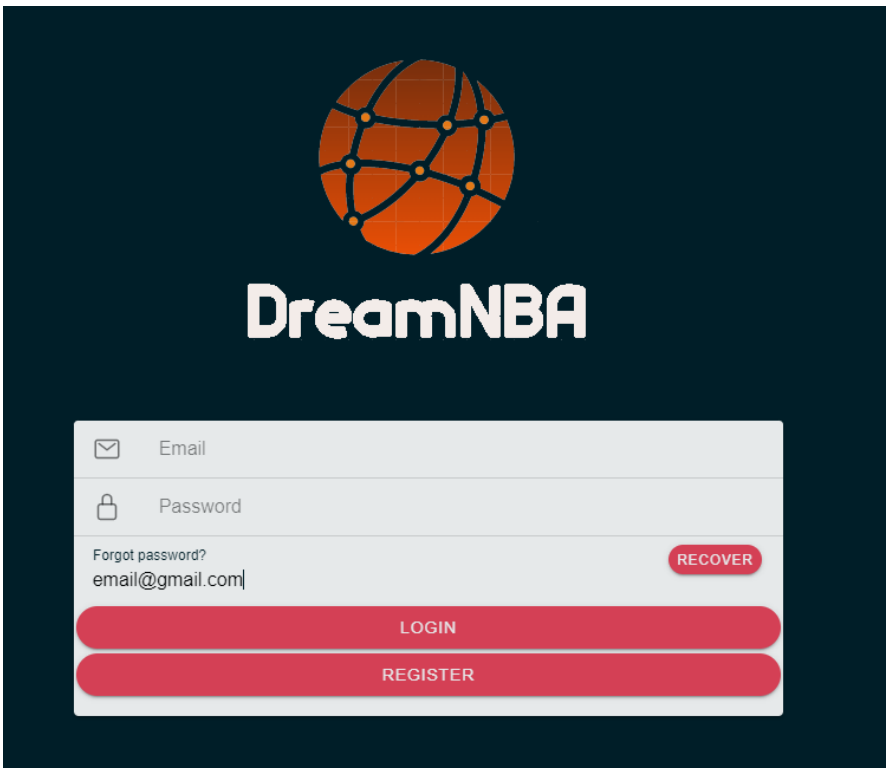


Ilustración 2: Recuperar contraseña

Se realizan las siguientes comprobaciones, que muestran los siguientes errores:

- Mal formato del email, *Ilustración 3*.
- Usuario no registrado, *Ilustración 5*.
- Contraseña incorrecta, *Ilustración 4*.

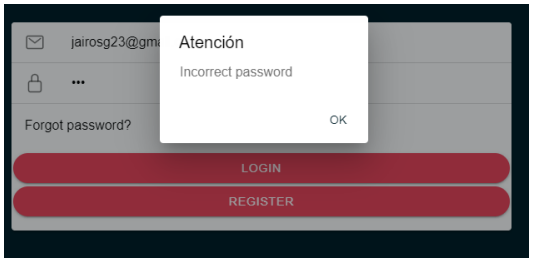


Ilustración 4: Pantalla Login, contraseña incorrecta

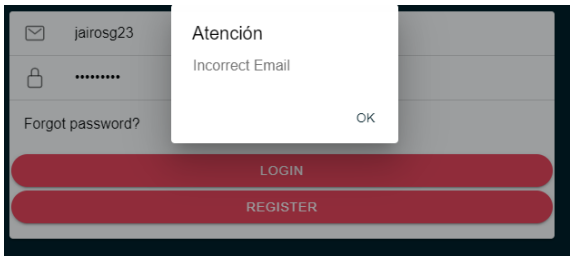


Ilustración 3: Pantalla Login, formato erróneo email

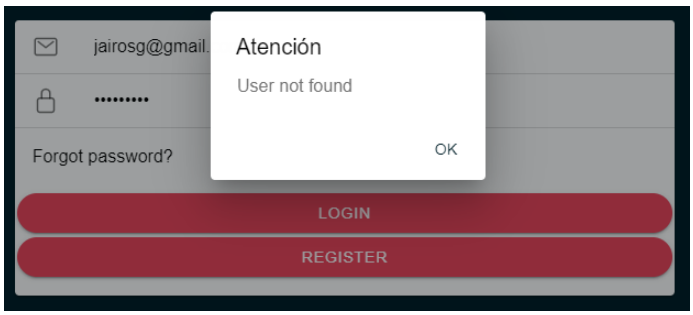


Ilustración 5: Pantalla Login, usuario no registrado

2.2 Pantalla de registro

A esta pantalla, *Ilustración 6*, se accede tras pulsar el botón '*Register*' desde la pantalla vista en el subapartado anterior.

Se tienen tres campos, uno para introducir el email, otro para el nombre de usuario y por último un campo para la contraseña. Además de estos campos, se tiene una colección con logos para seleccionar uno y el botón de registro.

El tipo de datos que se debe introducir en los campos es el siguiente:

- Campo '*Email*': email, con el formato email@compañía.extensión
- Campo '*Nickname*': texto.
- Campo '*Password*': texto o números, depende de la contraseña utilizada.

Se realizan una serie de comprobaciones que muestran los siguientes errores:

- Mal formato del email, *Ilustración 7*.
- Usuario ya existente, *Ilustración 8*.
- Algún campo vacío, *Ilustración 9.*

Para terminar con esta pantalla, se puede retornar hacia la página de login a través de un botón en la cabecera.

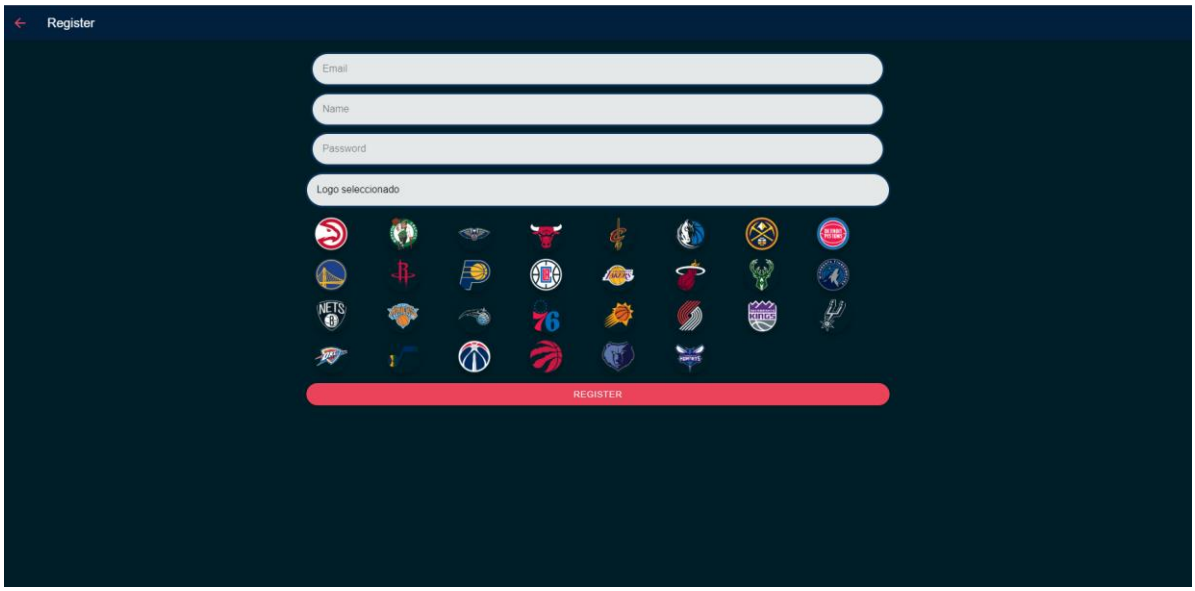


Ilustración 6: Pantalla Registro

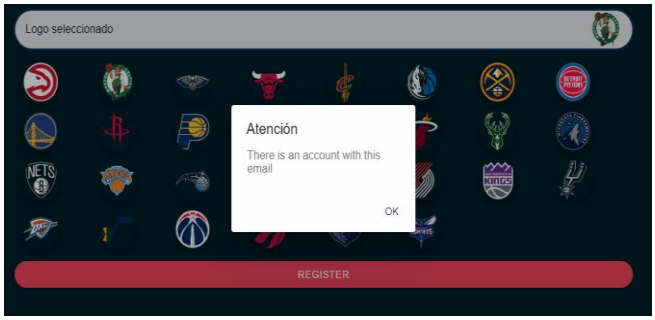


Ilustración 8: Pantalla Registro, usuario ya registrado

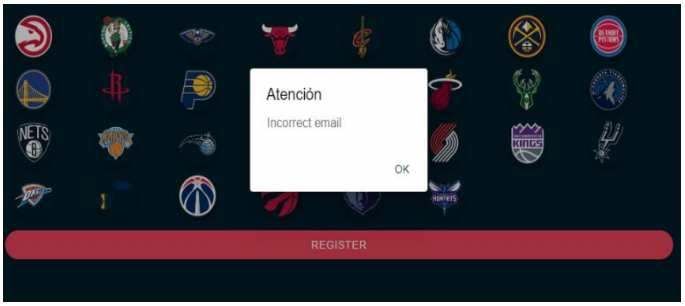


Ilustración 7: Pantalla Registro, formato de email erróneo

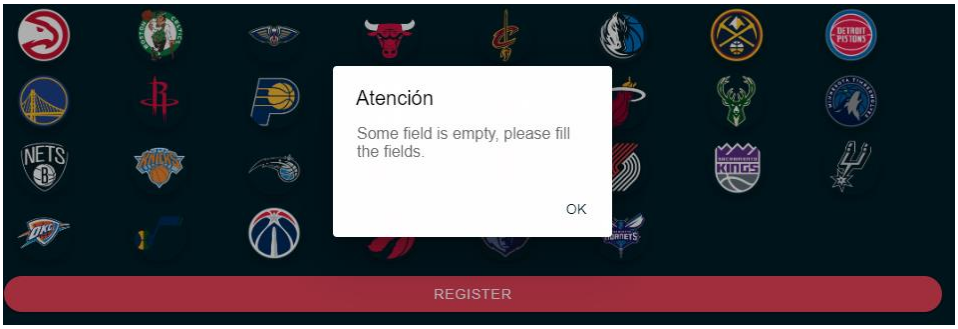


Ilustración 9: Pantalla Registro, campos vacíos

2.3 Pantalla inicial

Una vez se inicia sesión, la pantalla que se visualiza es la que se observa en la *Ilustración 10*, donde se muestran las noticias, tanto generales como personales, relacionadas con las ligas en las que se participa.

A parte de las noticias, en la cabecera se tiene un botón para abrir el menú lateral cuyas opciones se explicarán en posteriores puntos. En esta cabecera además del menú lateral, aparece el logo del sistema, el cual permite regresar a esta pantalla desde cualquier ventana de la aplicación.

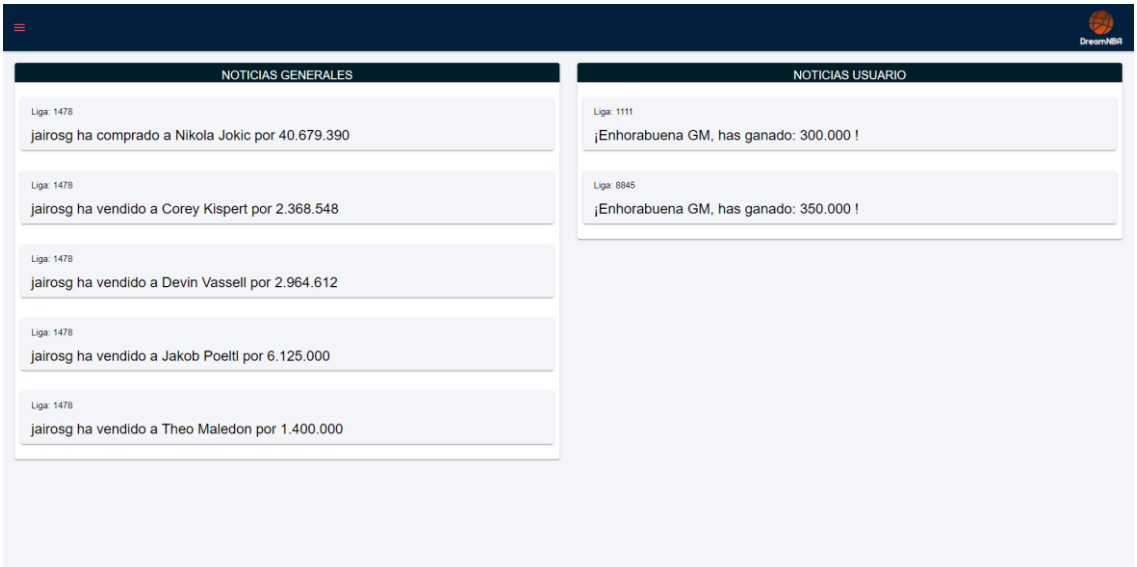


Ilustración 10: Pantalla inicial

2.4 Menú lateral

Se puede acceder a este menú desde la mayoría de las pantallas de la aplicación a través de un botón situado en la esquina izquierda de la cabecera. Dentro del menú hay cuatro opciones que son, perfil, crear liga, añadir liga, ligas y logout. Por último, para cerrar el menú, simplemente se debe pulsar fuera de él.

Ilustración 11: Menú lateral

2.4.1 Pantalla perfil

Se accede a través de la opción *Profile* del menú lateral, se puede ver su posición en la *Ilustración 12*.

Ilustración 12: Menú lateral, opción Profile

Una vez dentro de esta pantalla, *Ilustración 13*, se observa el logo del usuario, junto al email y nombre de usuario. Dispone de un campo para introducir el nuevo nombre de usuario, en donde el tipo de dato para este campo debe ser un texto; de un botón para mostrar la colección de logos en caso de querer modificarlo, como se observa en la *Ilustración 14*, y por último un botón para cambiar la contraseña, el cual enviará un correo a la dirección del usuario con un enlace para introducir la nueva contraseña. A parte, se tiene un botón para guardar la información nueva, de lo contrario no se realizaría ninguna modificación.

Para terminar, con el botón situado en la esquina izquierda de la cabecera se vuelve hacia la página anterior desde donde se accedió a esta.

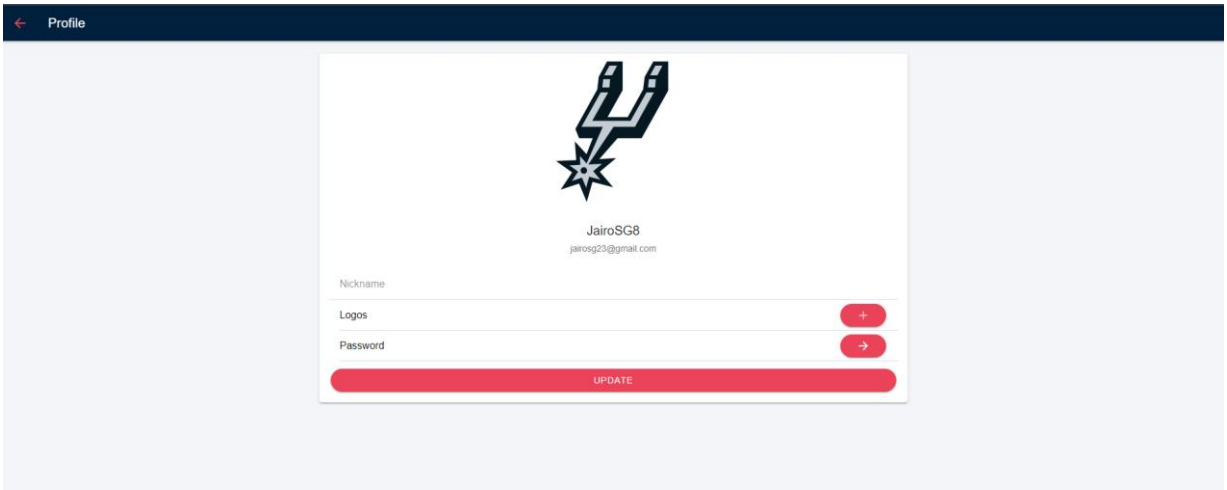


Ilustración 13: Pantalla Profile

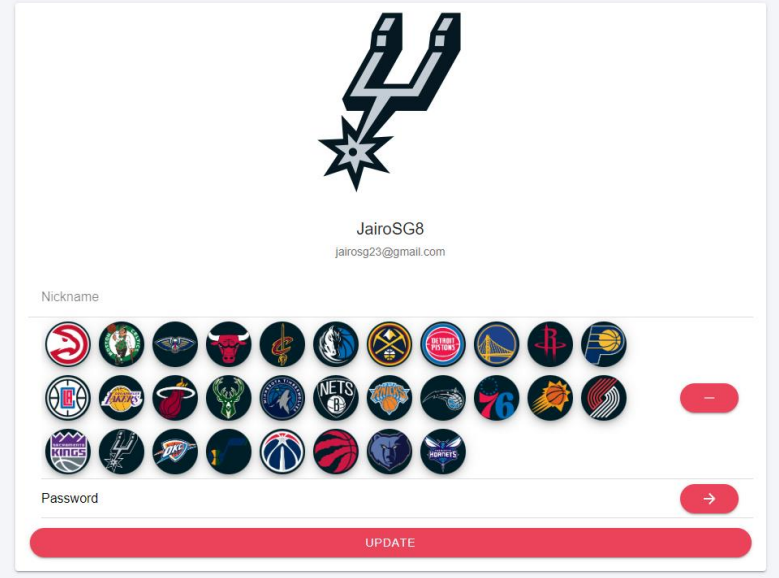


Ilustración 14: Pantalla Profile, opción cambiar logo

2.4.2 Pantalla crear liga

Se accede a través de la opción *Crear Liga* del menú lateral, se puede ver su posición en la *Ilustración 15*.

Ilustración 15: Menú lateral, opción Crear Liga

Esta pantalla, *Ilustración 16*, cuenta con dos campos, uno para introducir el código que tendrá la liga y otro para poner el nombre de esta. Por último, se tiene un botón para realizar la acción de crearla.

El tipo de datos para los campos de esta página son los siguientes:

- Campo '*Código*': numérico, longitud de 4 números.´
- Campo '*Nombre*': texto o número, dependiendo del nombre que se le quiera dar.

Los errores que se muestran son los indicados en la *Ilustración 17* e *Ilustración 18*.

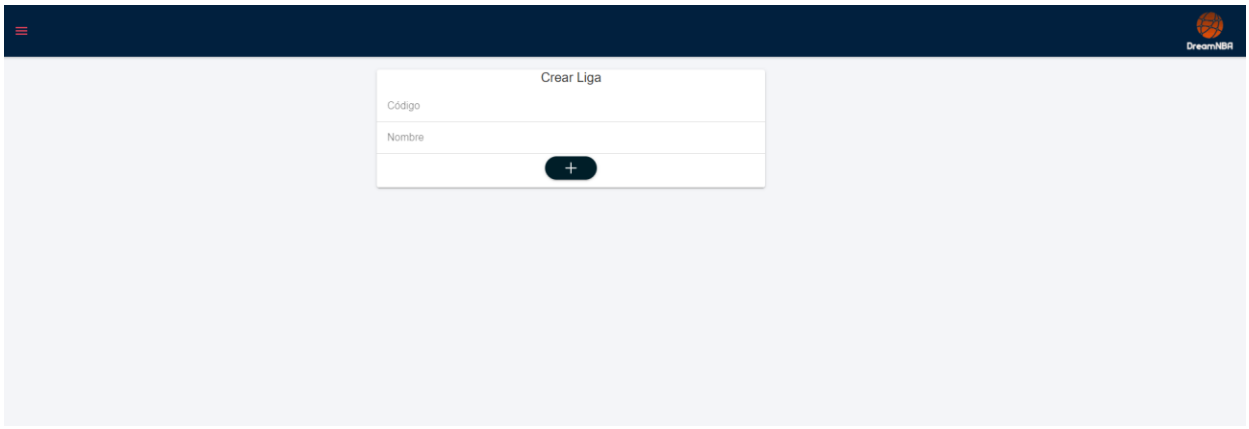


Ilustración 16: Pantalla Crear Liga

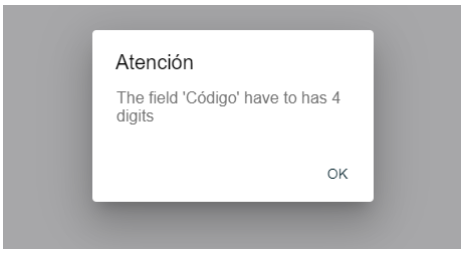


Ilustración 17: Pantalla Crear Liga, campo código menor que 4 dígitos

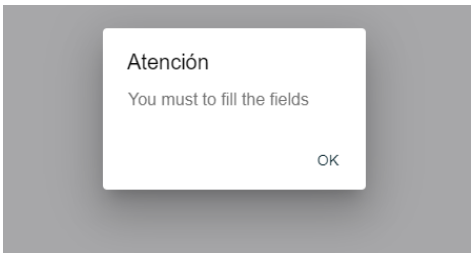


Ilustración 18: Pantalla Crear Liga, campos vacíos

2.4.3 Pantalla añadir liga

Se accede a través de la opción *Añadir Liga* del menú lateral, se puede ver su posición en la *Ilustración 19*.

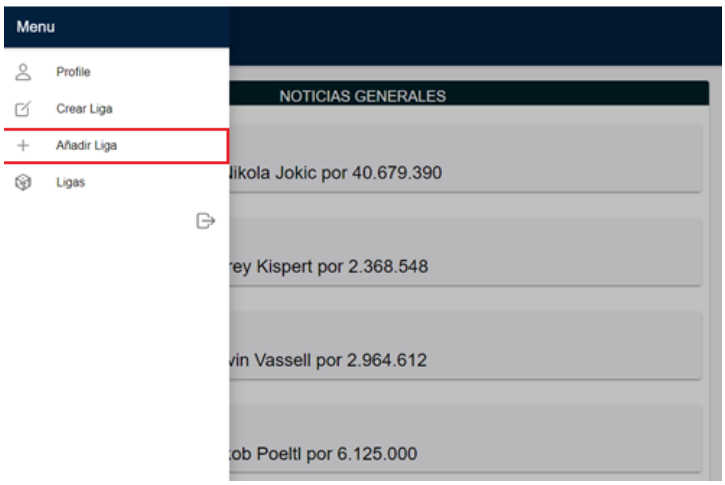


Ilustración 19: Menú lateral, opción Añadir Liga

Esta pantalla, *Ilustración 10*, solamente dispone de un campo para introducir el código de la liga a la que se quiere acceder, además de un botón para realizar la acción, el cual aparece en cuanto se ponga un dígito en el campo, el tipo de dato para este campo es numérico y se necesita una longitud mínima de 4 dígitos, de lo contrario se muestra un error como se observa en la *Ilustración 17*, es aplicable a esta pantalla.

Ilustración 20: Pantalla Añadir Liga

2.4.4 Pantalla Ligas

Se accede a través de la opción *Ligas* del menú lateral, se puede ver su posición en la *Ilustración 21*.

Ilustración 21: Menú lateral, opción Ligas

Al entrar en esta pantalla, *Ilustración 22*, se dispone de una lista de ligas en las que participa el usuario. Para acceder a una de ellas, basta con pinchar sobre su nombre. Además de la lista, para cada liga existe un botón para dejar de pertenecer a esta, el botón mostrará una ventana donde confirmar o cancelar la salida como se muestra en la *Ilustración 23.*

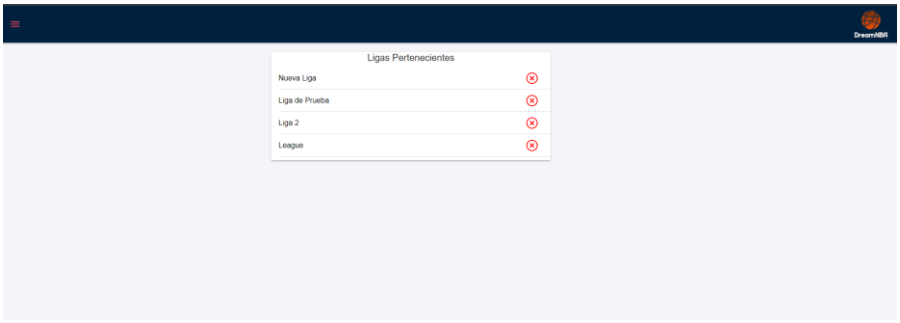


Ilustración 22: Pantalla Ligas

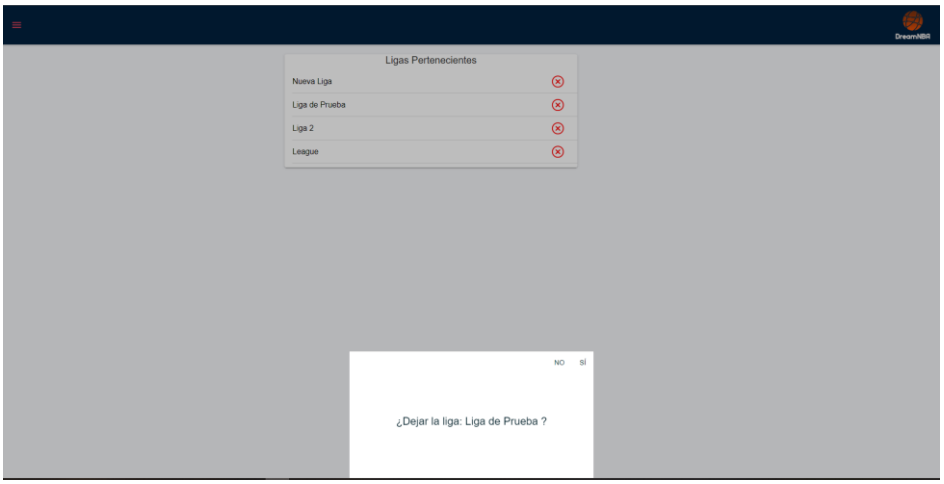


Ilustración 23: Pantalla Ligas, ventana de confirmación/cancelación

En caso de confirmación para dejar la liga se muestra el mensaje que aparece en la *Ilustración 24*.

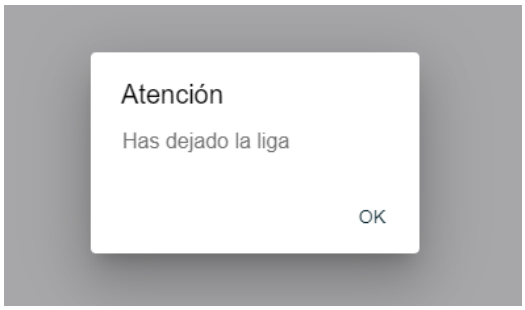


Ilustración 24: Pantalla Ligas, mensaje salida de liga

2.4.5 Logout

Para cerrar la sesión en el sistema el menú lateral contiene una opción, por lo que se puede salir de la sesión desde cualquier punto de la aplicación. La disposición de este botón de muestra en la *Ilustración 25*.

Ilustración 25: Menú lateral, opción Logout

2.5 Pantalla clasificación

La página de clasificación es la primera pantalla que se observa al entrar a una liga, *Ilustración 26,* la cual muestra el ranking general, con el logo, nickname y puntos de cada usuario. También se puede acceder a esta seleccionando la opción clasificación desde cualquier pantalla dentro de la liga como se indica en la *Ilustración 27*.

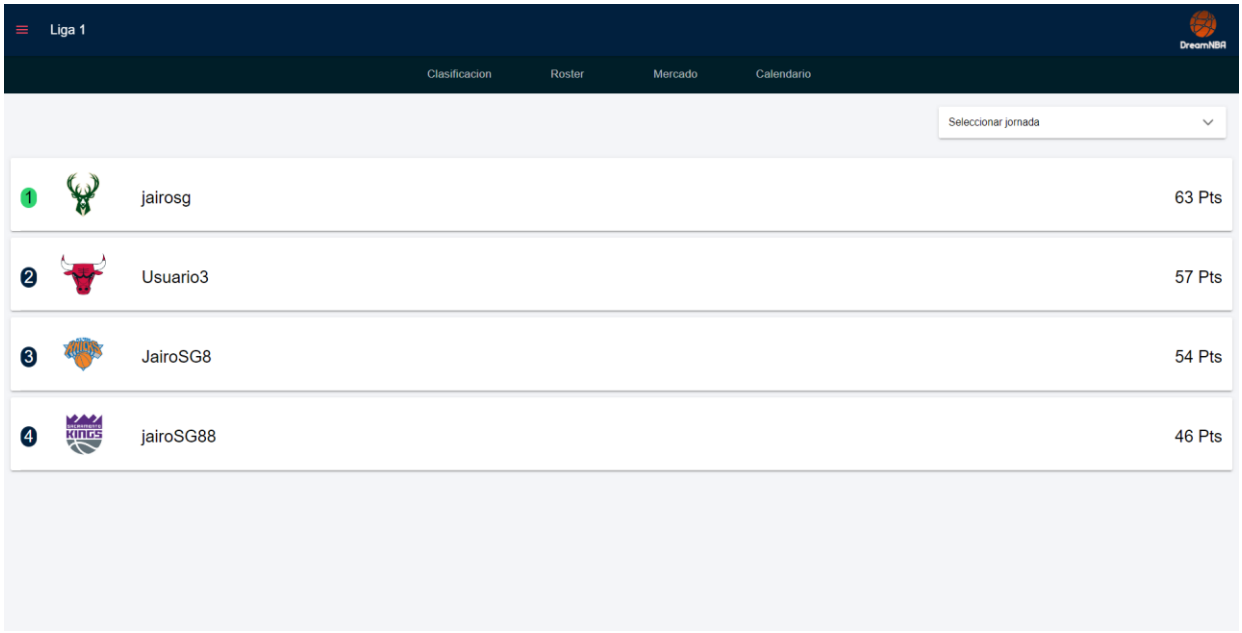


Ilustración 26: Pantalla Clasificación

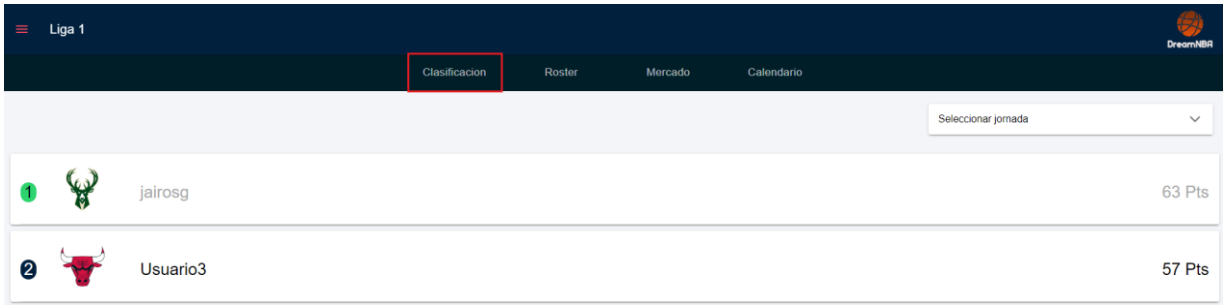


Ilustración 27: Pantalla Clasificación, opción Clasificación

Esta pantalla dispone de un botón, el cual despliega la lista de jornadas disputadas para seleccionar una en concreto y visualizar la clasificación de dicha jornada, como se observa en la *Ilustración 28.*

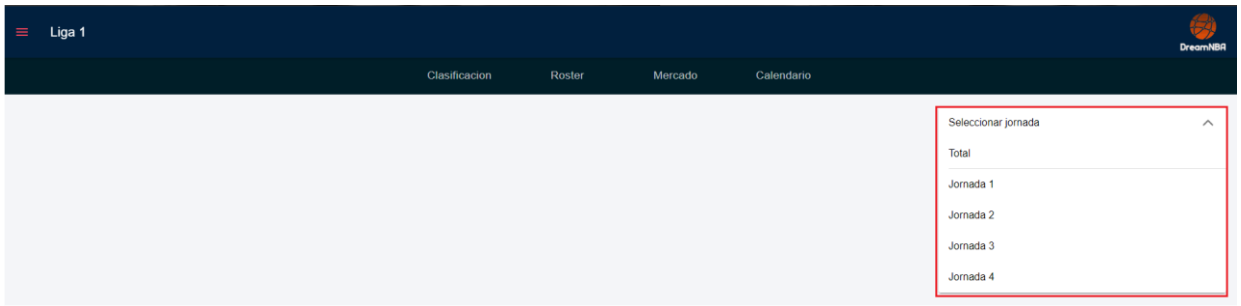


Ilustración 28: Pantalla Clasificación, despliegue de jornadas

2.5.1 Pantalla roster de un usuario/amigo

Se accede tras pulsar sobre la caja de un usuario en la pantalla clasificación, indicado en la *Ilustración 29*.

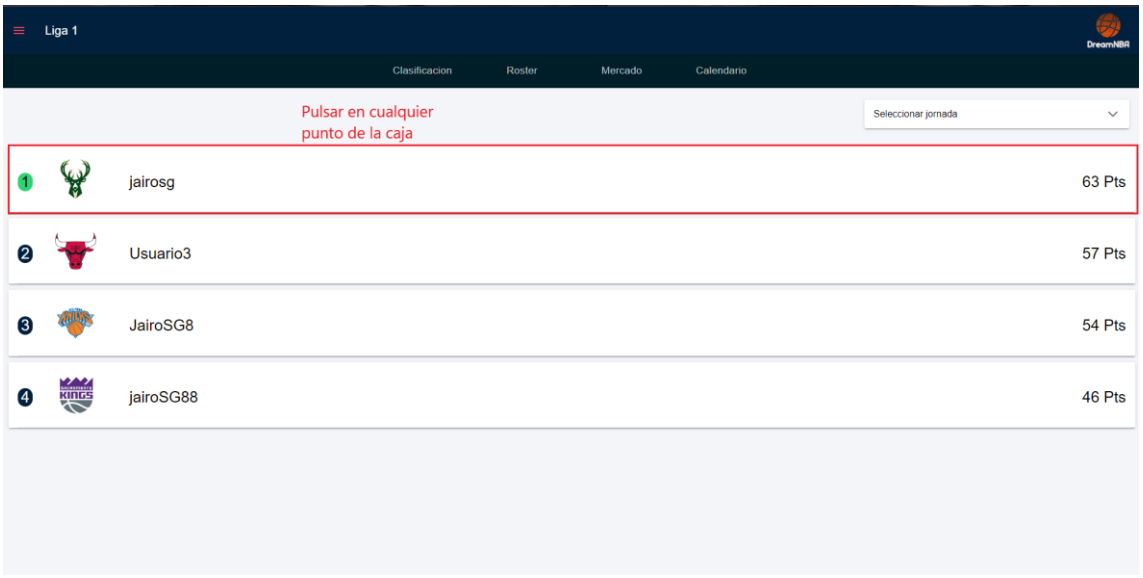


Ilustración 29: Pantalla Clasificación, acceso a roster de usuario/amigo

Se muestran situaciones diferentes en función si se ha seleccionado una jornada, como se explica en el apartado superior, o por el contrario, se está visualizando la clasificación general. En las siguientes imágenes, *Ilustración 30* e *Ilustración 31*, respectivamente, se pueden observar ambas opciones.

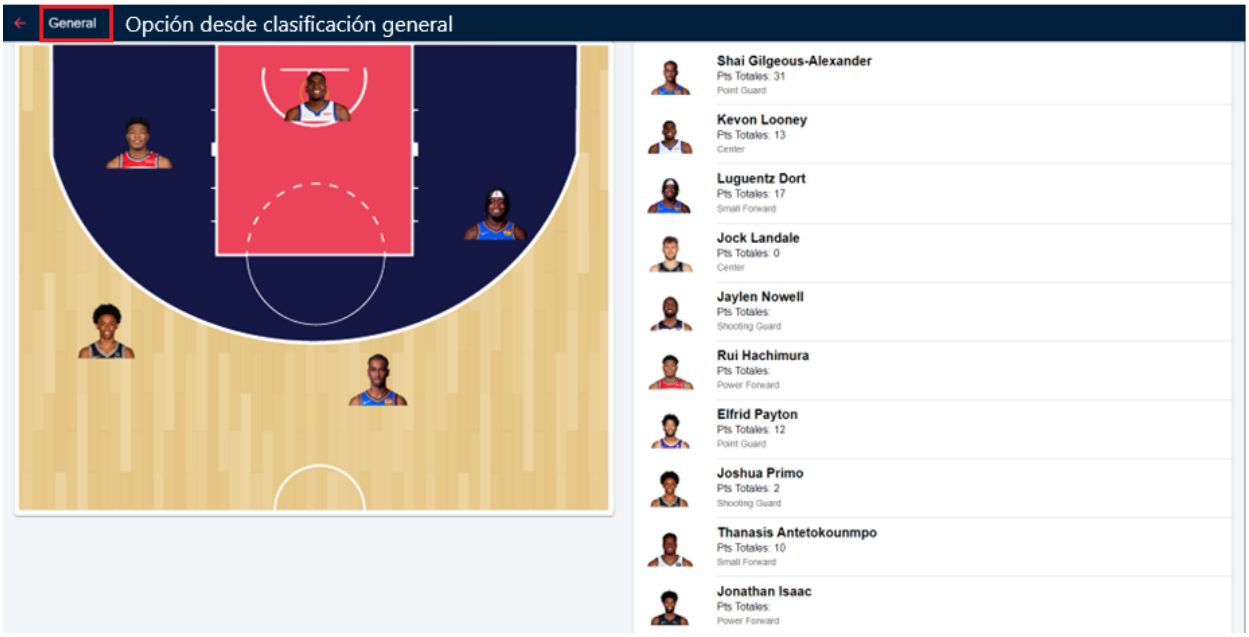


Ilustración 30: Pantalla Usuario/Amigo, opción general

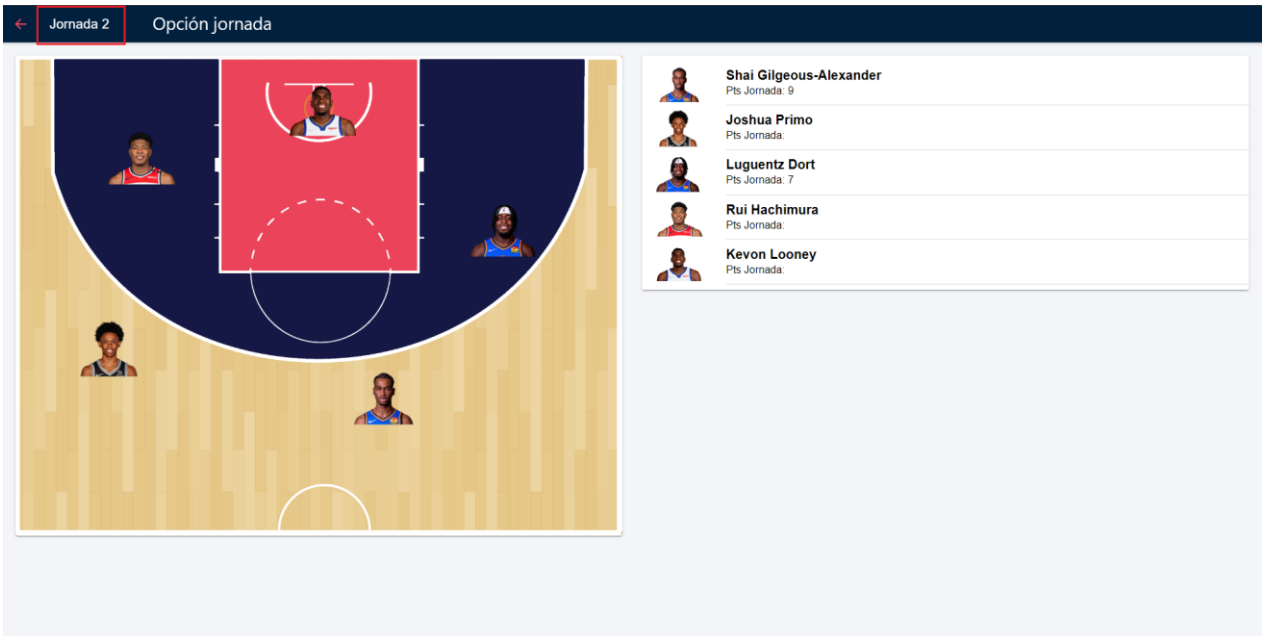


Ilustración 31: Pantalla Usuario/Amigo, opción jornada

Por último, en ambas opciones para retornar hacia la pantalla clasificación se tiene un botón en la esquina izquierda de la cabecera.

2.6 Pantalla roster

Se accede a esta página desde cualquier otra pantalla dentro de una liga como se indica en la *Ilustración 32*.

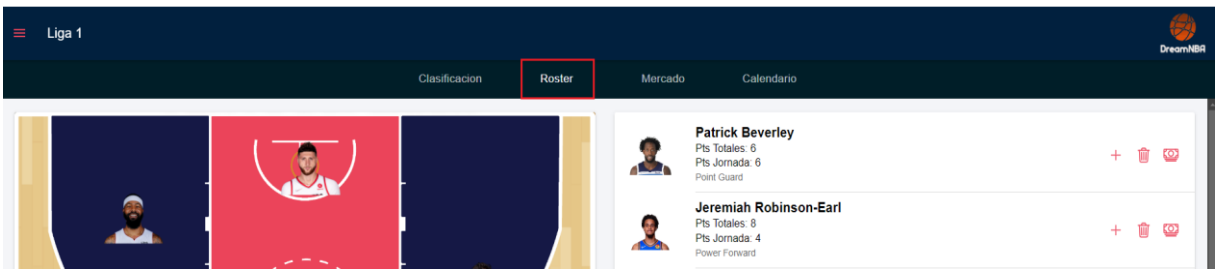


Ilustración 32: Pantalla Roster, opción Roster

Dentro de esta pantalla se dispone de dos partes diferenciadas: el quinteto y los jugadores. Empezando por el quinteto, este se despliega sobre una imagen de una mitad de cancha de baloncesto, en donde los jugadores seleccionados se posicionan sobre ella según la posición que ocupen, como se explica en la *Ilustración 33*. En caso de haber alguna modificación sobre el quinteto aparece un botón para realizar su guardado.

Ilustración 33: Pantalla Roster, quinteto

Siguiendo con la segunda parte de esta pantalla, los jugadores, se dispone de una lista con los jugadores que el usuario tiene, de cada uno se muestra su foto, nombre, posición, puntos totales, puntos obtenidos en la última jornada y posición. A parte de estas características, sobre cada jugador se pueden realizar tres acciones, poner al quinteto, quitar del quinteto y ponerlo a la venta, representadas por tres botones.

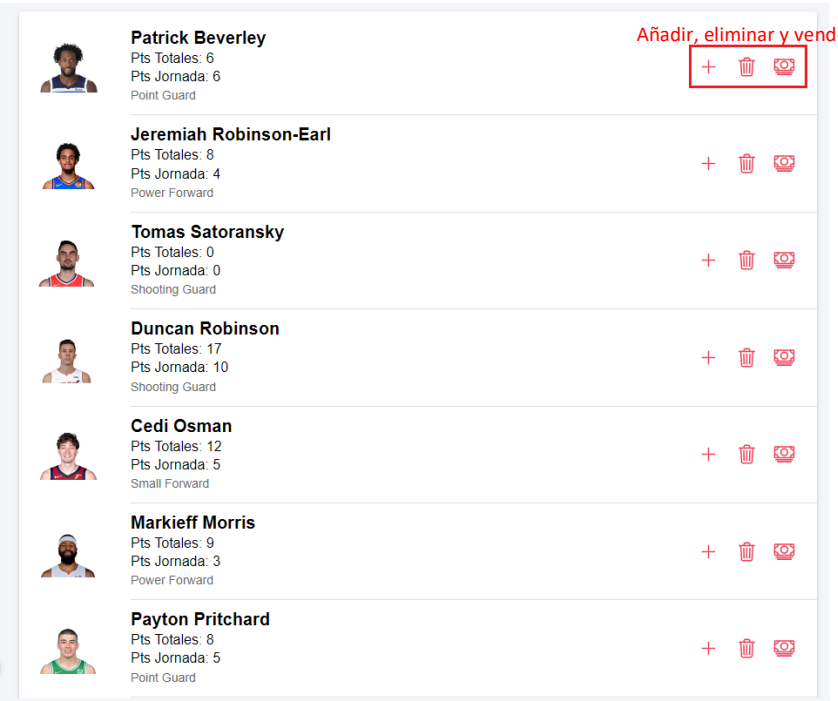


Ilustración 34: Pantalla Roster, jugadores

Por último, la vista general de la pantalla es la que se muestra en la *Ilustración 35.*

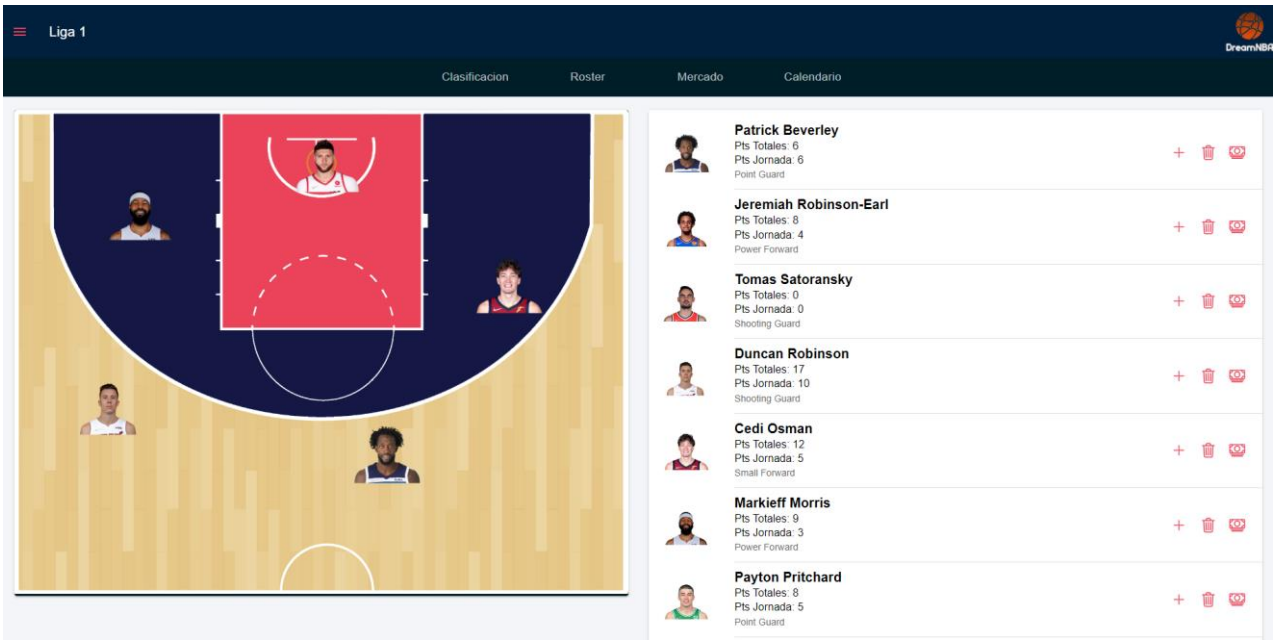


Ilustración 35: Pantalla Roster

2.7 Pantalla mercado

Se accede a esta página desde cualquier otra pantalla dentro de una liga como se indica en la *Ilustración 36*.

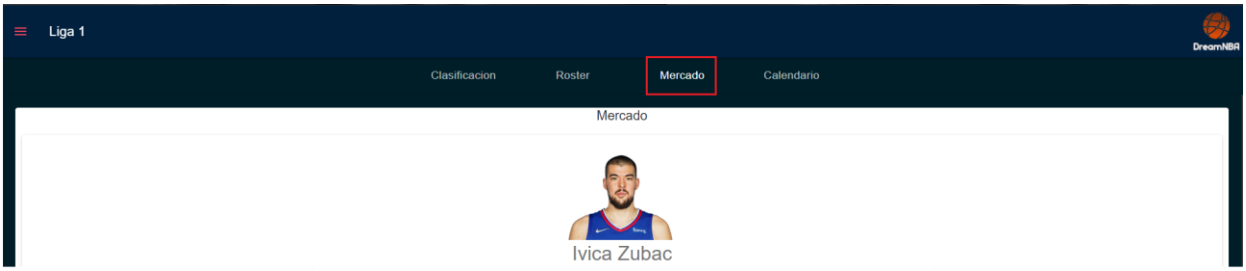


Ilustración 36: Pantalla Mercado, opción Mercado

En esta pantalla se muestra la lista de jugadores que dan forma al mercado*, Ilustración 37*, donde cada uno de ellos tiene su foto, nombre, puntos totales, valor y posición. Además de las características personales del jugador, existen varios botones con diferentes acciones en función de algunas variables como por ejemplo, si sobre el jugador se ha hecho alguna puja o no, o por el contrario, si el jugador pertenece al usuario que ve el mercado. A parte de lo relacionado con los jugadores, se muestra el presupuesto total y el gastado en pujas.

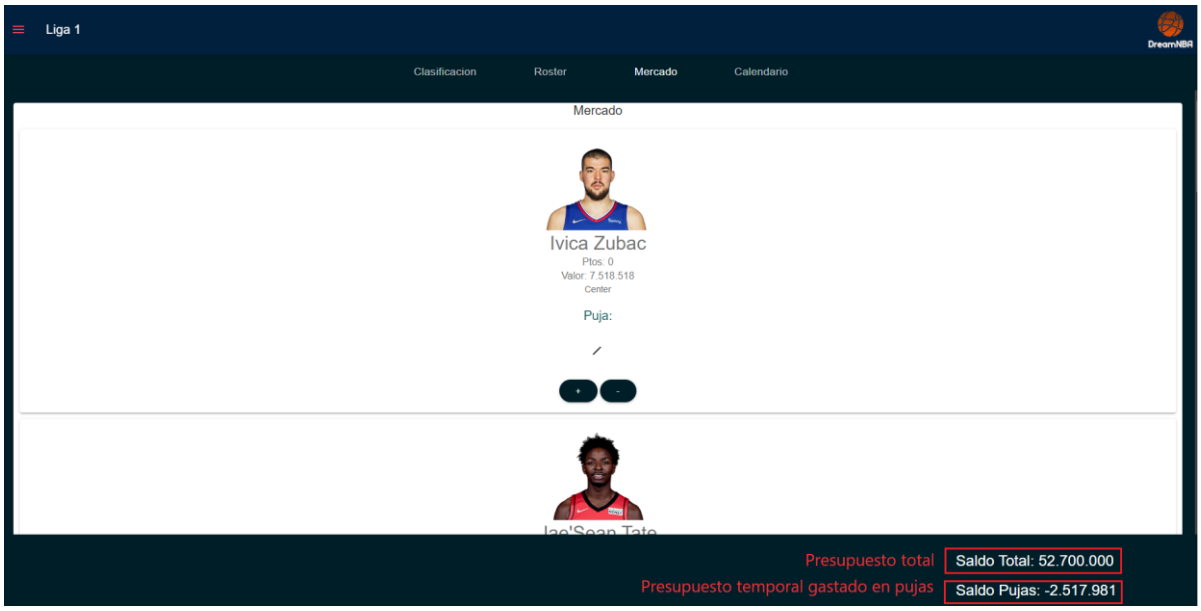


Ilustración 37: Pantalla Mercado

A continuación, en las siguientes ilustraciones se podrá observar cada una de las opciones descritas:

- Opción jugador sin puja: tiene dos botones, aumentar puja y decrementar puja, respectivamente, además de un campo donde introducir manualmente el valor de esta. Existe un tercer botón que se muestra cuando la puja tiene un valor. Se muestra en *Ilustración 38* e *Ilustración 39*.

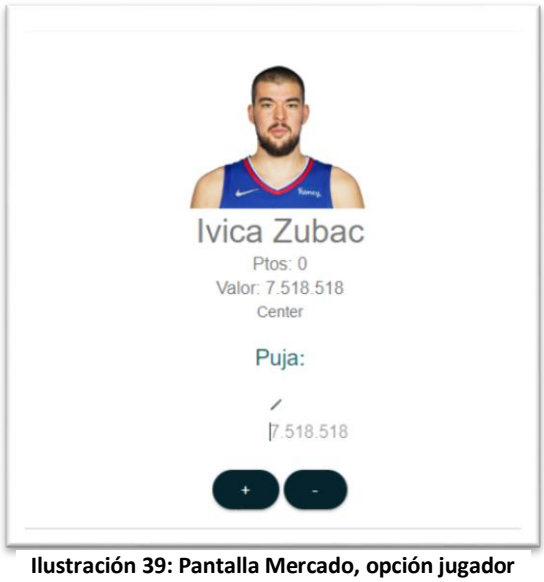


Ilustración 39: Pantalla Mercado, opción jugador sin puja II

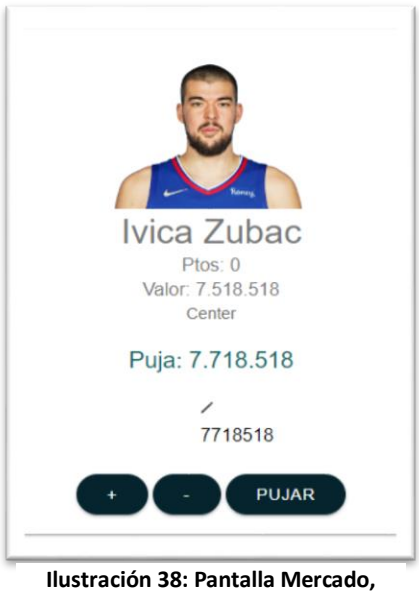


Ilustración 38: Pantalla Mercado, opción jugador sin puja I

- Opción jugador con puja: a diferencia de la opción anterior, a parte de los botones ya mencionados en ella, se añaden dos más, uno para eliminar la puja y otro para editarla. Los pasos para la edición de la puja se describen en la *Ilustración 40*.

Ilustración 40: Pantalla Mercado, opción jugador con puja

- Opción jugador propio: dos botones, uno para eliminar al jugador del mercado y otro para venderlo, este sólo estará disponible cuando otro usuario haya pujado por él.

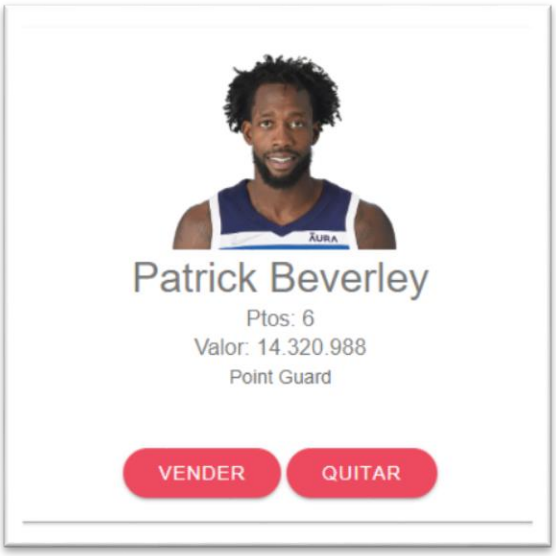


Ilustración 41: Pantalla Mercado, opción jugador propio

Por último, en esta pantalla se presentan ventanas de confirmación/cancelación con cada acción realizada, como las que se muestran en la *Ilustración 42*, *Ilustración 43*, *Ilustración 44* e *Ilustración 45*.

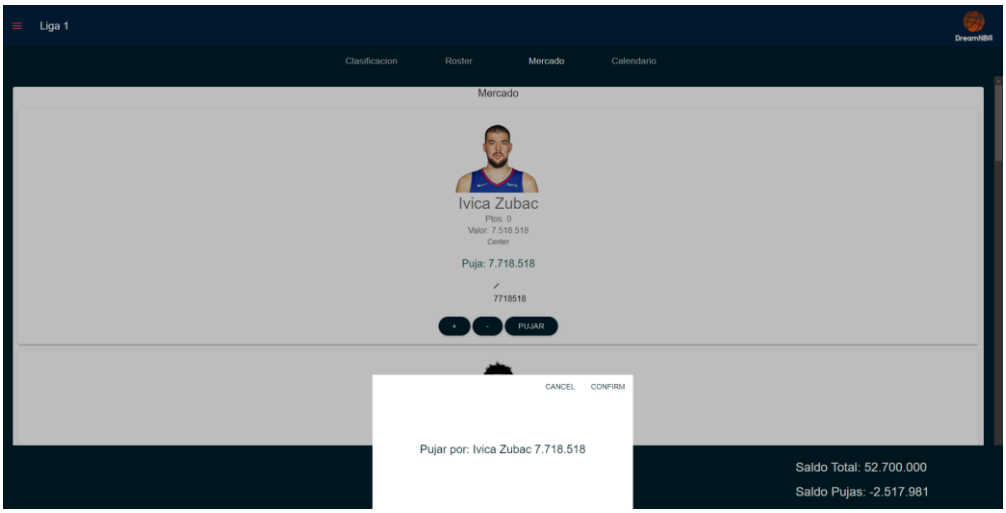


Ilustración 42: Pantalla Mercado, ventana confirmación/cancelación puja

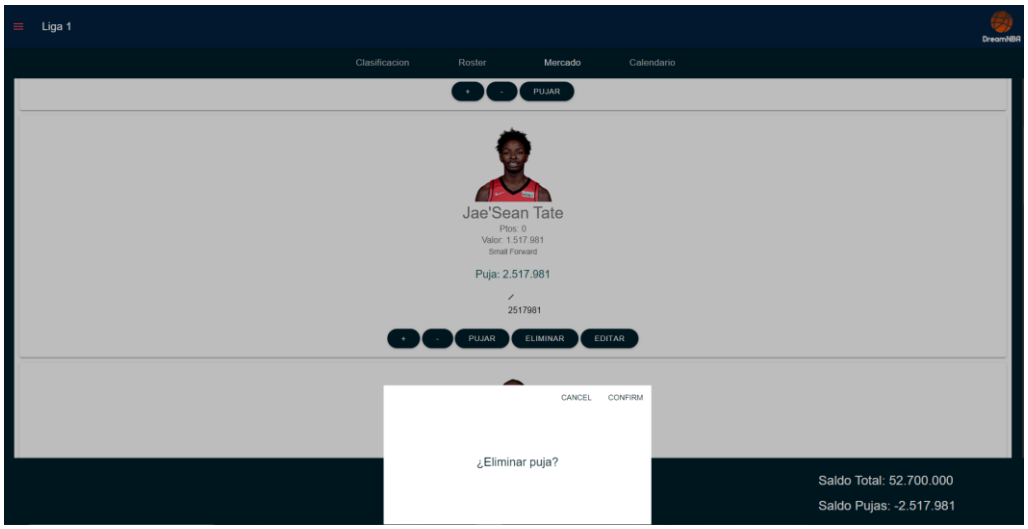


Ilustración 43: Pantalla Mercado, ventana confirmación/cancelación eliminar puja

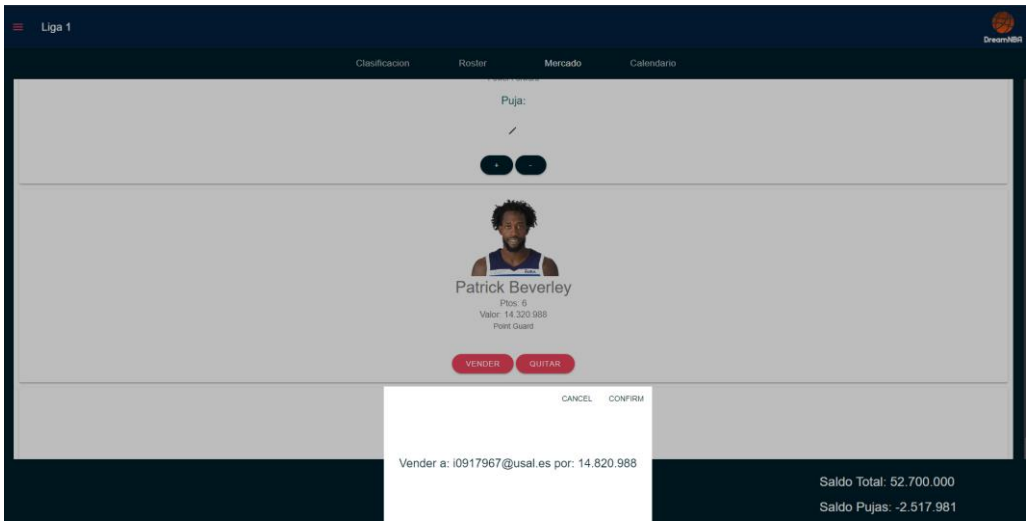


Ilustración 44: Pantalla Mercado, ventana confirmación/cancelación vender jugador

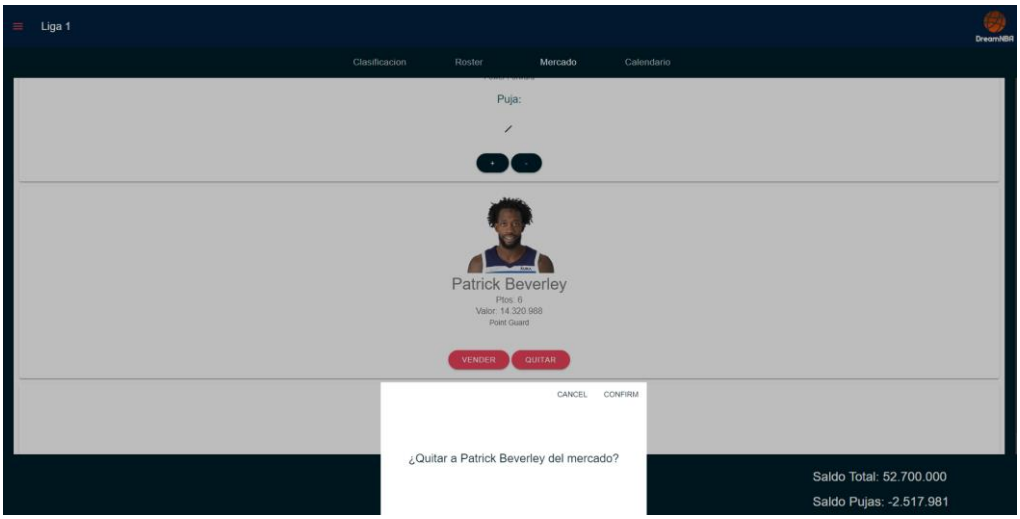


Ilustración 45: Pantalla Mercado, ventana confirmación/cancelación quitar jugador

Además se realiza la comprobación en caso de realizar una puja que supere el presupuesto, por lo que se muestra por pantalla un mensaje indicándolo, *Ilustración 46*. A parte de esta indicación, se ilustra al usuario cuando se ha llevado una de las acciones explicadas anteriormente, como por ejemplo, cuando ha realizado una puja o si se ha eliminado correctamente, a través de mensajes, como modelo se tienen las siguientes ilustraciones, *Ilustración 47* e *Ilustración 48*, haciendo referencia a las acciones comentadas.

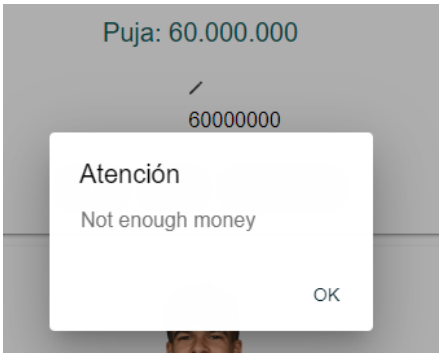


Ilustración 46: Pantalla Mercado, dinero insuficiente

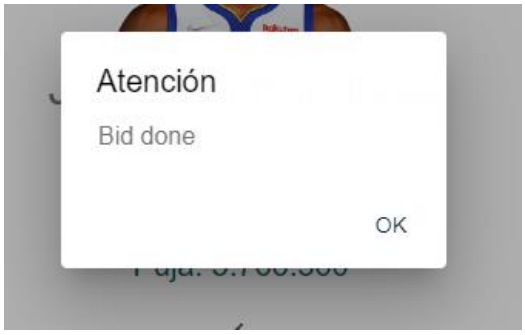


Ilustración 47: Pantalla Mercado, mensaje puja realizada

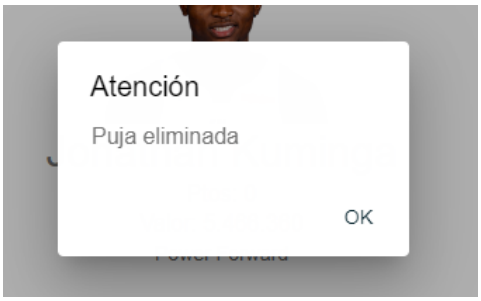


Ilustración 48: Pantalla Mercado, mensaje puja eliminada

2.8 Pantalla calendario

Se accede a esta página desde cualquier otra pantalla dentro de una liga como se indica en la *Ilustración 49*.

Ilustración 49: Pantalla Calendario, opción calendario

Inicialmente, cuando se accede a esta pantalla, aparecen los partidos relativos a la fecha en la que se visita. En adición , a través de un botón, indicada su posición en la *Ilustración 50* se despliega un calendario para elegir la fecha de la que se desea visualizar los partidos, *Ilustración 51.*

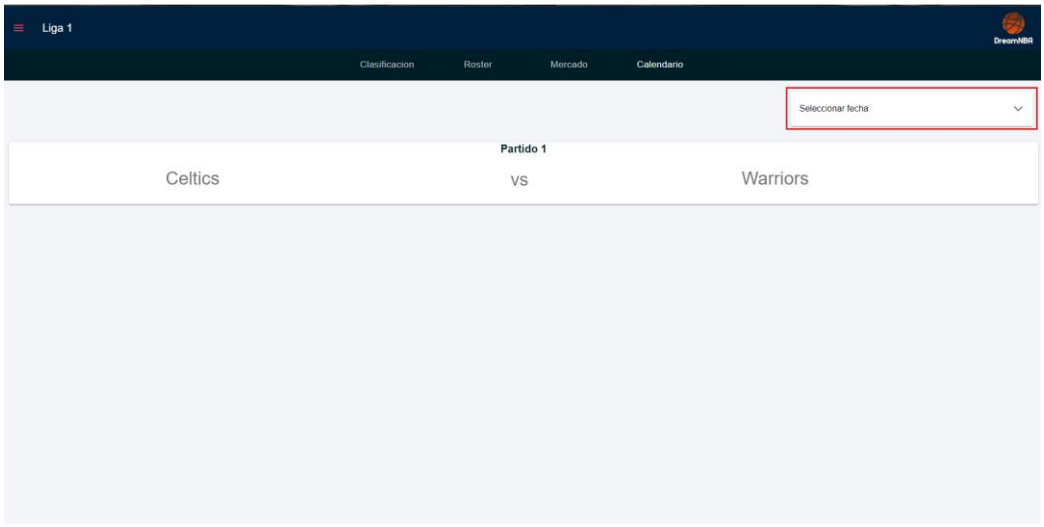


Ilustración 50: Pantalla Calendario

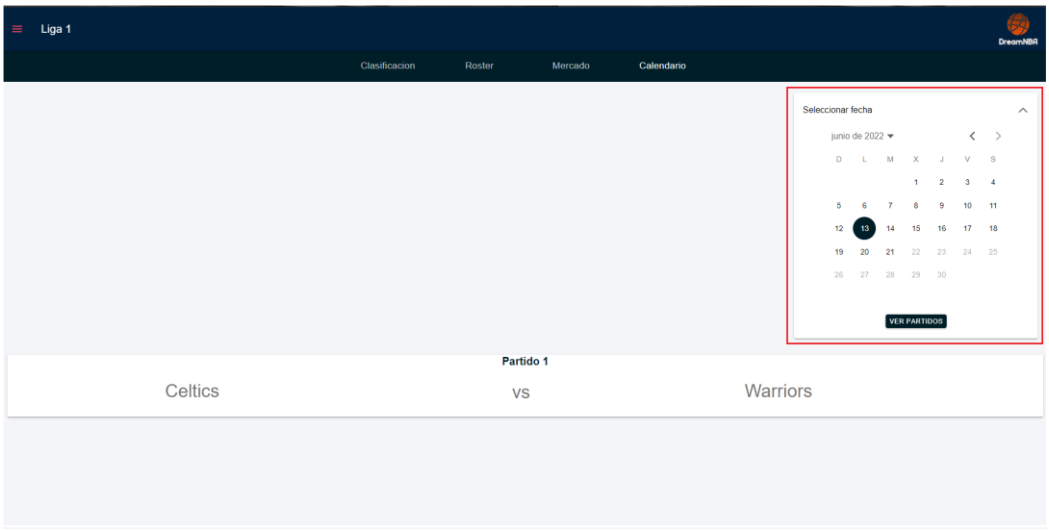


Ilustración 51: Pantalla Calendario, despliegue de calendario

Dependiendo de la fecha, junto con los equipos se muestra el resultado de dicho partido, siendo la fecha posterior a la actual, *Ilustración 52*, por el contrario, simplemente aparece como en la *Ilustración 50*.

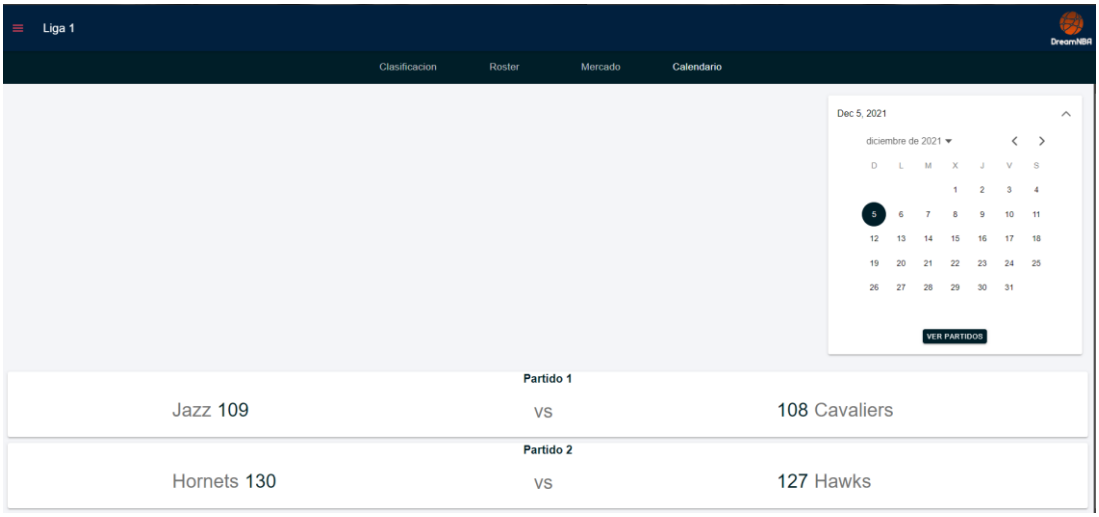


Ilustración 52: Pantalla Calendario, fecha posterior

Desde esta página, pulsando sobre un partido, indicado en la *Ilustración 53*, se accede a la página de estadísticas.

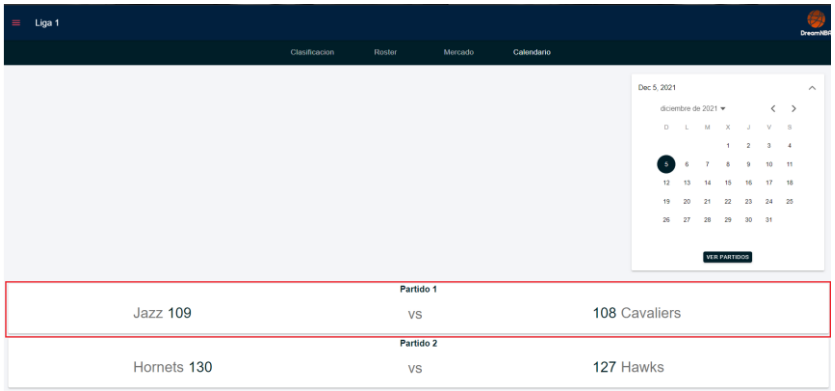


Ilustración 53: Pantalla Calendario, acceso a página estadísticas

2.8.1 Pantalla estadísticas

Esta página contiene las estadísticas individuales de los jugadores que participan en un partido. Se muestra una tabla con la foto, nombre, puntos, asistencias, rebotes, tapones, robos y pérdidas de cada jugador, *Ilustración 54*.

Para volver hacia la pantalla de calendario se dispone de un botón en la cabecera.

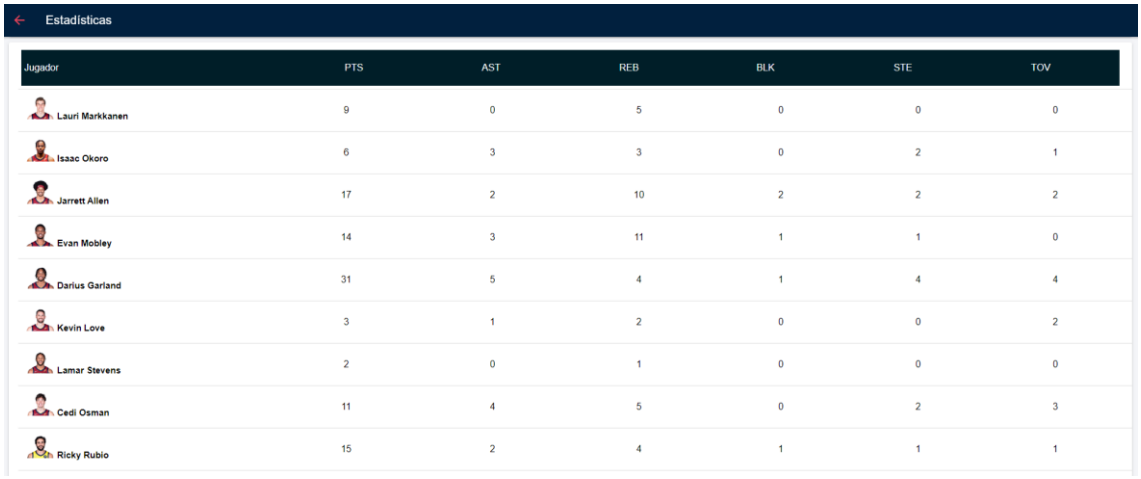


Ilustración 54: Pantalla estadísticas