

Departamento de Ciencias da Computación e Tecnoloxías da Información

Facultade de Informática



UNIVERSIDADE DA CORUÑA

TRABALLO FIN DE GRAO
GRAO EN ENXEÑERÍA INFORMÁTICA
MENCIÓN EN SISTEMAS DE INFORMACIÓN

PLAY2GETHER

**Plataforma colaborativa para la realización
de eventos deportivos informales**

Estudiante: Sandra Dopico Cantarero
Director/a/es/as: Alejandro Cortiñas Álvarez
Miguel Ángel Rodríguez Luaces

A Coruña, 19 de junio de 2019.

A mis padres Arturo y Mar, espero que estéis orgullosos de mi

Agradecimientos

En primer lugar quería agradecer a mis tutores, Alejandro y Miguel, por la paciencia , la ayuda y la atención que me prestaron. Ellos siempre sabían como y cual era la mejor manera de solucionar los problemas.

A mi familia, por intentar ponerse en mi lugar cuando estaba de mal humor por que las cosas no salían como tenían que salir.

A mis compañeros de Universidad, Cristina, Laura, David y José, gracias por hacer que esta etapa de mi vida haya sido mucho más amena, las penas son menos cuando se comparten y a vuestro lado hasta se hacían divertidas.

Y finalmente a Aarón, por recordarme día a día que valgo más de lo que yo pienso.

Resumen

En cualquier comunidad, tanto jóvenes como adultos practican deportes en grupo de manera habitual. Sin embargo, en algunos de ellos se requiere un número importante de personas, por lo que es frecuente tener problemas para encontrar jugadores. Por ello surgió Play2Gether, una aplicación que permite a sus usuarios formar parte de una comunidad virtual centrada en organizar y jugar partidos de fútbol, tenis, pádel, baloncesto...

Para ello se gestionan dos tipos de usuarios: administrador y usuarios normales. Los administradores son los encargados de gestionar los usuarios, deportes y las localizaciones. Por otro lado, los usuarios pueden principalmente planificar eventos deportivos y unirse a partidos organizados.

La aplicación se compone de un servicio REST, un cliente WEB y finalmente un SGBD relacional en cuanto a la parte de almacenamiento de la información. Este proyecto se gestionó siguiendo una metodología iterativa e incremental para el desarrollo de software.

Abstract

In any community, both young people and adults practice group sports on a regular basis. However, some of them require a significant number of people, so it is common to have problems finding players. This is why Play2Gether was born, an application that allows users to be part of a virtual community focused on organizing and playing football, tennis, paddle tennis, basketball...

To do this, check the types of users: administrator and normal users. The administrators are responsible for managing users, sports and locations. On the other hand, users can be, mainly, sporting events and join organized parties.

The application consists of a REST service, a WEB client and, finally, a DBMS. This project was managed following an iterative and incremental methodology for software development.

Palabras clave:

Aplicación Web

Java

Vue.js

Spring Boot

Hibernate

PostgreSQL

Leaflet

Git

Deportes

Partidos

Jugadores

Red Social

Keywords:

Web Application

Java

Vue.js

Spring Boot

Hibernate

PostgreSQL

Leaflet

Git

Sports

Games

Players

Social Network

Índice general

1	Introducción	1
1.1	Motivación	1
1.2	Objetivos	1
2	Fundamentos tecnológicos	3
2.1	Estado del arte	3
2.2	Tecnologías utilizadas	6
3	Metodología y planificación	9
3.1	Metodología de desarrollo	9
3.2	Planificación y seguimiento	11
4	Análisis	15
4.1	Actores	15
4.2	Requisitos	16
4.2.1	Requisitos del Cliente	16
4.2.2	Requisitos del Sistema	18
4.3	Arquitectura del sistema	25
4.4	Interfaz de usuario	26
4.4.1	Componentes y Navegación	26
4.4.2	Mockups Componentes	31
4.5	Modelo conceptual de datos	34
5	Diseño	39
5.1	Arquitectura tecnológica del sistema	39
5.2	Back-end	40
5.2.1	DAOs	41
5.2.2	Servicios	42

5.2.3	Controladores	44
5.3	Front-end	47
5.3.1	Componentes	48
5.3.2	Rutas	49
5.3.3	Comunicación con el servidor	50
6	Implementación y pruebas	55
6.1	Implementación	55
6.1.1	Recomendación de Partidos	55
6.1.2	Motor de Renderizado de pantallas	60
6.2	Pruebas	62
6.2.1	Pruebas Unitarias	62
6.2.2	Pruebas REST	64
6.2.3	Pruebas de integración y aceptación	65
7	Solución desarrollada	67
7.1	Acceso a la Aplicación	67
7.2	Administración	68
7.3	Usuarios Básicos	71
8	Conclusiones y trabajo futuro	77
8.1	Conclusiones	77
8.2	Trabajo Futuro	77
A	Glosario de acrónimos	81
B	Patrones	83
B.1	Patrones	83
B.1.1	Back-End	83
B.1.2	Front-End	86
C	Manual de Instalación	89
D	Contenido del CD	91
E	Mockups	93
	Bibliografía	113

Índice de figuras

2.1	Logo de TimPik	3
2.2	Pantalla de búsqueda de eventos deportivos	4
2.3	Pantalla de detalle de un partido	4
2.4	Logo de Fubles	4
2.5	Pantalla de búsqueda de eventos deportivos	5
2.6	Pantalla de detalle de un partido	5
3.1	Metodología Incremental	9
3.2	Seguimiento	13
3.3	Diagrama de Gantt seguimiento	13
4.1	Relaciones de herencia entre actores	16
4.2	Caso de Uso: Usuario Anónimo	19
4.3	Caso de Uso: Usuario Administrador	20
4.4	Caso de Uso Crear Partido: Usuario Básico	21
4.5	Caso de Uso Gestión Partidos: Usuario Básico/Creador/Jugador	21
4.6	Caso de Uso Gestión Valoraciones: Usuario Jugador	23
4.7	Caso de Uso Gestión Perfiles: Usuario Básico	24
4.8	Caso de Uso Gestión Social: Usuario Básico	24
4.9	Arquitectura del sistema	25
4.10	Componente Game: Mapa	31
4.11	Componente Game: Calendario	32
4.12	Componente GameDetail	32
4.13	Componente GameCreate	33
4.14	Componente GameUser	34
4.15	Diagrama de Clases	37
5.1	Arquitectura tecnológica del sistema	40

5.2	Diagrama PlayerDAO	41
5.3	User Service	42
5.4	Comment, Player, PlayerValoration y Sport Service	43
5.5	Game, Location, SocialRelationShip y Team Service	44
5.6	Estructura Componente vue	48
5.7	Diagrama de componentes del Cliente	49
5.8	Comunicación de los componentes del Usuario del Cliente al Servidor [1] . . .	51
5.9	Comunicación de los componentes del Usuario del Cliente al Servidor [2] . . .	52
5.10	Comunicación de los componentes del Usuario del Cliente al Servidor [3] . . .	52
5.11	Comunicación de los componentes de Administración del Cliente al Servidor .	53
5.12	Comunicación de los componentes comunes del Cliente al Servidor	53
6.1	Petición Cliente: Recomendaciones	55
6.2	Algoritmo de Recomendación: Jugadores	56
6.3	Algoritmo de Recomendación: Deportes	57
6.4	Algoritmo de Recomendación: Localizaciones	58
6.5	RecomendacionDTO	58
6.6	Diagrama de Secuencia del Algoritmo de Recomendación	59
6.7	Componentes de visualización de resultado según deporte	60
6.8	Cambio de plantilla según valor del componente de entrada	60
6.9	Resultado partido de Tenis (JSON)	61
6.10	Resultado partido de Fútbol (JSON)	61
6.11	Funcionamiento Motor Plantillas	61
6.12	Pruebas Realizadas sobre los Servicios	63
6.13	Pruebas Realizadas sobre el Servicio de Usuario	63
6.14	Pruebas de Excepciones	64
6.15	Petición POST de autenticación	64
6.16	Petición GET	65
7.1	Página de Login	67
7.2	Página de Registro	68
7.3	Página para dar de alta un deporte	69
7.4	Página para editar un deporte	69
7.5	Página para visualizar una localización	70
7.6	Página para dar de alta una localización	70
7.7	Página para dar de baja usuarios	71
7.8	Página de búsqueda de partidos: Mapa	72
7.9	Página de búsqueda de partidos: Mapa ubicación seleccionada	72

ÍNDICE DE FIGURAS

7.10 Página de búsqueda de partidos: Calendario	73
7.11 Página de detalle del partido	73
7.12 Página de detalle del partido: Meteorología	74
7.13 Página del Perfil Público	74
7.14 Página del Tablón de Actividades	75
7.15 Página de Mensajes Directos	75
7.16 Página de Datos Personales	76
 B.1 Diagrama del Patrón DAO [1]	83
B.2 Diagrama MVC Spring [2]	85
B.3 Diagrama Fachada [3]	85
B.4 Patrón Model-View-ViewModel [4]	86
B.5 Patrón Callback [5]	87
B.6 Patrón Promise [6]	87
 E.1 Iniciar Sesión.	93
E.2 Registro.	94
E.3 Búsqueda de Partidos: Mapa.	94
E.4 Búsqueda de Partidos: Selección en el Mapa.	95
E.5 Búsqueda de Partidos: Filtros.	95
E.6 Búsqueda de Partidos: Calendario.	96
E.7 Búsqueda de Partidos: Calendario.	96
E.8 Crear Partido: Seleccionar Fecha.	97
E.9 Crear Partido: Seleccionar Hora.	97
E.10 Crear Partido: Consultar Metereología.	98
E.11 Perfil Usuario: Partidos Organizados.	98
E.12 Página Perfil Usuario: Partidos Jugados.	99
E.13 Perfil Usuario: Partidos Próximos	99
E.14 Perfil Usuario: Partidos Recomendados	100
E.15 Perfil Usuario: Comentarios Cocontrincantes	100
E.16 Perfil Usuario: Notificaciones.	101
E.17 Perfil Usuario: Seguidores.	101
E.18 Perfil Usuario: Seguidos.	102
E.19 Perfil Amigo.	102
E.20 Detalles Partido Fútbol.	103
E.21 Detalles Partido Fútbol: Apuntarse.	103
E.22 Detalles Partido Fútbol: Apuntado.	104
E.23 Detalles Partido Fútbol: Desapuntarse.	104

E.24	Detalles Partido Fútbol: Notificación.	105
E.25	Detalles Partido Fútbol: Notificaciones.	105
E.26	Detalles Partido Fútbol: Comentarios.	106
E.27	Detalles Partido Fútbol: Escribir Comentario.	106
E.28	Detalles Partido Fútbol: Organizado.	107
E.29	Detalles Partido Fútbol: Rellenar Resultados.	107
E.30	Detalles Partido Tenis.	108
E.31	Detalles Partido Finalizado Tenis	108
E.32	Detalles Partido Finalizado Tenis: Valorar.	109
E.33	Detalles Partido Finalizado Tenis: Añadir comentario a una valoración de un jugador.	109
E.34	Detalles Partido Finalizado Tenis: Valorado.	110
E.35	Detalles Partido Organizado Baloncesto.	110
E.36	Detalles Partido Organizado Baloncesto: Cancelar.	111
E.37	Detalles Partido Finalizado Baloncesto.	111
E.38	Tablón de Actividades.	112

Índice de cuadros

3.1	Costes de los Recursos Humanos	12
4.1	Componentes de las pantallas del Administrador y Navegación	26
4.2	Componentes de las pantallas del Usuario y navegación [1]	27
4.3	Componentes de las pantallas del Usuario y navegación [2]	28
4.4	Componentes de las pantallas del Usuario y navegación [3]	29
4.5	Componentes comunes a todos los usuarios	30
5.1	Acceso a UserResource	45
5.2	Acceso a SocialResource	45
5.3	Acceso a TeamResource	45
5.4	Acceso a PlayerValorationResource	46
5.5	Acceso a SportResource	46
5.6	Acceso a CommentResource	46
5.7	Acceso a LocationResource	46
5.8	Acceso a AccountResource	47
5.9	Acceso a PlayerResource	47
5.10	Acceso a GameResource	47
5.11	Rutas de los componentes de Administración y Comunes	50
5.12	Rutas de los componentes de los Usuarios Básicos	50

Capítulo 1

Introducción

En este capítulo se hablará de la motivación principal para la realización de este proyecto y los objetivos que ha de cumplir para realizarse con éxito.

1.1 Motivación

En cualquier comunidad se practican deportes en grupo de manera habitual. Sin embargo, en deportes que requieren un número importante de personas, como puede ser el fútbol, es frecuente tener problemas para encontrar jugadores, ya sea por que la persona no dispone de mucha gente conocida con la que ponerse en contacto o simplemente por la disponibilidad de las personas. En este TFG se ha creado una aplicación Web que permitirá a sus usuarios formar parte de una comunidad virtual centrada en organizar y jugar partidos de fútbol, tenis, pádel, baloncesto, etc. La aplicación se llama Play2Gether y permitirá poner en contacto a usuarios con las mismas necesidades deportivas evitando así esas búsquedas exhaustivas de participantes a través de las redes sociales, que en su mayoría no obtenían resultados satisfactorios.

El objetivo principal es facilitar la planificación de estas actividades deportivas proporcionando una comunidad virtual que facilite la búsqueda de jugadores y proporcione herramientas al usuario que le permitan buscar lo que él necesita.

1.2 Objetivos

A partir del objetivo principal que se ha descrito en la sección anterior, este trabajo de fin de grado debe alcanzar los siguientes objetivos específicos:

- **Facilitar la planificación de los partidos.** En el momento de creación de un partido, se tendrá acceso al pronóstico meteorológico por si se ha escogido una localización al aire libre. Una vez creado, otros usuarios de la aplicación podrán apuntarse hasta

alcanzar un número máximo de participantes. Del mismo modo podrá cancelarse si no se llega al mínimo de jugadores y, como consecuencia, se enviará un correo electrónico a todas las personas apuntadas y a las que tenían el partido marcado para que se les notificara de cualquier cambio.

- **Conseguir participantes de una manera más sencilla.** Uno de los problemas de los deportes en grupo es la dificultad de encontrar jugadores que tengan disponibilidad. Al publicar el partido en nuestra plataforma todas aquellas personas interesadas en él se podrán apuntar sin necesidad de ponernos en contacto con ellos, por lo que el esfuerzo de búsqueda de participantes acaba desapareciendo.
- **Conseguir que los usuarios encuentren lo que buscan.** En primer lugar la plataforma ofrecerá dos formas diferentes de visualizar los partidos vigentes: en mapa o en calendario. En mapa, el usuario podrá visualizar los partidos por su localización, a parte puede filtrar por creador del partido, tipo de deporte, por la media de edad de los jugadores e incluso por dificultad o la experiencia que poseen los jugadores apuntados. En calendario, el usuario podrá ver los partidos en función de su día y hora de realización. También nos recomendará partidos a los usuarios en función de su histórico, ya sea por deporte, localización o jugadores.
- **La flexibilidad de la aplicación.** La aplicación ofrecerá la posibilidad de crear partidos de fútbol, baloncesto, tenis y pádel, pero en un futuro se podrán añadir nuevos deportes por lo que esta transición ha de ser lo más sencilla posible. Para ello esta aplicación será lo más flexible posible para que la incorporación de nuevos deportes no ocasione un gran impacto.
- **Servir de canal de comunicación entre jugadores y creadores.** Cada partido creado en la aplicación tendrá un foro de mensajes que podrán utilizar los usuarios de la aplicación para comunicarse entre ellos, de forma que cualquier duda del partido pueda ser respondida por los participantes en el mismo. En caso de cancelación se avisará por correo a todos los jugadores, evitando así la necesidad de tener que comunicarnos por aplicaciones externas.

Capítulo 2

Fundamentos tecnológicos

En este capítulo se comentarán aplicaciones web similares en las que la autora se ha basado y las tecnologías utilizadas en este proyecto.

2.1 Estado del arte

Partiendo de los objetivos descritos en la Sección 1.2, se ha realizado una búsqueda de herramientas y aplicaciones que puedan dar soporte a nuestras necesidades. Entre las aplicaciones encontradas destacan dos alternativas populares: Timpik y Fubles.

Timpik [7] es una herramienta para poder realizar eventos deportivos. Su mayor ventaja es la gran variedad de deportes que ofrece. Además, también gestiona pagos dentro de la aplicación a través de un monedero virtual. Así, cuando un organizador crea un evento puede detallar sanciones por no acudir finalmente a un partido o el coste por jugador para unirse al evento.



Figura 2.1: Logo de TimPik

La aplicación permite organizar un partido, buscar eventos, buscar jugadores a través de un correo electrónico o Facebook y como funcionalidad destacable, crear un club para organizar partidos con tu grupo de amigos de una manera más rápida y directa. En cuanto a la búsqueda de eventos, se realiza a través de un calendario donde nos permite filtrar por provincia o deportes favoritos. En la pantalla del evento aparece información básica del partido junto con los jugadores y un apartado de comentarios. Finalmente, en el muro público de un

2.1. Estado del arte

jugador se observar el número de partidos jugados, organizados, deportes que practica, clubs a los que pertenece, medallas conseguidas...

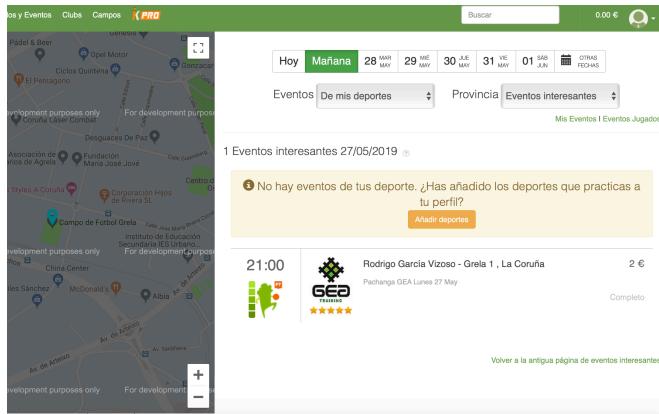


Figura 2.2: Pantalla de búsqueda de eventos deportivos

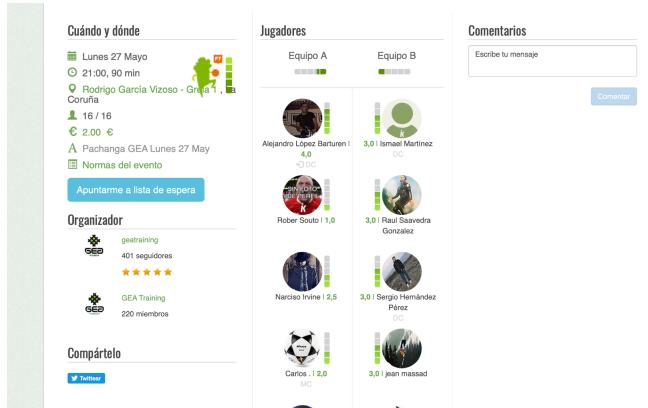


Figura 2.3: Pantalla de detalle de un partido

Fubles [8] es una aplicación similar a Timpik pero que, a diferencia de ella, se centra principalmente en el fútbol. En esta aplicación los pagos suelen abonarse en efectivo en vez de gestionar un monedero virtual.



Figura 2.4: Logo de Fubles

CAPÍTULO 2. FUNDAMENTOS TECNOLÓGICOS

Hay que destacar una funcionalidad bastante interesante, y es que permite a los usuarios nombrar “procurador” a alguno de sus contactos, dándole permisos para gestionar su inscripción o cancelación en las quedadas. La búsqueda de eventos se realiza a través de un mapa. El filtrado de los mismos es más avanzado que en el resto de aplicaciones, pues nos permite filtrar por pabellones, espacios con trgamonedas o eventos deportivos que tengan lugar en ubicaciones de interior.

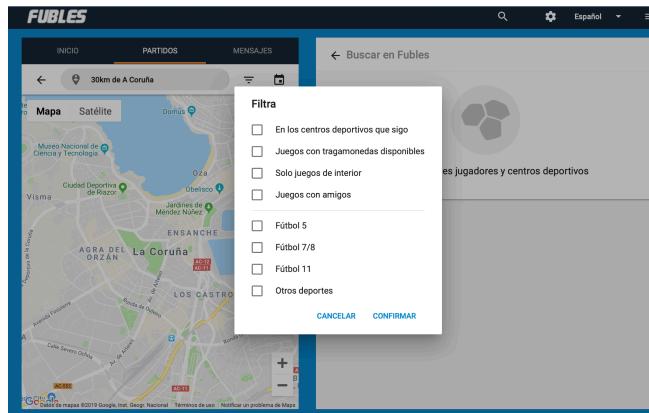


Figura 2.5: Pantalla de búsqueda de eventos deportivos

En el perfil del usuario se pueden ver los datos básicos de cada jugador más los partidos jugados, creados, las veces que ha sido votado mejor jugador en el partido, los goles, las veces que no ha aparecido a un partido finalmente a pesar de estar apuntado... Por último, en la pantalla del evento concreto aparece la información básica del partido junto con los jugadores apuntados, las valoraciones, los resultados finales y un apartado donde la gente puede dejar comentarios acerca del partido.

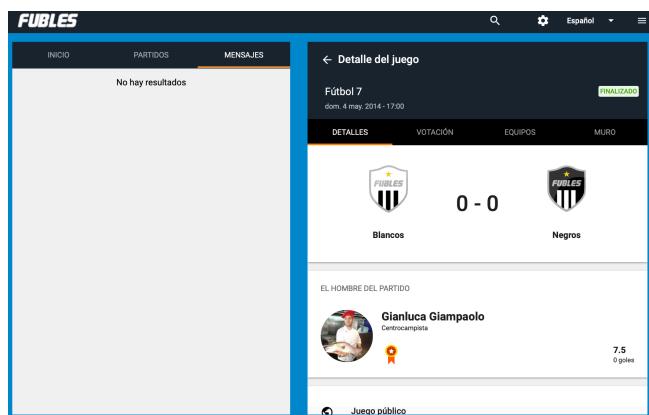


Figura 2.6: Pantalla de detalle de un partido

Una vez analizadas estas dos aplicaciones, se ha comprobado que existen características y funcionalidades no cubiertas que Play2gether debe implementar. Para lograr ser **un canal de comunicación entre usuarios** se tomará como referencia las dos aplicaciones en cuanto a mensajes directos, peticiones de amistad... y se añadirán funcionalidades nuevas como la notificación de usuario, y el acceso a un tablón de actividades. Para **conseguir que los usuarios encuentren lo que buscan**, la mejor de ambas aplicaciones es Fubles, ya que posee filtros más avanzados. A su conjunto de filtros añadiremos otros más específicos como la búsqueda por media de edad y la recomendación de partidos basada en el histórico.

2.2 Tecnologías utilizadas

- **Vue.js** [9, 10, 11, 12, 13], es un framework progresivo para construir interfaces de usuario. Su núcleo es bastante pequeño y se escala a través de plugins. Engloba en un mismo archivo HTML, CSS y JavaScript.
 - **Node.js** [14] es una plataforma de desarrollo para la creación de aplicaciones destinadas a la Web, orientada a redes y centrada en la velocidad y la escalabilidad.
 - **CSS (Cascading StyleSheets)** es un lenguaje de estilo para describir el diseño de una página Web (colores, disposición de elementos y fuentes). Aunque es independiente de HTML (HyperText Markup Language) suele emplearse conjuntamente.
 - **HTML (HyperText Markup Language)** es el lenguaje de marcado o lenguaje de marcas para describir la estructura de páginas web.
 - **JavaScript** es el lenguaje utilizado en páginas Web para incrementar la funcionalidad de las mismas y la interacción con el usuario.
- Leaflet.js** [15] es una librería JavaScript Open Source para creación de mapas y visualización de datos geográficos.
- **Bootstrap** [16] es un conjunto de herramientas para desarrollo con HTML, CSS y JavaScript que ofrece variables Sass con las que personalizar la interfaz.
 - **Java** se trata de un lenguaje de programación interpretado. Posee varias cualidades: orientado a objetos, multiplataforma, multihilo, orientación a la red...
 - **Hibernate** [17, 18, 19], es un framework ORM (Object-Relational Mapping), es decir, una herramienta para realizar una correspondencia entre objetos y las tablas de una base de datos relacional.

- **Spring Boot** [20, 21], es un framework que facilita la configuración de un proyecto Spring, convirtiéndola en automática.
- **PostgreSQL** [22] es un sistema de gestión de base de datos relacional orientado a objetos. Destacar el control de concurrencias multiversión (MVCC) y el uso de bloqueos de lectura.

2.2. Tecnologías utilizadas

Capítulo 3

Metodología y planificación

En este capítulo se va a comentar la metodología de desarrollo seguida en este proyecto y la planificación realizada junto con sus desviaciones finales y costes.

3.1 Metodología de desarrollo

En este proyecto se ha optado por utilizar una metodología iterativa e incremental, además de estar dirigida por las funcionalidades del sistema. Cada iteración ha supuesto un incremento en la funcionalidad hasta alcanzar un producto entregable.

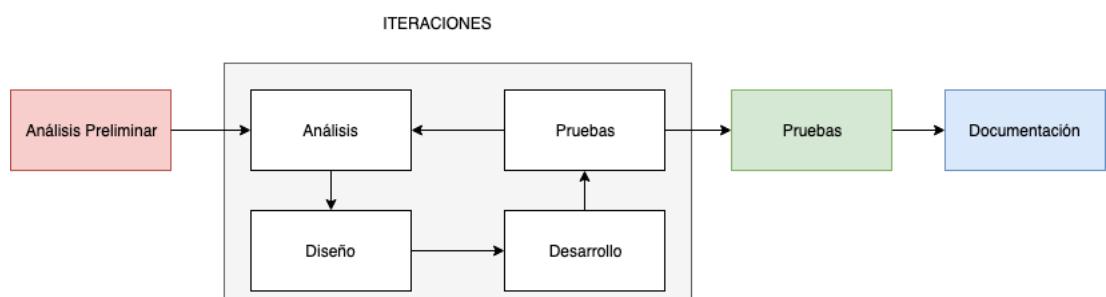


Figura 3.1: Metodología Incremental

Lo primero que se consideró realizar fue el **análisis preliminar**, principalmente el alcance del proyecto. Después comenzó el estudio de las tecnologías necesarias. Así, se realizó una búsqueda y análisis de diferentes herramientas, de las cuales se escogieron las citadas en la Sección 2.2. Finalmente después de haber estudiado el objetivo global del proyecto era necesario empezar a pensar en los objetivos individuales que debería alcanzar cada iteración. Para ello previamente se construyó un punto de partida que consistía en:

- **Modelo de casos de uso:** Sus funcionalidades eran implementadas en sus respectivas iteraciones. Gracias a él definimos el comportamiento del sistema

- **Mockups:** Se crearon maquetas de como serían las interfaces de la aplicación. Junto con el modelo de casos de uso, se utilizaron para planificar las iteraciones.
- **Modelo de datos:** Con este modelo definimos los diferentes datos que debe manejar el sistema y las distintas relaciones que existían entre ellos.

Cada una de las **iteraciones** incluye las siguientes fases:

- **Análisis:** Se analizará la iteración teniendo claro los nuevos requisitos, y analizando las diferentes formas de alcanzar los objetivos.
- **Diseño:** Se definirán los distintos módulos de manera cuidadosa ya que estos podrán ser reutilizados en iteraciones posteriores.
- **Desarrollo:** Se implementará lo definido en la fase anterior, definiéndose las entidades y métodos para poder llevar a cabo los casos de uso definidos.
- **Pruebas:** Se realizarán primero pruebas individuales y luego globales de las funcionalidades creadas, las cuales comprobarán que lo realizado cumple con las expectativas marcadas y posee un comportamiento acertado.

Una vez finalizadas las iteraciones, se realizarán **pruebas y se documentará** lo realizado. Pruebas de uso de la aplicación para verificar que todas las relaciones entre módulos funcionan correctamente, es decir, que estos módulos trabajan en consonancia para ofrecer la funcionalidad deseada como se había planteado desde un principio. En cuanto a la documentación, consiste en la redacción de esta memoria.

Las **herramientas utilizadas** han sido las siguientes:

- **Latex [23]:** Es un sistema de composición de textos. Se ha utilizado para la realización de esta memoria.
- **Postman [24]:** Herramienta que permite realizar peticiones HTTP a cualquier API para comprobar que todas nuestras conexiones REST funcionaban correctamente.
- **Eclipse [25]:** No es más que un entorno de desarrollo integrado (IDE). Se utilizó para la codificación de la capa de servicio.
- **Sublime [26]:** Es un editor de código multiplataforma y ligero. Fue utilizado para la codificación de la capa cliente.
- **MagicDraw [27]:** Herramienta CASE compatible con el estándar UML. Se ha utilizado en este proyecto para la elaboración de los diferentes diagramas de clases y casos de uso.

- **Balsamiq Mockups [28]:** Es una herramienta que permite diseñar maquetas de interfaz. Se ha utilizado para desarrollar una primera idea de como serían las pantallas de la aplicación.
- **Draw.io [29]:** Es una aplicación Web que nos permite crear diagramas de todo tipo desde nuestro navegador.

3.2 Planificación y seguimiento

En esta sección se mostrará una planificación inicial del proyecto y se hará una estimación de este. Por último se tendrán en cuenta las desviaciones producidas y el coste.

El trabajo se divide en cuatro iteraciones de desarrollo con una duración de entre tres y cuatro semanas, dependiendo de la carga de trabajo y de la disponibilidad de la autora. Las **tareas** a realizar en cada iteración son las siguientes:

- **Primera Iteración:** En la primera iteración se realizará todo lo referente a la autenticación de los usuarios (Inicio de Sesión, Registro...) y lo que sería el Módulo de Administración (Gestión de Deportes, Localizaciones y Usuarios).
- **Segunda Iteración:** Se centraría en los partidos, es decir, es su creación, las diferentes formas de mostrarle a los jugadores los partidos disponibles, la gestión de jugadores...
- **Tercera Iteración:** Aquí se trabajarían las funcionalidades relacionadas con las valoraciones, comentarios, relleno de resultados, renderizado de pantallas según el deporte, la recomendación de partidos según su histórico...
- **Cuarta Iteración:** Aquí se realizará la parte social del proyecto, es decir, mensajes directos, foros, seguidores, notificaciones, tablón de actividades...

En cuanto a los **recursos** para este proyecto podemos diferenciar dos roles diferentes: Directores y Analista/Programador.

- **Directores:** En este caso se encargaban de la planificación y supervisión tanto del proyecto global, como de las iteraciones. En cuanto a la comunicación entre los directores y la autora se han realizado reuniones de seguimiento cada 4 semanas, dependiendo de la duración de la iteración y de la disponibilidad de los asistentes, en las que se planifica qué realizar en la siguiente iteración y se revisa lo anterior.

Hora de trabajo del Director: 26,69 €/h.

- **Analista/Programador:** Este sería el rol de la autora, se encargaría del análisis, diseño, desarrollo y pruebas de cada una de las iteraciones a realizar.

Hora de trabajo del Analista/Programador: 19,06 €/h.

Para la **planificación** se ha realizado un Diagrama de Gantt exponiendo como tareas las iteraciones mencionadas en la Sección 3.2. Las fechas de fin e inicio del proyecto correspondearía con el 28 de Noviembre de 2018 y el 21 de junio de 2019. Se debe mencionar que durante el análisis preliminar se interrumpió el proyecto en dos ocasiones, aunque estas paradas estaban previstas desde un principio, por lo que no afectó a la planificación del proyecto:

- **Del 8 al 21 de Enero:** La causa fue la preparación de los exámenes del primer cuatrimestre.
- **Del 1 al 3 de Febrero:** Por motivos personales de la autora.

En cuanto a los **costes**, la autora trabajará **5h al día** todos los días de la semana, lo que suponen **35 horas semanales aproximadamente**. Teniendo en cuenta que la duración del proyecto ascenderá aproximadamente a **205 días totales**, el total de horas ascenderían a **1025 horas**. Por otra parte, los directores del proyecto dedicarán aproximadamente **40 horas** entre reuniones y la revisión de la memoria. Los costes de los recursos se han obtenido de la siguiente manera:

Actor	Horas/persona
Sandra Dopico Cantarero	1025
Alejandro Cortiñas Álvarez	40
Miguel Ángel Rodríguez Luaces	40

Cuadro 3.1: Costes de los Recursos Humanos

- **Director del Proyecto:** Suponemos que el salario bruto sería de unos 35.000 €/año, pero si tenemos en cuenta lo que la empresa paga a la seguridad social por ese trabajador (multiplicamos por 1.35) quedaría en unos 47.250 €/año. Contando con que un año laboral consta de 1770 horas, haciendo el cálculo obtendríamos esos 26,69 €/h.
- **Analista/Programador:** Se parte de un salario bruto de 25.000 € y se calcularía de la misma forma, quedaría en uno 33.750 €, y haciendo el calculo obtendríamos esos 19,03 €/h.

En cuanto al **seguimiento**, durante la realización del proyecto se produjeron una serie de desviaciones en la duración de algunas tareas. Pero estas desviaciones en las duraciones de las tareas, al ser de uno o dos días únicamente, acabaron siendo compensados en iteraciones posteriores, por lo que el esfuerzo final para la realización del proyecto no ha variado. Únicamente se incluyó el Diagrama de Gantt del Seguimiento que se puede observar en la Figura 3.2 debido a la escasa diferencia respecto a la Planificación Inicial.

CAPÍTULO 3. METODOLOGÍA Y PLANIFICACIÓN

Etapas proyecto	Fecha Inicio	Fecha Fin
Estudio de la Tecnología	28 Noviembre	8 Febrero
Análisis Preliminar	28 Noviembre	8 Febrero
Funcionalidad		
Prototipos de Pantalla		
Diagrama conceptual BD		
Servicio REST		
Planificación	7 Febrero	8 Febrero
Dividir en Iteraciones		
Desarrollo 1 Iteración	8 Febrero	10 Marzo
Registro de Usuarios		
Inicio Sesión		
Perfiles de Usuario		
Gestión de Deportes		
Gestión de Localizaciones		
Gestion de Usuarios		
Desarrollo 2 Iteración	10 Marzo	3 Abril
Gestión de Partidos		
Gestión de Jugadores		
Metereología		
Mapas y Calendarios		
Filtrado de Partidos		
Desarrollo 3 Iteración	5 Abril	4 Mayo
Notificaciones de partidos		
Relleno de Resultados		
Visualización de Resultados		
Valoraciones jugadores		
Valoraciones partidos		
Partidos Recomendados		
Desarrollo 4 Iteración	6 Mayo	25 Mayo
Foro del partido		
Mensajería		
Solicitudes de amistad		
Notificación de usuarios		
Tablón de Actividades		
Memoria	26 Mayo	21 Junio

Figura 3.2: Seguimiento



Figura 3.3: Diagrama de Gantt seguimiento

3.2. Planificación y seguimiento

Capítulo 4

Análisis

En este capítulo de la memoria nos centraremos en el análisis de la aplicación, describiendo los actores y requisitos, la arquitectura del sistema y los datos mediante un Diagrama de Clases.

4.1 Actores

Un actor es una entidad que participa en un caso de uso, y que puede tener relaciones de herencia con otros actores. En esta sección se mostrarán los cinco actores presentes en la aplicación.

- **Usuario Anónimo:** Sólo podrá iniciar sesión o registrarse en la aplicación.
- **Administrador:** Se encarga de la gestión de Deportes, Localizaciones y Usuarios.
- **Usuario Básico:** Tiene acceso a prácticamente todas las funcionalidades de la aplicación, excepto a las de gestión del Administrador y las funcionalidades añadidas que tienen los dos siguientes actores.
- **Usuario Creador:** Es la persona responsable de la creación de un partido. Engloba todas las funcionalidades del Usuario Básico solo que tendrá funcionalidades añadidas sobre el partido que ha creado.
- **Usuario Jugador:** Es la persona que ha jugado un determinado partido. Engloba todas las funcionalidades del Usuario Básico y además este tendrá funcionalidades añadidas sobre el partido que ha jugado y sobre los demás jugadores del partido.

En resumen se pueden diferenciar tres usuarios principales, que son el Usuario Anónimo, Administrador y Usuario Básico. Luego un Usuario Básico a mayores puede ser Usuario Creador y Usuario Jugador, con funcionalidades añadidas.

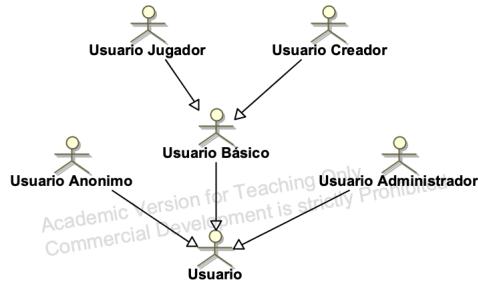


Figura 4.1: Relaciones de herencia entre actores

4.2 Requisitos

4.2.1 Requisitos del Cliente

Uno de los pasos que debemos realizar antes del desarrollo para describir el comportamiento del sistema es la Especificación de Requisitos (ERS). A continuación se detallan los distintos requisitos del cliente agrupados por tipo de funcionalidad.

Autentificación

- Es obligatorio que todos los usuarios se autentiquen para poder llevar a cabo cualquiera de las funcionalidades que ofrece la aplicación.
- Los Usuarios Anónimos podrán registrarse o loguearse, mientras que el Usuario Básico y el Administrador podrán hacer logout.

Deportes

- El Administrador podrá dar de alta deportes.
- Se podrán visualizar los detalles del deporte.
- Se podrán editar las localizaciones donde se juega ese deporte, el ícono del deporte y los componentes de entrada y visualización del deportes necesarios para el renderizado de pantallas en la parte del cliente del formulario de los resultados del deportes y sus detalles finales.

Localizaciones

- El Administrador podrá dar de alta localizaciones a través de un mapa.
- Se podrán visualizar los detalles de la localización.
- Se podrá editar el nombre de la localización y las coordenadas de esta a través del mapa.

Equipos

- Un Usuario Básico podrá dar de alta en el sistema nuevos equipos externos.

Usuarios

- El Administrador podrá eliminar usuarios de la aplicación.
- Cada usuario podrá ver sus datos personales.
- Cada usuario podrá editar cualquier información personal suya, la foto de perfil, la contraseña y los equipos a los que pertenece externamente fuera de la aplicación.
- Podrán buscar a otros usuarios gracias a su nombre y apellidos.
- En el perfil de cada uno se podrá observar tanto los próximos partidos que va a jugar, como los organizados, jugados, comentarios de otros jugadores sobre él y los partidos que la aplicación te recomendará según los partidos que jugaste en un pasado. Estas recomendaciones serán en base a los deportes, ubicaciones o usuarios con los que jugaste y poseen una buena valoración.
- Para cada usuario se calculará la experiencia media que posee en función de las valoraciones que ha recibido en anteriores partidos.
- Tendrán la posibilidad de empezara seguir a otros usuarios, por lo que cada usuario de la aplicación tendrá una lista de seguidores y seguidos.
- Podrá enviar mensajes directos.
- Observar las actividades de sus seguidos en un Tablón, donde se notificará cuando alguien que sigue crea un partido, se apunta, se desapunta...

Partidos

- Se podrán crear partidos
- Se podrán visualizar los partidos tanto en un mapa como en un calendario.

- Se podrán filtrar partidos.
- Se podrá ver el detalle del partido, es decir, su información básica, los jugadores apuntados a él y hasta su información meteorológica.
- Se podrá cancelar un partido, avisando a través de correo electrónico a todos los jugadores o a las personas que tienen marcado este partido para que se les notifique de cualquier cambio.
- Marcar un partido para recibir notificaciones sobre él, por ejemplo quién se apunta al partido, quién se desapunta...
- Completar los resultados finales del partido por parte del creador.
- Todos los jugadores de un partido podrán valorar el partido una vez finalizado, dándole una nota del 1 al 5.
- Cualquier usuario Básico podrá comentar en el foro de un partido.

Jugadores

- Se crearán jugadores una vez que un usuario se apunte a un partido.
- Se podrán valorar jugadores. La valoración consistirá en una nota del 1 al 5 y un comentario.

4.2.2 Requisitos del Sistema

A partir de los requisitos del cliente, detallaremos en esta sección los requisitos con suficiente detalle, describiendo la secuencia de iteraciones entre un actor y el sistema, como para que se puedan implementar. En nuestro caso disponemos de los actores citados en la Sección 4.1.

R1-Usuario Anónimo

Es aquel usuario que accede a la página Web y solo tiene acceso a la página para **Loguearse** (R1.1) o **Registrarse** (R1.2).

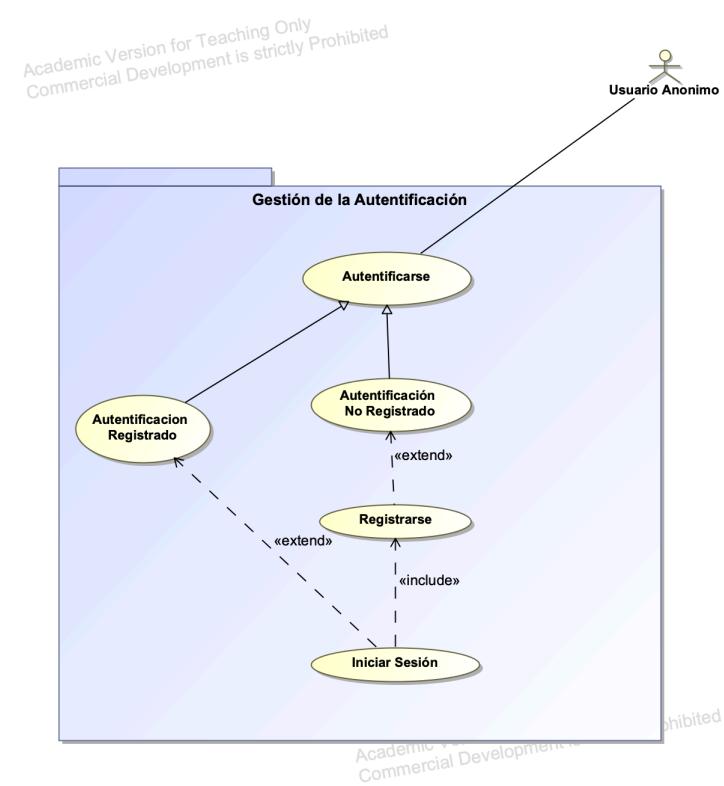


Figura 4.2: Caso de Uso: Usuario Anónimo

R2-Usuario Administrador

Este actor es el encargado de gestionar la aplicación, este por defecto lo creamos nosotros al dar de alta el sistema. Este administrador cuenta con actividades de gestión y no tiene acceso a ninguna de las funcionalidades de un Usuario Básico. Para acceder debemos ingresar para con un login y password de un Usuario Administrador.

R2.1-Gestión de Deportes:

El administrador podrá dar de alta nuevos deportes en el sistema como darlos de baja y editar los existentes, tanto las localizaciones donde se puede jugar a ese deporte, como los nombres de los componentes de visualización de los deportes en la parte cliente, como el icono de este.

R2.2-Gestión de Localizaciones:

El administrador podrá dar de alta localizaciones apoyándose en un mapa y darlas de baja en el sistema. También podrá editarlas, únicamente su nombre o sus coordenadas geográficas.

R2.3-Gestión de Usuarios:

Finalmente el administrador tendrá la posibilidad de observar los usuarios vigentes en la aplicación y dar de baja usuarios.

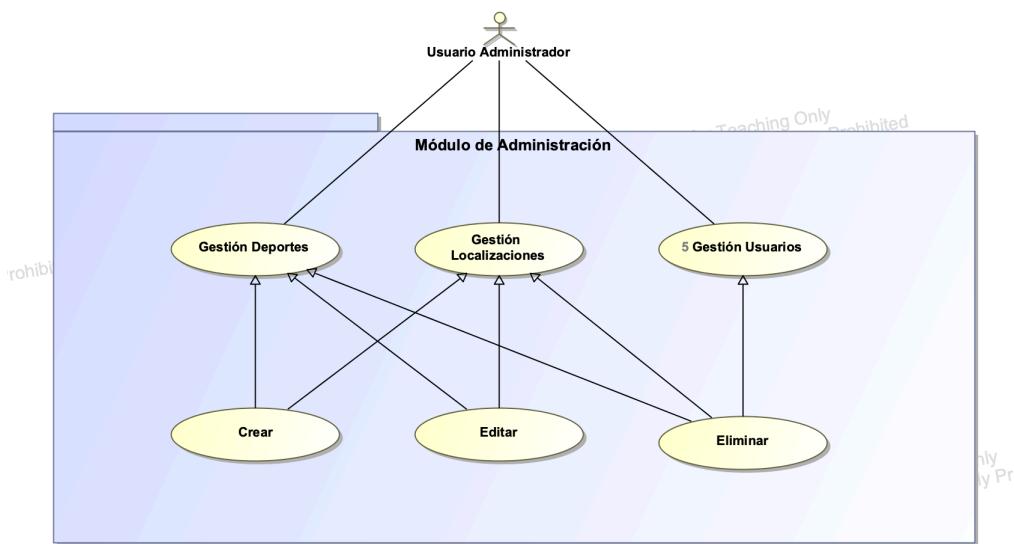


Figura 4.3: Caso de Uso: Usuario Administrador

R3-Usuario Básico, Creador y Jugador

Estos tres usuarios pueden hacer lo mismo, pero el Usuario Creador y el Usuario Jugador tienen actividades adicionales que pueden llevar a cabo, mientras que el Usuario Básico no. Como son muchas funcionalidades lo vamos a dividir en cinco grupos bien diferenciados con sus respectivos casos de uso.

R3.1-Crear Partido

En este caso de uso el usuario ha de seleccionar, para poder **dar de alta en el sistema un evento deportivo**, un tipo de deporte, una localización, la fecha y hora de inicio y fin, y por último un máximo y mínimo de jugadores. Una vez seleccionada una localización, el **calendario** que se dispone a modo de ayuda nos va a ir mostrando los partidos que se van a celebrar en esa localización a lo largo de los días. También disponemos de **información meteorológica** de la localización elegida.

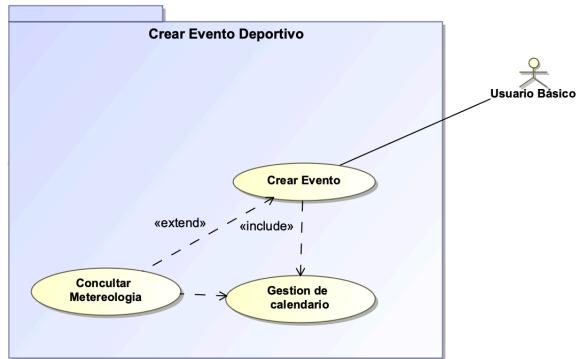


Figura 4.4: Caso de Uso Crear Partido: Usuario Básico

R3.2-Gestión Partidos

Este es un módulo de gestión bastante completo, hay que tener en cuenta que muchos de los casos de uso presentes dependen mucho del 'estado' del partido. Además, todos los casos de uso podrán ser llevados a cabo por cualquiera de los tres actores, excepto los marcados directamente con su actor.

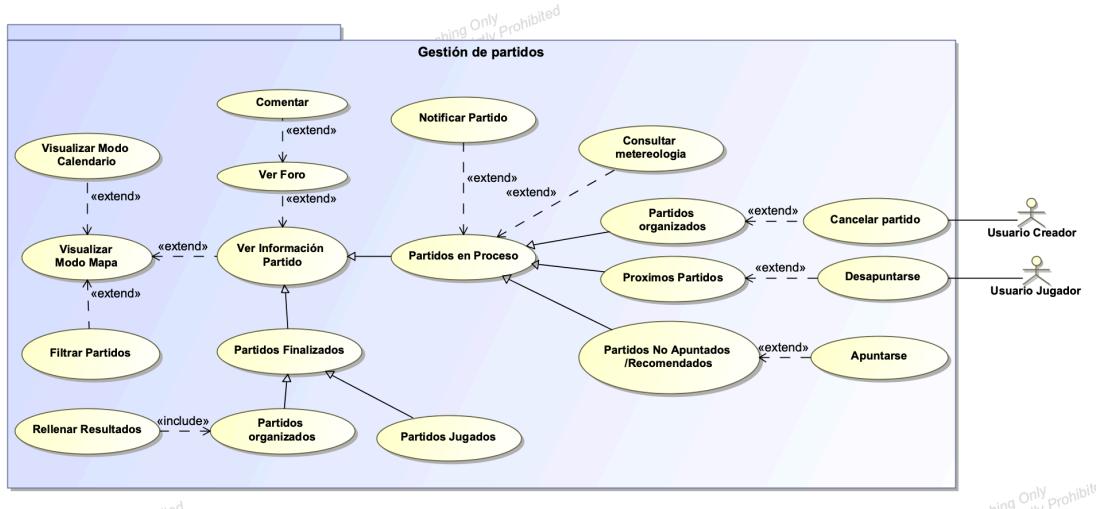


Figura 4.5: Caso de Uso Gestión Partidos: Usuario Básico/Creador/Jugador

R3.2.1-Búsqueda de Partidos

Primero, la aplicación nos facilita la **búsqueda de partidos**, nos ofrece dos vistas distintas, en modo **mapa o en calendario**. Además hay que destacar que nos permite aplicar **filtros** por si queremos una búsqueda más concreta, ya sea por creador del partido, por tipos de deportes, por dificultad o por media de edad.

R3.2.2-Detalles del Partido

Cuando accedemos a un partido, este en el estado en el que este, podemos observar la **información del partido**, que esto incluye tanto la información básica como los jugadores apuntados al partido.

R3.2.2.1-Partido en Proceso

Si el partido está aun en proceso, podremos **consultar la meteorología, marcar el partido para que nos notifiquen** de cualquier cambio en él. En el caso de ser el Usuario Creador podrá decidir **cancelar el partido**, como consecuencia se enviará un correo avisando a todos los jugadores y a las personas que lo tienen el partido marcado para que se les notifique de cualquier cambio. En el caso de ser un Usuario Básico, se nos da la posibilidad de **apuntarnos al partido**, en cambio, si somos un Usuario Jugador, tendremos la opción de **desapuntarnos**.

R3.2.2.2-Partido Finalizado

Si el partido está finalizado si fuimos el creador (Usuario Creador), debemos **rellenar los resultados**, por lo tanto este partido se incorporará a partidos pendientes de llenar resultado. Si fuimos jugadores (Usuario Jugador) de ese partido, posteriormente debemos **valorar tanto a los jugadores como al partido**.

R3.3-Gestión de Valoraciones

Será realizada únicamente por el Usuario Jugador, es decir que cuando este usuario haya finalizado un partido, este pasa al apartado de partidos pendientes de valorar. **La valoración se hace de los partidos y jugadores** y consiste en darle un valor numérico entre 1 al 5 tanto al partido como a los demás jugadores, además, junto a la nota numérica, podemos dejarle a los jugadores un **comentario**.

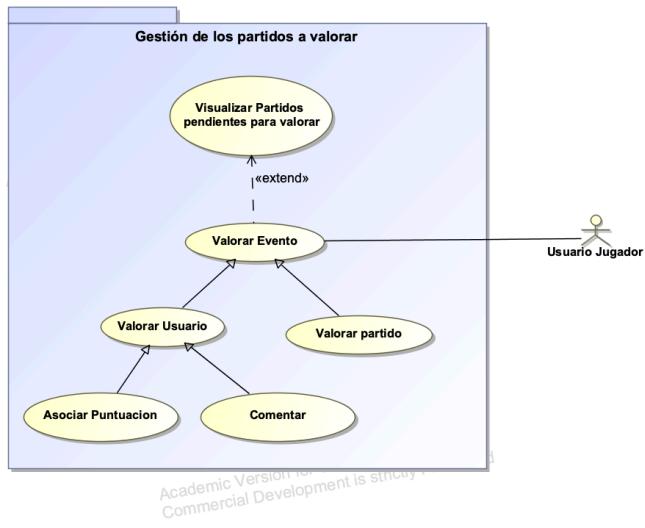


Figura 4.6: Caso de Uso Gestión Valoraciones: Usuario Jugador

R3.4-Gestión de Perfiles

En la Figura 4.7, reflejamos la parte del perfil del usuario. En la parte privada del usuario podrá tanto ver sus **datos personales**, como **editarlos**, no solo la información básica, si no que podrá modificar la contraseña, la foto de perfil e incluso añadir, eliminar y dar de alta en el sistema equipos externos de los que forma parte. Por otra lado, tenemos el perfil público donde se muestra nombre, ciudad a la que pertenece y experiencia. El usuario podrá observar los **comentarios anónimos**, los **próximos partidos** que tiene por jugar, **los organizados**, **los jugados** y **partidos recomendados** por parte de la aplicación en función de los partidos jugados. Hay que destacar que el resto de usuarios van a poder observar, aparte del nombre, ciudad y experiencia, los comentarios, próximos partidos, y los organizados por esa persona.

R3.5-Gestión social

En la Figura 4.8 se consideró representar así los casos de uso ya que no se pueden acceder a través de ningún otro, si no que muchas son accesibles mediante el menú. Cada usuario tendrá una **lista de seguidos y seguidores**. Desde esta lista podrá acceder al perfil de cada uno de ellos. Tiene la opción de **seguir** o **dejar de seguir**, además si está siguiendo a una persona, puede **marcar a ese usuario para que se le notifique** a través de correo de las actividades que realiza. Por otro lado tenemos la **mensajería**, nos permite entablar conversaciones con cualquier usuario. Finalmente en el **Tablón de Actividades** se mostrarán las actividades de todas las personas que el usuario sigue.

4.2. Requisitos

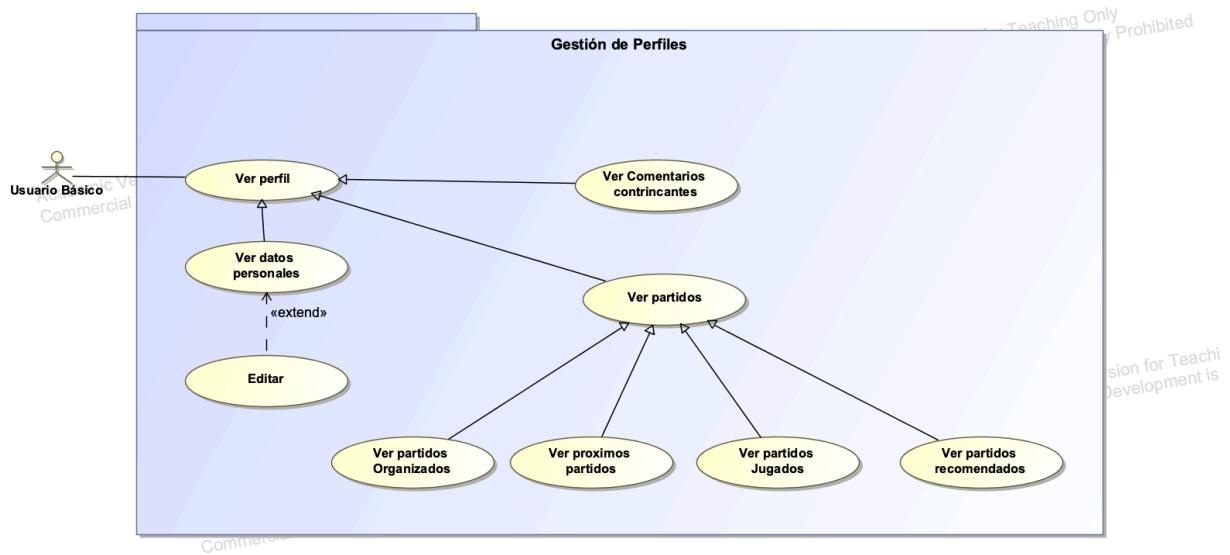


Figura 4.7: Caso de Uso Gestión Perfiles: Usuario Básico

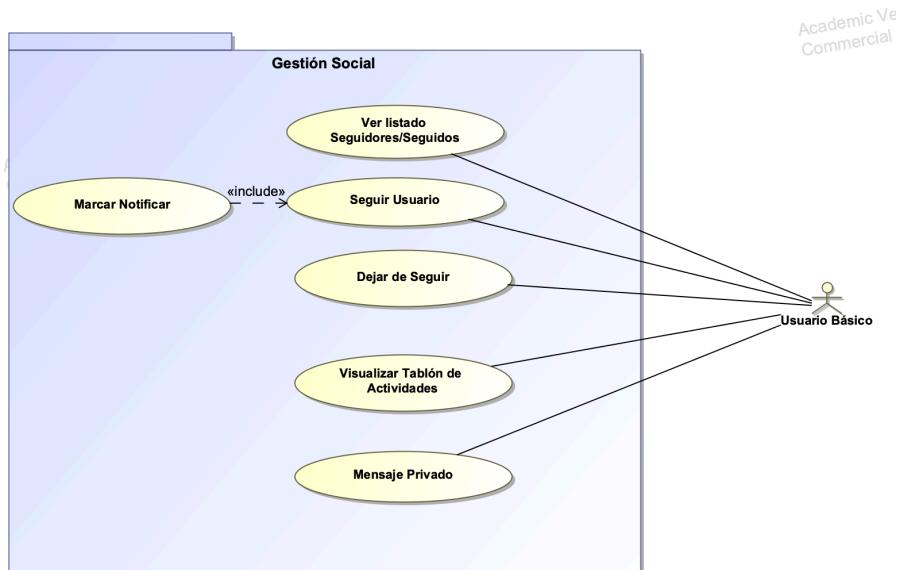


Figura 4.8: Caso de Uso Gestión Social: Usuario Básico

4.3 Arquitectura del sistema

Tenemos una arquitectura en tres capas (modelo, controlador, vista) orientada a servicios, ya que utilizamos un servicio externo para la carga de datos iniciales. Tenemos un Cliente Web que realizará peticiones REST al Servidor Web. Solo se conoce la capa inmediatamente inferior consiguiendo abstracción, aislamiento de cambios, incremento de escalabilidad, tolerancia a fallos y rendimiento.

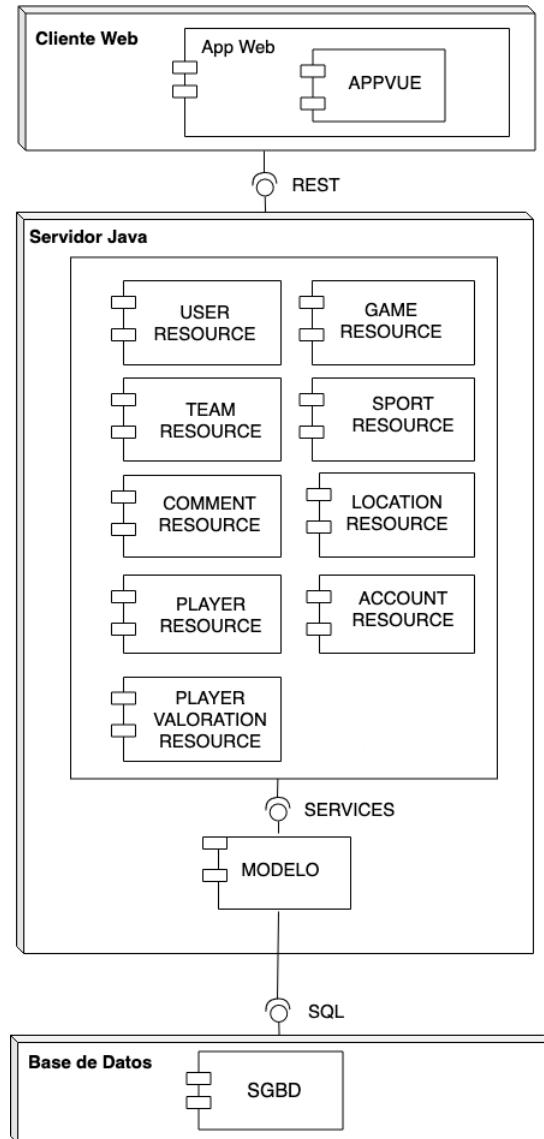


Figura 4.9: Arquitectura del sistema

4.4 Interfaz de usuario

Todas las pantallas tienen un menú de navegación, un pie de página y luego el contenido. Este contenido es el que va a ir variando dependiendo de la ruta a la que accedamos. Este tema lo trataremos en la Sección 5.3.2.

4.4.1 Componentes y Navegación

En este apartado se mostrarán todos los componentes que conforman el cliente y la navegación a través de ellos. No mostraremos los componentes de “LoadingPage” y “NotFound” ya que no poseen ninguna funcionalidad importante, ni el “AppVue”, ya que únicamente añade la barra de navegación y el pie de página a todos los demás componentes.

Componentes Administración	Descripción
Sports	En este componente se visualizará la lista de deportes. Al pulsar en “Nuevo Deporte” se cargará a la derecha el componente “SportsForm”
SportsDetail	En este componente se visualizarán los detalles del deporte y se podrán editar estos datos dándole a “Editar”, cambiándose el componente por el “SportsForm”.
SportsForm	Es el componente utilizado para la creación de nuevos deportes o la edición de los existentes.
LocationsDetail	En este componente se visualizarán los detalles de la Localización y se podrán editar estos datos dándole a “Editar” emitiéndose una señal al componente padre “Map”, recargándose una parte del componente del parent solo apta para la edición.
Map	En este componente se visualiza un mapa con todas las localizaciones vigentes (GIS), en el momento de darle al botón de “Nueva Localización” se recargará una parte de este componente solo apta para cuando se quiera dar de alta localizaciones.
UserList	Es un componente que muestra la lista de todos los usuarios existentes en la aplicación y da la posibilidad de darlos de baja.

Cuadro 4.1: Componentes de las pantallas del Administrador y Navegación

CAPÍTULO 4. ANÁLISIS

Componentes Usuario	Descripción
CommentGame	Es un componente para visualizar el foro de un juego. Se observará en los datos básicos del partidos, los comentarios y se podrá crear nuevos comentarios desde este componente.
Recomendations	Es un componente que se encarga de filtrar las recomendaciones según Deporte, Localización y Jugadores. Depende de lo que elijamos el componente “GameList” mostrará en pantalla unos partidos recomendados u otros.
ActivitiesFollowed	En este componente se visualiza el Tablón de Actividades precargando dentro de este componente el componente “GameList”, es decir, se van a mostrar en una lista las diferentes actividades que realizaron las personas a las que seguidos. Dependiendo del tipo de actividad, si seleccionamos en ella nos llevará a el componente “GameDetail”, si el usuario ha creado o se ha apuntado a un partido, o a el componente “CommentGame” si ha comentado en un partido.
Calendar	Es un componente para visualizar el calendario, se utiliza tanto en el componente “GameCreate” para visualizar los partidos creados y ver la disponibilidad que tenemos como en el componente “Game” para buscar partidos por fechas concretas. En ambos casos, cuando pinches en un evento te lleva a su detalle, al componente “GameDetail”.
FutbolForm	Es el componente que se carga al clickear sobre algún partido de fútbol o de baloncesto que salga en el modal del icono de la exclamación en el componente “MenuBar”. Se utiliza para llenar los resultados finales con pequeños cambios dependiendo de cada deporte, una vez enviado el formulario nos llevará al componente “FutbolResult”.
FutbolResult	Es el componente que visualiza los resultados finales de un partido de fútbol o de Baloncesto con pequeños cambios dependiendo de cada deporte. Cabe comentar que cuando pulsamos en un jugador, nos enviará al perfil público de esa persona, al componente “GameUser”.

Cuadro 4.2: Componentes de las pantallas del Usuario y navegación [1]

Componentes Usuario	Descripción
GameCreate	Es el componente que nos permite crear partidos. A la derecha de todo se carga a la vez el componente “Calendar”. Cuando elegimos una localización nos da la opción de ver la previsión del tiempo para esa localización cargando el componente “Weather”. Finalmente cuando le damos a crear nos enviará al componente “GameDetail”.
GameDetail	Es el componente que nos permite ver los detalles del partido, tanto la información básica como sus jugadores. Si pulsamos sobre el botón de la nube nos permitirá observar la previsión metereológica cargando el componente “Weather”, podemos pulsar en la campanita para activar las notificaciones del partido que nos llegarán a través de correo electrónico. Luego tenemos el botón de comentarios que nos enviará al componente “CommentGame”. Finalmente, cuando pulsamos en un jugador, nos enviará al perfil público de esa persona, al componente “GameUser”.
GameList	Es el componente que va a utilizar tanto el componente “GameUser” para mostrar los partidos próximos, organizados, jugados y recomendados, como los comentarios anónimos que dejan otros jugadores sobre ese usuario. Cuando se pulse en cualquier partido se pasará al componente “GameDetail” en caso de ser un partido aun vigente o un partido terminado pero al que aún su creador no completó los resultados”, al componente “TennisResult” o “FutbolResult” en caso de partidos finalizados.
GameUser	Es el componente que va a visualizar el perfil público de la persona, tanto sus datos personales como los partidos próximos, jugados, organizados y recomendados, junto con los comentarios de otras personas hacia él, gracias al componente “GameList”. Y se podrá acceder a los detalles de los partidos pulsando en cada uno, la navegación de componentes en este caso está descrita en el anterior componente.
ValorationGame	Es el componente que se carga cuando pulsamos algún partido que salga en el modal que se abre al pulsar la campanita del componente “MenuBar”. Nos permite tanto valorar el partido como a los jugadores, una vez valorado nos redirige al componente “GameDetail” en caso de ser un partido sin resultados aún rellenados por el creador, o al componente “TennisResult” o “FutbolResult” dependiendo del deporte, si es tenis, pádel, fútbol o baloncesto.

Cuadro 4.3: Componentes de las pantallas del Usuario y navegación [2]

Componentes Usuario	Descripción
Weather	Este componente se encarga de realizar una llamada a la API de DarkSky a través de una clave. De esta forma podremos obtener la previsión meteorológica de cualquier localización a lo largo de una semana únicamente gracias a la latitud y longitud. Se utiliza este componente tanto en el componente “GameCreate” como en el “GameDetail”.
UserDetail	Es un componente que se encarga de visualizar toda la información personal del usuario. Cuando pulsas editar te recarga el componente “Registro” para poder llevar a cabo esa edición.
TennisForm	Es el componente que se carga al clickear sobre algún partido de tenis o de pádel que salga en el modal del ícono de la exclamación en el componente “MenuBar”. Se utiliza para llenar los resultados finales con pequeños cambios dependiendo de cada deporte, una vez enviado el formulario nos llevará al componente “TennisResult”.
TennisResult	Es el componente que visualiza los resultados finales de un partido de tenis o de pádel con pequeños cambios dependiendo de cada deporte. Cabe comentar que cuando pulsamos en un jugador, nos enviará al perfil público de esa persona, al componente “GameUser”.
Game	En este componente se visualizarán todos los partidos, ya sea en mapa o en calendario. Cuando pulsamos en un marcador del mapa se nos abre una lista con todos los partidos vigentes en esa ubicación. Si pulsamos en un partido de la lista o en un evento del calendario nos lleva a el componente “GameDetail” para ver su detalle. En este componente cabe destacar que se puede realizar un filtrado de los partidos sobre el mapa.

Cuadro 4.4: Componentes de las pantallas del Usuario y navegación [3]

Componentes Comunes	Descripción
Home	Es la pantalla de inicio, no ejerce ninguna funcionalidad en concreto, simplemente visualiza unas imágenes con un pequeño resumen de que es la aplicación. Se puede acceder a él clickeando en el logo de la aplicación arriba a la izquierda.
Login	Es el componente utilizado para el inicio de sesión. Se accede cuando se clickean en “Login” arriba a la derecha en el componente “MenuBar”. Desde este componente también podemos dirigirnos al componente de “Registro” clickeando en su respectivo botón.
MenuBar	Visualiza lo que sería la barra de navegación de la aplicación. Muestra tanto las opciones de la barra, las del menú del usuario. En el caso de estar logueado como usuario, a mayores muestra las notificaciones de partidos pendientes de llenar resultados, partidos pendientes de valorar y la sección de mensajería. Este componente cambia dependiendo de los permisos que se tenga.
Register	Este componente permite a los usuarios crearse una cuenta en la aplicación para poder disfrutar de todas las funcionalidades que ofrece esta. Este componente también se reutiliza para la edición de los datos personales del usuario. Se puede acceder desde el componente “Login” pulsando en “Registro”, como desde el componente “UserDetail” pulsando en “Editar”.

Cuadro 4.5: Componentes comunes a todos los usuarios

4.4.2 Mockups Componentes

En este capítulo se van a mostrar los mockups de los componentes más importantes. Estas maquetas sirve para tener plasmada una primera idea de lo que queremos hacer y así poder ir planificando nuestro trabajo en función de estos mockups.

En la Figura 4.10 y 4.11 se muestra el componente Game. La Figura 4.10 consiste en un mapa con los partidos vigentes a los que podemos acceder a su pantalla de detalles, al componente GameDetail, pulsando en el pop up que saldrá al clickear en alguno de los marcadores del mapa. Desde la opción “Filtrar por” se abrirá un menu con todos los filtros que podremos aplicar a los partidos. Encima del mapa tenemos las opciones de visualización de los partidos, en mapa o calendario, siendo esta última opción la Figura 4.11. En esta última figura se accederá a los detalles de los partidos, al componente GameDetail, pulsando en los eventos del calendario. Se puede observar que si accedemos al icono del usuario se abrirá un menú vertical en la parte superior derecha que nos permitirá acceder a las diferentes secciones. En concreto este menú permite al usuario acceder a los Partidos (Figuras 4.10 y 4.11), al Tablón de Actividades, Crear un Evento (Figura 4.13), Ver Perfil (Figura 4.14) y Cerrar Sesión, es decir, a los componentes Games, ActivitiesFollowed, GameCreate y GameUser. En el caso de ser administrador este menú nos dará acceso a Deportes, Localizaciones, Usuarios y Cerrar Sesión, es decir, a los componentes Sports, Map y UserList.

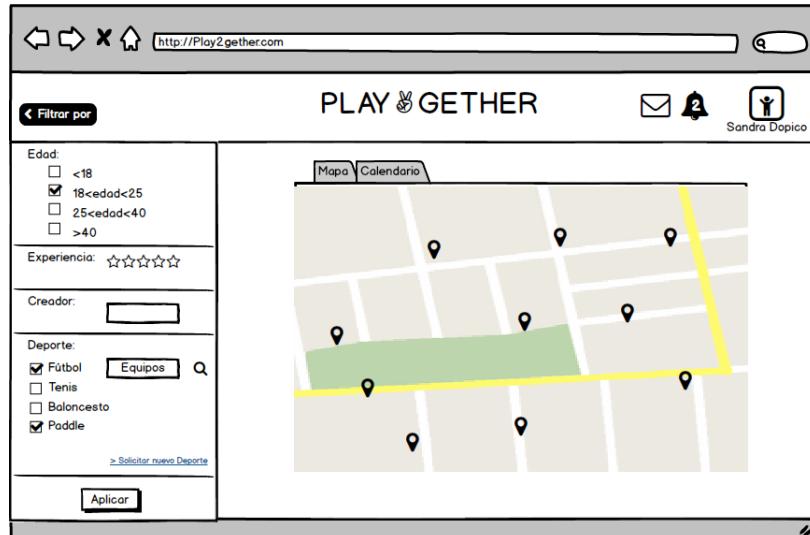


Figura 4.10: Componente Game: Mapa

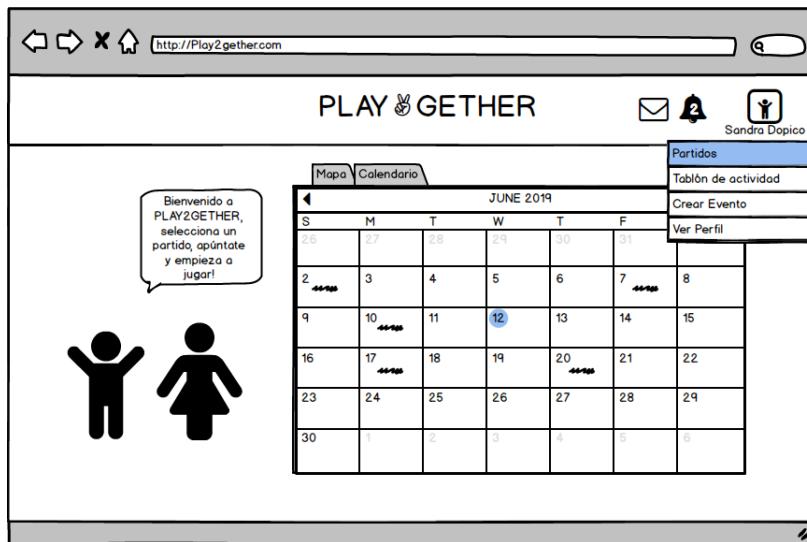


Figura 4.11: Componente Game: Calendario

En la Figura 4.12 se muestra el componente GameDetail. Se puede visualizar los jugadores que hay apuntados, a la izquierda la información básica del partido y un botón a la derecha donde poder apuntarnos al partido. Pulsando en cualquier jugador nos llevaría al perfil público del usuario, al componente GameUser. En cuanto a los iconos arriba a la izquierda, el primero por la izquierda sería para recibir notificaciones del partido por si alguien se apunta, se desapunta o se cancela, y el segundo ícono para acceder al foro del partido, al componente CommentGame.

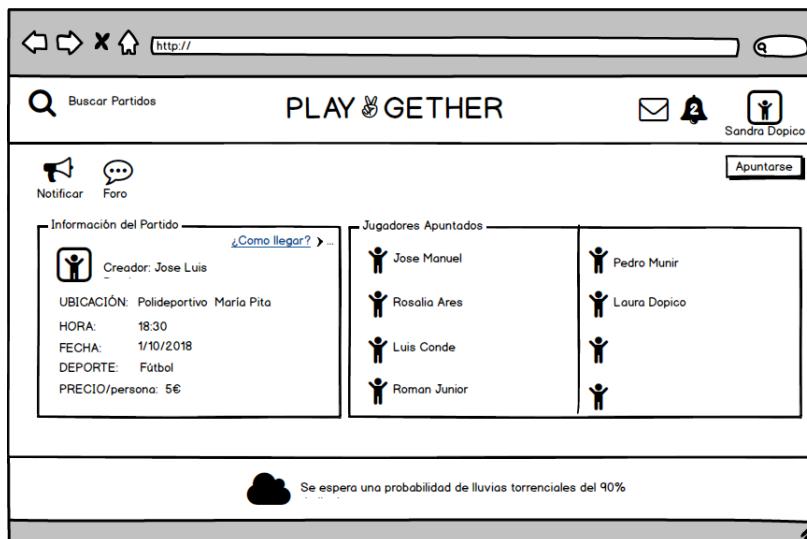


Figura 4.12: Componente GameDetail

En la Figura 4.13 se muestra el componente GameCreate, que nos permite crear partidos de diferentes tipos de deportes en diferentes ubicaciones. Se puede observar a la derecha un calendario que mostrará los eventos vigentes a día de hoy, pudiendo seleccionar la fecha y día a través de este. Una vez seleccionada la localización donde tendrá lugar el evento, en el calendario únicamente aparecerán los eventos que tendrán lugar en dicha ubicación. Finalmente hay que elegir un máximo y mínimo de participantes, para impedir que se apunten más personas de las permitidas y para cancelar el partido si no se llega al mínimo de jugadores necesarios. Una vez creado el partido la aplicación nos llevará al componente GameDetail, la Figura 4.12.



Figura 4.13: Componente GameCreate

En la Figura 4.14 se muestra el componente GameUser, lo que sería el perfil público de un usuario. En el podemos ver la información básica del usuario a la derecha, tanto su ciudad, nombre, experiencia, su foto de perfil, hasta tendremos acceso a un listado de Seguidos y Seguidores, donde clickeando en un usuario, iremos al perfil público de esa persona. En la parte superior podemos observar un menú horizontal que nos permitirá acceder a las diferentes secciones. En concreto, este menú permite al usuario acceder a sus partidos organizados, a los próximos partidos, a los jugados, a los recomendados y finalmente a los comentarios que le deja la gente después de terminar un partido. En la parte de arriba de todo de la pantalla aparecen unos iconos comunes a todas las pantallas, que serían los mensajes y en el ícono de la campana tendremos todos aquellos partidos pendientes de ser valorados, que si pulsamos en alguno de ellos nos llevará al componente ValorationGame.

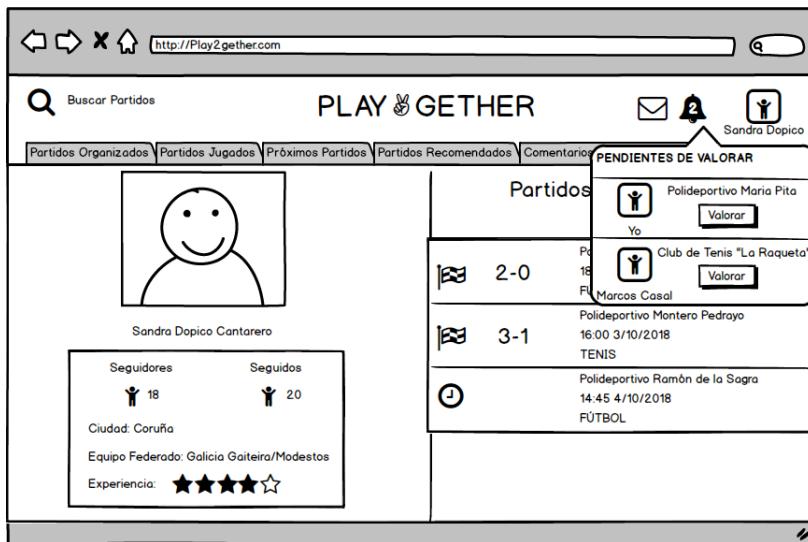


Figura 4.14: Componente GameUser

4.5 Modelo conceptual de datos

En esta sección se va a hablar de la capa de modelo, la encargada de implementar los casos de uso y la lógica de negocio. Para ello vamos a mostrar el diseño de la BD, con sus entidades y atributos que forman la base de esta aplicación.

User: Representa los elementos comunes de todos los usuarios de la aplicación, tanto Administradores como Usuarios Básicos.

- *idUser*: Identificador del usuario.
- *login*: Login del usuario.
- *password*: Contraseña del usuario.
- *email*: Correo electrónico del usuario.
- *name*: Nombre del usuario.
- *surname1*: Primer apellido del usuario.
- *surname2*: Segundo apellido del usuario.
- *authority*: Permisos del usuario.

UserAuthority: Es un enumerado que representa los permisos de los usuarios.

- *USER*: Para los Usuarios Básicos. Tiene acceso a su información de usuario.
- *ADMIN*: Para los Usuarios Administradores. Se encarga de la gestión de los datos del sistema.

NormalUser: Usuario no Administrador

- *city*: Ciudad de residencia del usuario.
- *birthday*: Fecha de nacimiento del usuario.
- *experience*: Media de las valoraciones recibidas en los partidos jugados por los el resto de jugadores.

AdminUser: Usuario Administrador

SocialRelationship: Representa las relaciones entre usuarios, de amistad o de bloqueo

- *idSocial*: Identificador de la relación entre 2 usuarios .
- *lastUpdate*: Fecha de alta de la relación.

SocialFriendship: Representa una relación de amistad entre 2 usuarios.

- *notification*: Indica si queremos recibir notificaciones sobre las actividades que realiza esta amistad.

Team: Representa los equipos externos de los que forman parte los usuarios.

- *idTeam*: Identificador del equipo.
- *name*: Nombre del equipo.

Sport: Representa los deportes que hay dados de alta en la aplicación.

- *idSport*: Identificador del deporte.
- *type*: Indica el tipo de deporte que es, sea Fútbol, Padel, Tennis, Baloncesto...
- *componente de Entrada*: Indica el nombre de la plantilla para el relleno de resultados.
- *componente de Visualización*: Indica el nombre de la plantilla para visualizar los resultados del partido.

Location: Representa las localizaciones donde se van a jugar los eventos deportivos.

- *idLocation*: Identificador de la localización.
- *name*: Nombre de la localización.
- *latitud*: Coordenada que mide el ángulo entre cualquier punto al ecuador.
- *longitud*: Coordenada que mide el ángulo a lo largo del ecuador desde cualquier punto.

Game: Representa el evento deportivo que se va a realizar.

- *idGame*: Identificador del partido.
- *date*: Fecha en la que tendrá lugar el partido.
- *timeStart*: Hora de inicio del partido.
- *timeEnd*: Hora de finalización del partido.

- *maxPlayers*: Número máximo de jugadores permitidos.
- *minPlayers*: Número mínimo de jugadores necesarios para llevarse a cabo el partido.
- *result*: Resultado del partido en formato JSON para que sea utilizable para todo tipo de deportes.

Player: Representa a los jugadores que han decidido apuntarse a un evento deportivo

- *idPlayer*: Identificador del jugador.
- *valorationOfTheGame*: Valoración del juego por parte de ese jugador.
- *equipo*: Equipo del que forma parte dentro del partido (A o B).

PlayerValoration: Representa las valoraciones de cada uno de los jugadores.

- *idPlayerValoration*: Identificador de la valoración de un jugador.
- *valoration*: Valoración numérica entre 1 y 5.
- *review*: Comentario sobre el jugador.

Comment: Representa los elementos comunes tanto de los comentarios realizados en el foro del partido, como los mensajes directos.

- *idComment*: Identificador del comentario.
- *contentComment*: Contenido del comentario.
- *date*: Fecha de creación del comentario.

UserMessage: Representa los mensajes directos entre los usuarios.

- *viewed*: Estado del mensaje, si fue visto o no.

GameMessage: Representa los comentarios realizados en el foro de un partido.

CAPÍTULO 4. ANÁLISIS

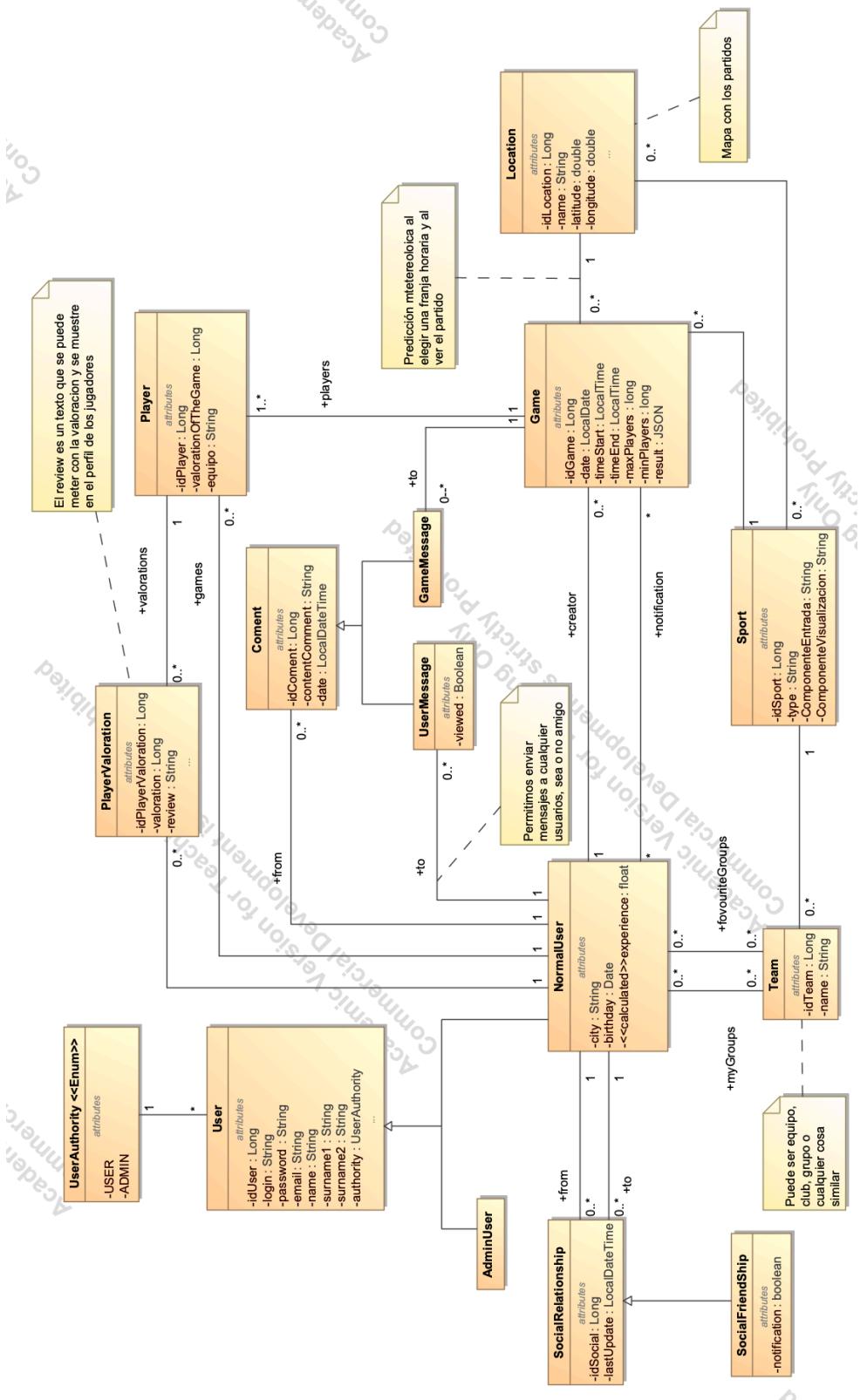


Figura 4.15: Diagrama de Clases

Capítulo 5

Diseño

En este capítulo se expondrán los resultados en cuanto a diseño. Para ello, se explicará la arquitectura que seguirá el proyecto y el diseño de la aplicación, tanto la parte front-end como la back-end.

5.1 Arquitectura tecnológica del sistema

En esta sección se va a realizar una descripción tecnológica del sistema, para ello se utilizará la Figura 4.3 de la parte de Análisis señalando en ella las tecnologías utilizadas en cada componente.

Primero, en la capa inferior de la arquitectura, en la Base de Datos, se utilizará PostgreSQL, un sistema de gestión de bases de datos relacional orientado a objetos y de código abierto. Para la conexión de la BD con la capa superior se utilizará JDBC (Java Database Connectivity), es una API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java utilizando el dialecto SQL del modelo de base de datos que se utilice.

Para la conexión entre la capa del modelo y el servidor se utilizarán los DAOs (Data Access Object) como objeto de acceso a datos, suministra una interfaz común entre la aplicación y uno o más dispositivos de almacenamiento de datos. Estos DAOs se comunican con los servicios implementados en Spring e Hibernate.

Para conectar el servidor Web con el cliente Web se hará uso de REST. Este cliente está implementado con Vue.js y hará uso de Leaflet, una librería de JavaScript para crear mapas interactivos y Bootstrap, un framework CSS.

Esta aplicación se dividirá en dos partes. La parte front-end, el cliente, es la parte con la que interactúan los usuarios y por otro lado está la parte back-end, el servicio, la parte que el usuario final no puede ver.

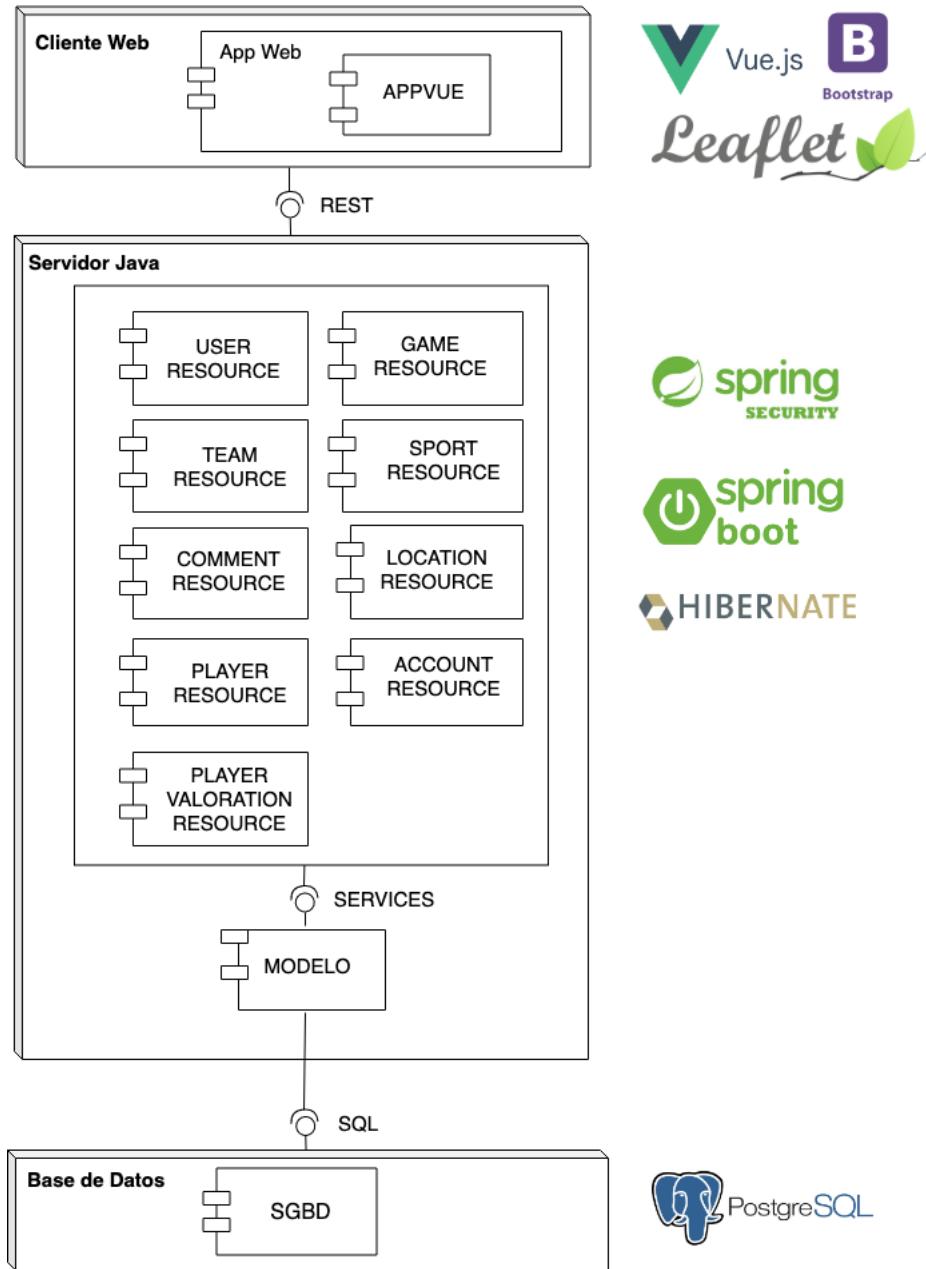


Figura 5.1: Arquitectura tecnológica del sistema

5.2 Back-end

Esta parte de la aplicación es el servicio REST. Se divide en capa de persistencia, de acceso a los datos, los servicios y finalmente los controladores a través de los cuales el Cliente Web se comunica a través de peticiones con el Servidor Web.

5.2.1 DAOs

Data Access Object (DAO), permite separar la lógica de acceso a datos de los Business Objects, de tal forma que el DAO encapsula toda la lógica de acceso de datos al resto de la aplicación. De esta forma, el DAO proporcionará los métodos necesarios para insertar, actualizar, borrar y consultar la información (métodos CRUD), mientras que por otro lado la capa de negocio solo se preocupará por la lógica de negocio y utilizará los DAOs para interactuar con la fuente de datos.

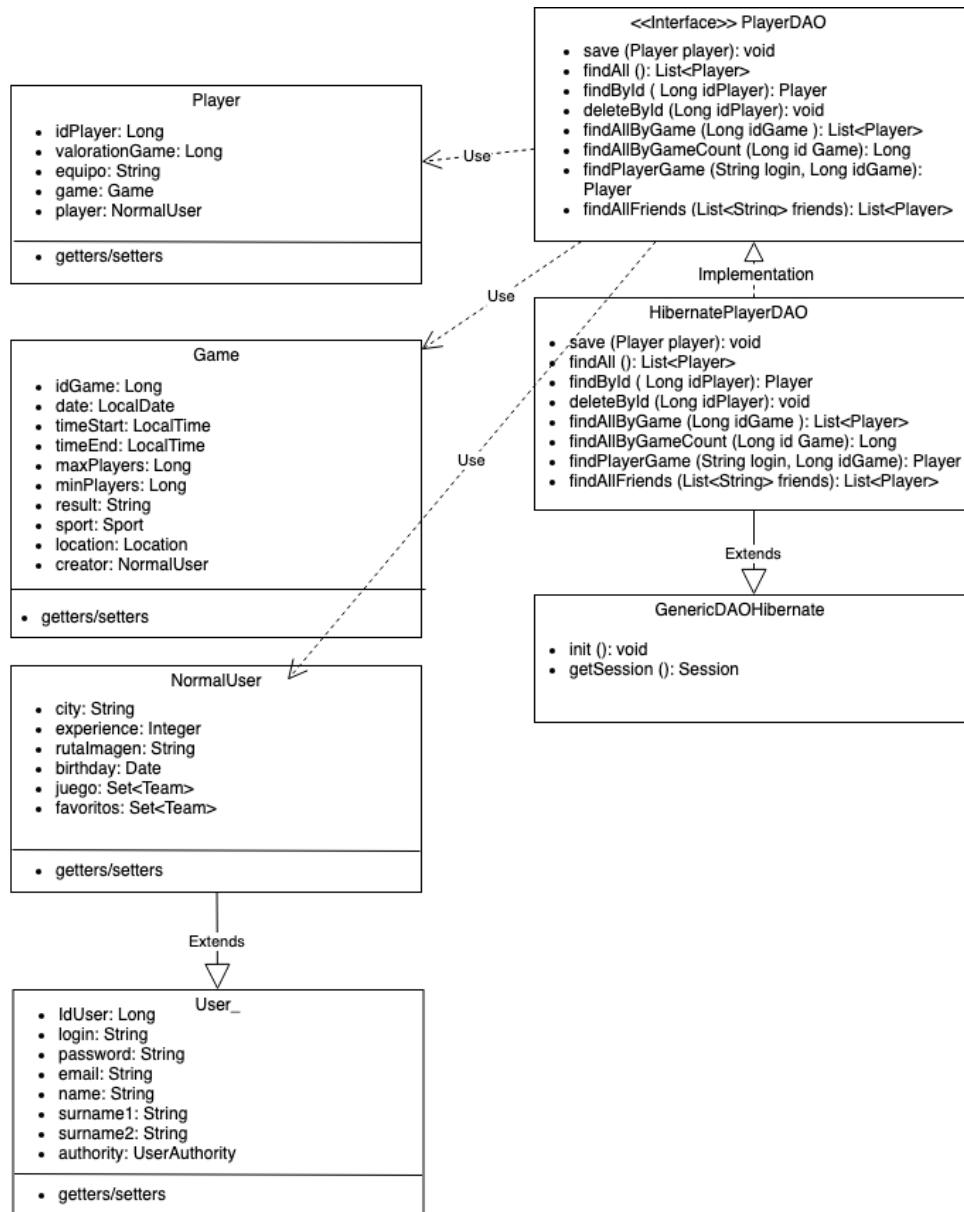


Figura 5.2: Diagrama PlayerDAO

En el proyecto, si vamos a la carpeta `es.udc.lbd.asi.restexample.model.repository` se encontrará el DAO `es.udc.lbd.asi.restexample.model.repository.PlayerDAO` y su implementación `es.udc.lbd.asi.restexample.model.repository.PlayerDAOHibernate`.

A mayores de los métodos CRUD, ya mencionados, nuestros DAOs tienen otros métodos necesarios como buscar todos los jugadores de un partido, buscar el jugador concreto de un usuario para un partido, buscar el número de jugadores apuntados a un partido o buscar todos los jugadores de todas las personas a las que sigo.

5.2.2 Servicios

La capa de servicios consiste en la lógica que realiza las funciones principales de la aplicación, las operaciones de alto nivel. Para ello las operaciones llamarán a los DAOs que se encargarán del acceso a datos y luego en el servicio se utilizarán estos datos para ofrecer funcionalidades mucho más complejas y avanzadas.

En el proyecto, si vamos a la carpeta `es.udc.lbd.asi.restexample.model.service` se encontrará la fachada del Servicio `es.udc.lbd.asi.restexample.model.service.PlayerServiceInterface` y su implementación `es.udc.lbd.asi.restexample.model.service.playerService`. En nuestro caso se encontrarán varios servicios con sus respectivas operaciones a las que se podrá acceder desde el cliente a través de peticiones a los controladores. Los distintos servicios que se encontrarán son:



Figura 5.3: User Service

CAPÍTULO 5. DISEÑO

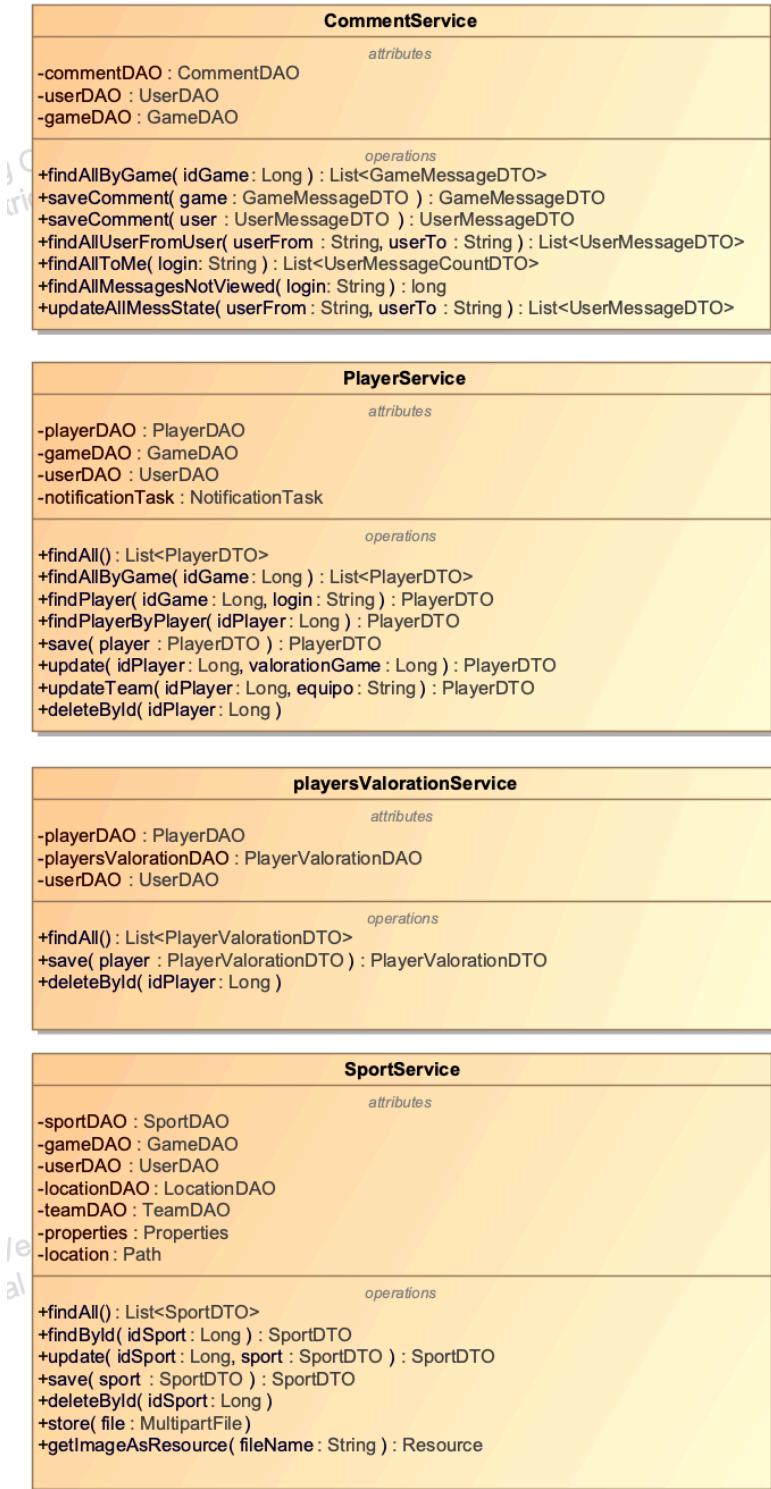


Figura 5.4: Comment, Player, PlayerValoration y Sport Service



Figura 5.5: Game, Location, SocialRelationship y Team Service

5.2.3 Controladores

Los controladores se sitúan en el paquete `es.udc.lbd.asi.restexample.web`. En esta sección se mostrarán sus respectivas operaciones REST y las URL necesarias para acceder a estas desde el cliente a través de peticiones REST.

CAPÍTULO 5. DISEÑO

URL	OPERACIÓN	ACCIÓN
/api/users/	GET	findAll
/api/users/normal	GET	findAllNormalUsers
/api/users/{login}	GET	findOneUserByLogin
/api/users/{login}/organizados	GET	findAllPartidosOrganizados
/api/users/{login}/recomendados	GET	findAllPartidosRecomendados
/api/users/{login}/proximos	GET	findAllPartidosProximos
/api/users/{idUser}	PUT	update
/api/users/{login}/password	PUT	updatePassword
/api/users/notifications/{login}/{idGame}/{bool}	PUT	updateNotifications
/api/users/activities/{login}	GET	findAllActividades
/api/users/notifications/{login}/{idGame}	PUT	getNotification
/api/users/{idUser}	DELETE	delete
/api/users/uploadFile	POST	loadImage
/api/users/imagenes/{login}	GET	getImagen

Cuadro 5.1: Acceso a UserResource

URL	OPERACIÓN	ACCIÓN
/api/social/{loginFrom}/{loginTo}	GET	findARelation
/api/social/follow/{login}/{type}	GET	findRelations
/api/social/friendShip	POST	save
/api/social/{loginFrom}/{loginTo}/{notification}	PUT	update
/api/social/friendShip/{loginFrom}/{loginTo}/{typeRelation}	DELETE	deleteRelationShip

Cuadro 5.2: Acceso a SocialResource

URL	OPERACIÓN	ACCIÓN
/api/teams	GET	findAll
/api/teams	POST	save

Cuadro 5.3: Acceso a TeamResource

URL	OPERACIÓN	ACCIÓN
/api/playersValoration	GET	findAll
/api/playersValoration	POST	save

Cuadro 5.4: Acceso a PlayerValorationResource

URL	OPERACIÓN	ACCIÓN
/api/sports/	GET	findAll
/api/sports/{idSport}	GET	findOne
/api/sports/{idSport}	PUT	update
/api/sports/organizados	POST	save
/api/sports/{idSport}	DELETE	delete
/api/sports/uploadFile	POST	loadImage
/api/sports/imagenes/{type}	GET	getImage

Cuadro 5.5: Acceso a SportResource

URL	OPERACIÓN	ACCIÓN
/api/comments/user/{userFrom}/{userTo}	GET	findAllUserFromUser
/api/comments/user/{login}	GET	findAllUserMessage
/api/comments/game/{idGame}	GET	findAllByGame
/api/comments/user/countMessages/{login}	GET	findAllMessagesNotViewed
/api/comments/user/{userFrom}/{userTo}	PUT	updateState
/api/comments/game	POST	saveCommentGame
/api/comments/user	POST	saveMessageUser

Cuadro 5.6: Acceso a CommentResource

URL	OPERACIÓN	ACCIÓN
/api/locations	GET	findAll
/api/locations/filter/{idSport}	GET	findAllSport
/api/locations/{idLocation}	GET	findOne
/api/locations/{idLocation}	DELETE	delete
/api/locations/{idLocation}	PUT	update
/api/locations	POST	save

Cuadro 5.7: Acceso a LocationResource

URL	OPERACIÓN	ACCIÓN
/api/authenticate	POST	authenticate
/api/register	POST	registerAccount
/api/account	GET	getAccount

Cuadro 5.8: Acceso a AccountResource

URL	OPERACIÓN	ACCIÓN
/api/players	GET	findAll
/api/players/{idGame}	GET	findAllByGame
/api/players/{idGame}/{login}	GET	findPlayerInAGame
/api/players/findPlayer/{idPlayer}	GET	findOne
/api/players/{idPartido}/team/{equipo}	PUT	updateTeam
/api/players/{idPartido}/{valorationGame}	PUT	updateValorationGame
/api/players	POST	save
/api/players/{idPlayer}	DELETE	delete

Cuadro 5.9: Acceso a PlayerResource

URL	OPERACIÓN	ACCIÓN
/api/games	GET	findAll
/api/games/{idGame}	GET	findOne
/api/games/{filtro}	GET	findAllFilters
/api/games/locations/{idLocation}	GET	findAllLocation
/api/games/sportsLocation/{idSport}/{idLocation}	GET	findAllSportLocation
/api/games/{idGame}	PUT	update
/api/games/	POST	save
/api/games/{idGame}	DELETE	delete

Cuadro 5.10: Acceso a GameResource

5.3 Front-end

Para la parte Front-end se ha decidido utilizar Vue.js. Vue se basa en una implementación muy ligera del patrón VMMV como se explicaba en la Sección 4.4 que ayudará a relacionar nuestra capa de presentación con nuestra capa de negocio de forma sencilla y eficiente. Es un framework progresivo, es decir, que es muy fácil añadir a cualquier proyecto ya existente; en cambio frameworks como Angular están más orientados a comenzar proyectos desde cero.

5.3.1 Componentes

Para la parte del cliente se divide en componentes que son elementos HTML reusables y configurables. Básicamente es una instancia Vue que será utilizada dentro de otra, y que será referenciada mediante una etiqueta específica. Los componentes son componibles, de tal modo que se pueden añadir componentes dentro de componentes, y tienen un orden jerárquico: los padres pueden modificar los datos de los hijos, pero no al revés. Los hijos únicamente pueden notificar eventos a los padres, pero no modificar su estado directamente.

Un componente en Vue está formado por tres partes diferenciadas como se puede observar en la Figura 5.6:

- **Template:** Se localiza dentro del HTML que define la vista, Vue lo interpretará y renderizará de acuerdo a un modelo de datos. De este modo, podemos utilizar Vue en el front-end de manera similar a cualquier motor de templates de back-end. En el caso de Vue, todo lo que se encuentra entre corchetes en determinados tags html se interpreta como javascript y su valor es modificado en función del valor de nuestros objetos.
- **Script:** Es la parte JavaScript que define el comportamiento del componente. También nos permite añadir aplicaciones Web ya existentes y sin mayores problemas de instalación, simplemente cargándolo en esta parte del componente.
- **Style:** En este apartado es donde añadiremos los estilos que se aplicarán a ese componente

```

</div>
</div>
</template>

<script>
import auth from './common/auth'
import MenuBar from './components/MenuBar'
import { HTTP } from './common/http-common'

export default {
  name: 'App',
  components: { MenuBar },
  data() {
    return {
      // enlazamos el objeto donde vamos a guardar los datos de auth
      // de esta forma se activa el data-binding y los podemos usar
      // propiedades públicas en los componentes
      storeAuth: auth,
    }
  },
}
</script>
<style scoped lang="scss">

```

Figura 5.6: Estructura Componente vue

Se hará una clara distinción entre los componentes citados en la Sección 4.4.1 según los permisos. Por un lado están los componentes de Administración, los componentes de los Usuarios Básicos y finalmente los componentes Comunes, que serán accesibles tanto para Usuarios Anónimos, como para el Administrador como para los Usuarios Básicos. En este diagrama no se mostrarán los componentes “LoadingPage.vue” ni “NotFound.vue” ya que son componentes sin ninguna funcionalidad importante. Cabe destacar que estos componentes, pueden incluir otros componentes. Por ejemplo, los componentes “Game.vue” y “GameCreate.vue” hacen uso del componente “Calendar.vue”.

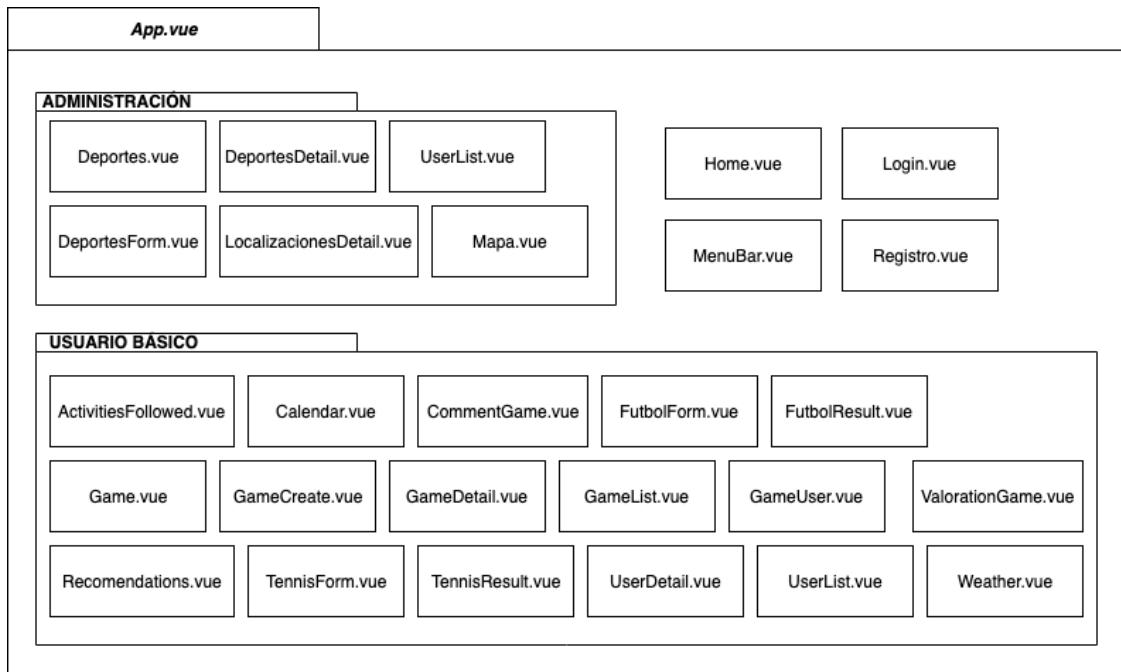


Figura 5.7: Diagrama de componentes del Cliente

5.3.2 Rutas

Para el enrutamiento, es decir, para asociar a cada componente una ruta, Vue.js utiliza VueRoute. Se van a definir las rutas en el AppRoute.vue. Luego en nuestro componente principal (App.vue) en el router-view se va a inyectar un componente u otro dependiendo de las rutas a las que vayamos acceder. Las rutas de los componentes del cliente son las siguientes.

Nombre	Ruta	Componente
Home	/	Home.vue
Registro	/login/createAccount	Registro.vue
Login	/login	Login.vue
UserDetail	/users/:id	UserDetail.vue
UserUpdate	/users/:id/edit	Registro.vue
UserList	/users	UserList.vue
LocalizacionesDetail	/localizaciones/details	LocalizacionesDetail.vue
Mapa	/localizaciones/mapaue	Mapa.vue

Cuadro 5.11: Rutas de los componentes de Administración y Comunes

Nombre	Ruta	Componente
Deportes	/deportes	/ Deportes.vue
DeportesDetail	/deportes/:id	DeportesDetail.vue
DeportesCreate	/deportes/new	DeportesForm.vue
DeportesUpdate	/deportes/:id/edit	DeportesForm.vue
GameCreate	/games/new	GameCreate.vue
GameDetail	/games/:id	GameDetail.vue
CommentGame	/games/:id/comment	CommentGame.vue
ValorationGame	/games/:id/valoraciones	ValorationGame.vue
FutbolForm	/games/:id/futbol/completeResultado	FutbolForm.vue
TennisForm	/games/:id/tennis/completeResultado	Tennisform.vue
FutbolResult	/games/:id/futbol/resultado	FutbolResult.vue
TennisResult	/games/:id/tennis/resultado	TennisResult.vue
GameUser	/users/:id/perfilPublico	GameUser.vue
GameList	/games/organizados/user/:id	GameList.vue
Recomendations	/games/recomendados	Recomendations.vue
Game	/games	Game.vue
Weather	/games/weather	Weather.vue
Calendar	/games/calendar	Calendar.vue
ActivitiesFollowed	/activities/followed	ActivitiesFollowed.vue

Cuadro 5.12: Rutas de los componentes de los Usuarios Básicos

5.3.3 Comunicación con el servidor

En esta sección se verán las comunicaciones de los componentes del cliente con los controladores del servidor a través de peticiones HTTP.

En la Figura 5.11 se muestra la comunicación de los componentes de Administración del Cliente con el Servidor. Realizan peticiones para recuperar los deportes, localizaciones y usuarios, para crear, editar o recuperar los datos de algún deporte o localización en concreto...

En la Figura 5.12 se muestra la comunicación de los componentes del Cliente comunes a todos los usuarios. Las peticiones que van a hacer son para autentificación, para recuperar los partidos pendientes de llenar resultados, los mensajes, los partidos pendientes de valorar...

Finalmente en las Figuras 5.8, 5.10 y 5.9 se muestra la comunicación de los componentes del Cliente de los Usuarios Básicos, estos componentes hacen peticiones al Servidor para recuperar los partidos, crear nuevos, recuperar sus datos, apuntarse, recuperar sus datos personales...

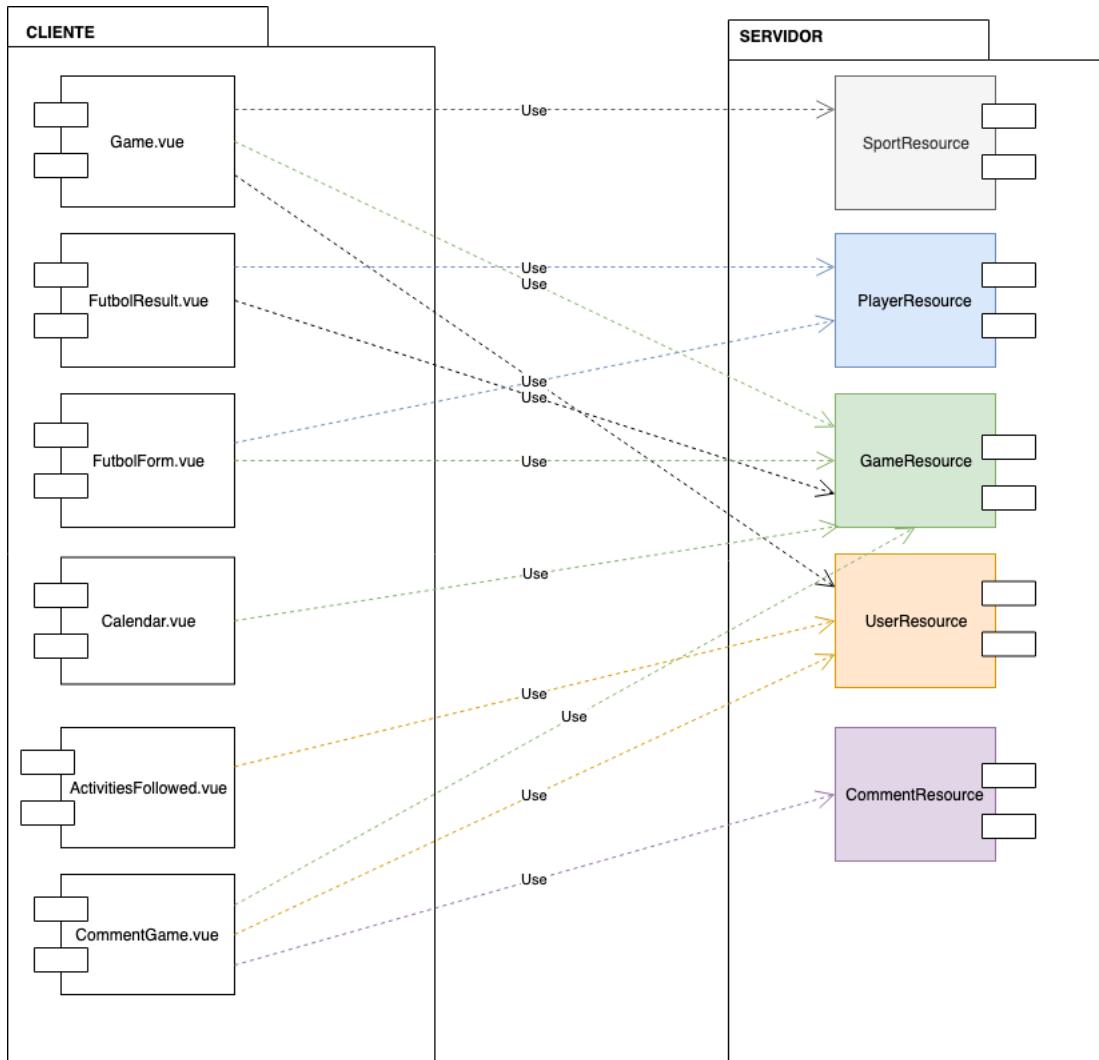


Figura 5.8: Comunicación de los componentes del Usuario del Cliente al Servidor [1]

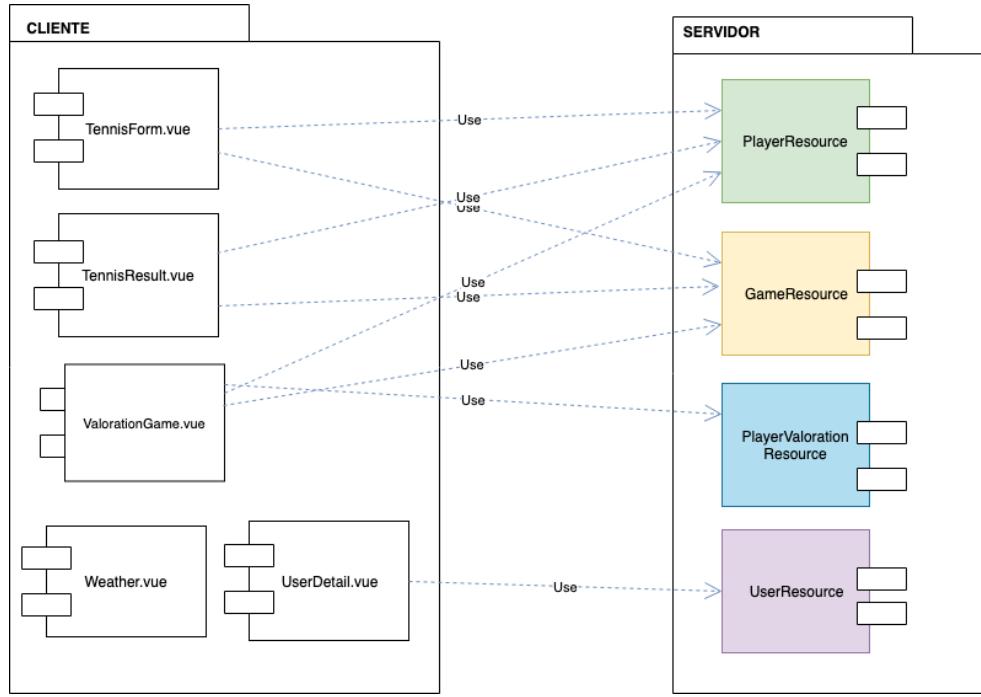


Figura 5.9: Comunicación de los componentes del Usuario del Cliente al Servidor [2]

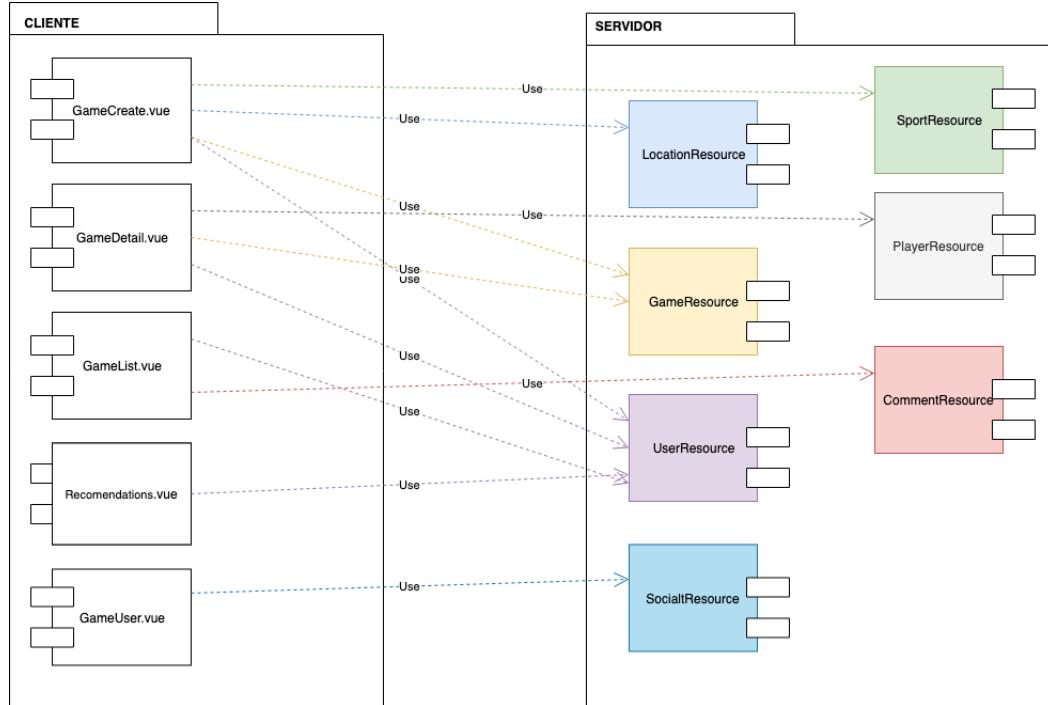


Figura 5.10: Comunicación de los componentes del Usuario del Cliente al Servidor [3]

CAPÍTULO 5. DISEÑO

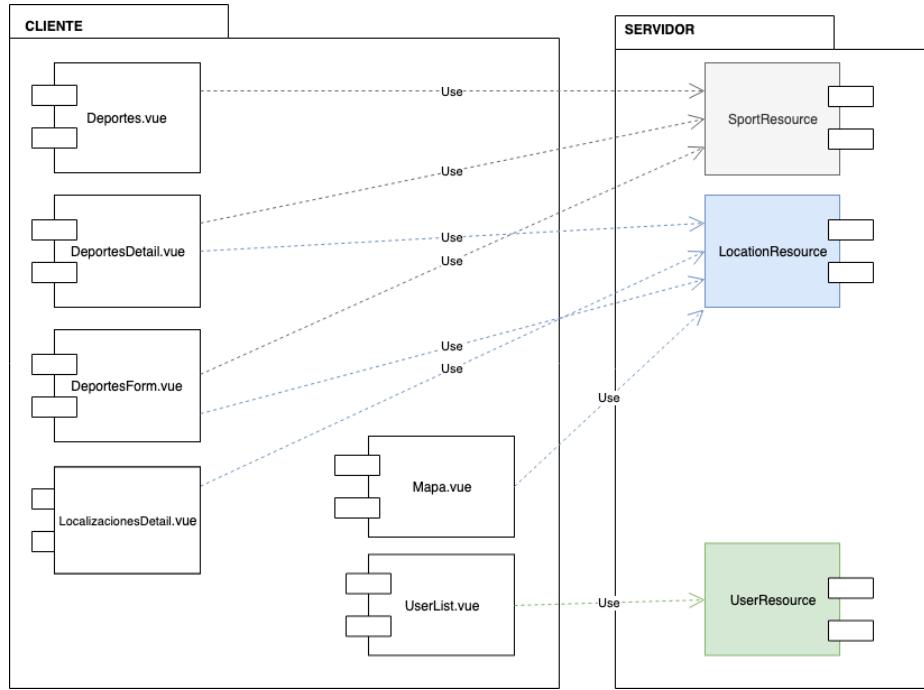


Figura 5.11: Comunicación de los componentes de Administración del Cliente al Servidor

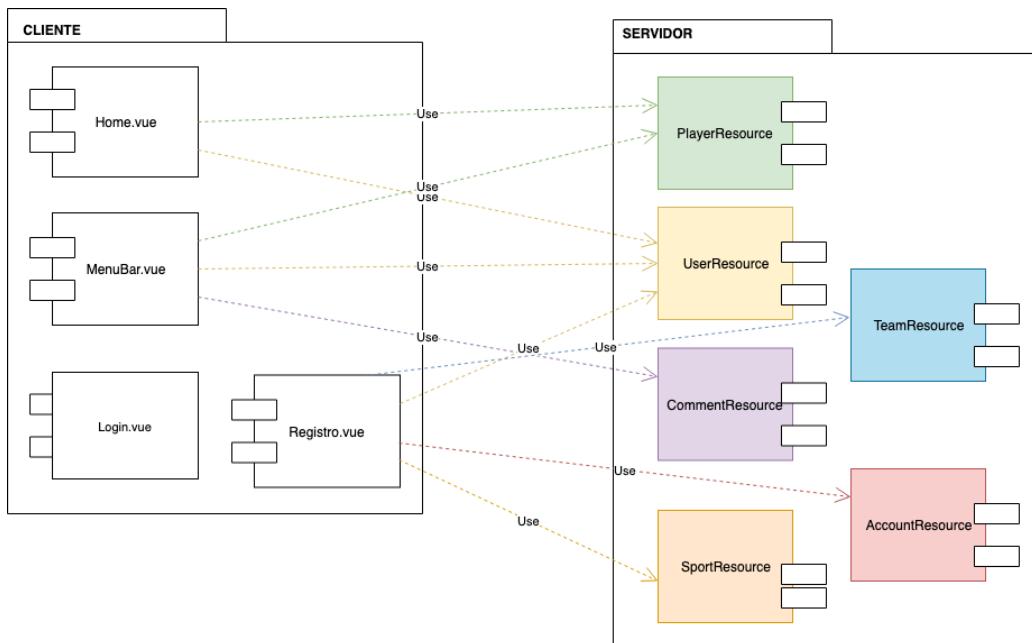


Figura 5.12: Comunicación de los componentes comunes del Cliente al Servidor

Capítulo 6

Implementación y pruebas

En este capítulo se describirán aquellos algoritmos complejos de los que disponemos a lo largo de la en la aplicación. Aparte se mostrarán las distintas pruebas que se han llevado a cabo.

6.1 Implementación

6.1.1 Recomendación de Partidos

Para la recomendación de partidos se desarrolló un algoritmo propio para recomendar a cada jugador partidos en función de los eventos deportivos jugados. Las recomendaciones que se van a realizar van a ser en función de los deportes de los partidos jugados, las ubicaciones y jugadores con los que jugamos en partidos anteriores y tienen buena valoración de media. A continuación se expondrá y se explicará paso a paso el algoritmo de recomendación realizado.

El método `public List<RecomendacionDTO> findGamesRecomendados(String login)` se encuentra en el `UserService`. Primero se accederá a él en la parte del cliente desde el componente “GameList.vue” haciendo una petición GET al controlador `UserResource`. Después de recibir la respuesta se ejecutará en la parte del cliente el método `filterGamesRecomendados` que mostrarán los partidos recomendados en función de lo que el cliente clickeará previamente, es decir, si elige por jugadores, deporte o localización...

```
HTTP.get(`users/${this.WhatLogin()}/recomendados`)
  .then(response => {this.gamesRecomendados = response.data return response.data})
  .then(this.filterGamesRecomendados)
```

Figura 6.1: Petición Cliente: Recomendaciones

Se procederá a explicar paso por paso el algoritmo del lado del Servidor:

Recomendación de partidos según jugadores

Consistirá en recomendarle al usuario partidos en los que estén apuntados usuarios con los que él previamente haya jugado y tenga una valoración superior a tres, para así únicamente recomendarle jugadores buenos con los que ya haya jugado alguna vez.

Antes de nada se recuperará los partidos jugados por ese usuario, ya que si no ha jugado ningún partido no se le podrá recomendar nada, ya que estas recomendaciones se basan en el histórico. Después, de todos los partidos jugados se recuperarán los jugadores con una experiencia superior a tres puntos. Se decide quedarse con este conjunto de jugadores para así únicamente recomendarle al usuario aquellos que sean buenos según las valoraciones del resto de usuarios, ya que esta experiencia es la media de las valoraciones anónimas recibidas. Finalmente se buscarán los partidos que estén aún vigentes donde estén apuntados esos jugadores que fuimos recolectando. Finalmente se utilizará un HashMap para evitar devolver recomendaciones de partidos repetidos. Finalmente se guardarán los partidos recomendados en un *RecomendacionDTO*, donde se guardará una lista de juegos recomendados y un mensaje que indicará el porqué de esa recomendación, y este objeto se guardará en una lista de objetos, donde en la posición cero se guardarán las recomendaciones de los jugadores.

```

if(jugados.size()!=0){
    //Recomendar partidos en los que jueguen jugadores buenos con los que ya jugó
    for(Game game:jugados){
        jugadoresJugados.addAll(playerDAO.findAllByGame(game.getIdGame()));
    }
    for(Player j :jugadoresJugados ){
        if((j.getPlayer().getExperience()>new Double(3))
            &&(j.getPlayer().getLogin()!=getCurrentUserWithoutAuthority().getLogin())){
            jugadoresValoradosBien.add(j);
        }
    }
    for(Player j:jugadoresValoradosBien){
        recomendados.addAll(userDAO.findRecomendadosPlayers(j.getPlayer().getLogin()))
            .stream().map(game -> new GameDTO(game)).collect(Collectors.toList());
    }
    Map<Long,GameDTO> mapGames=new HashMap<Long, GameDTO>(recomendados.size());
    for(GameDTO g : recomendados) {
        mapGames.put(g.getIdGame(), g);
    }

    for(Entry<Long, GameDTO> g : mapGames.entrySet()) {
        recomendadosLimpia.add(g.getValue());
    }

    RecomendacionDTO recomendacion=new RecomendacionDTO();
    recomendacion.setMensaje("Por que jugaste con ellos");
    recomendacion.setGames(recomendadosLimpia);
    recomendadosFinal.add(0, recomendacion);
}

```

Figura 6.2: Algoritmo de Recomendación: Jugadores

Recomendación de partidos según deportes

Consistirá en recomendarle al usuario partidos de los deportes que él más haya jugado, es decir, si el usuario ha jugado 8 partidos de tenis y 2 de baloncesto, el algoritmo de recomendación por deportes le recomendará al usuario partidos de tenis, ya que son los que más suele frecuentar.

Después de haber recuperado los partidos recomendados según jugadores con los que el usuario ha jugado y poseen una buena experiencia, ahora se pasará a recomendar partidos de los deportes que el usuario suele jugar. Primero se recorrerán todos los partidos jugados, se irán guardando en un array de deportes en la posición de cada idSport de cada deporte el número que aparece ese deporte en la lista de partidos jugados. El deporte que aparezca más veces entre los más jugados se utilizará para buscar los partidos recomendados por deportes, buscando los partidos vigentes que sean de ese tipo de Deportes. Además decidimos que en caso de empate, se recomendarán ambos partidos, así que buscamos si algún otro deportes consiguió el mismo número de eventos jugados y buscamos sus partidos recomendados y los unimos a los anteriores. De esta forma tenemos ya el array de partidos recomendados según los deportes, que se filtrarán a través del HashMap y se almacenará en el objeto *RecomendacionDTO* que finalmente se meterá en la posición uno de la lista de objetos.

```
//Recomendar partido por el deporte que más juega
for(Game game:jugados){
    sports[game.getSport().getIdSport().intValue()]= (sports[game.getSport().getIdSport().intValue()]+1);
}

Integer iNumeroMayor = sports[1];
Integer iPosicion = 0;

for (int x=1;x<sports.length;x++){
    if (sports[x]>=iNumeroMayor){
        iNumeroMayor = sports[x];
        iPosicion = x;
    }
}

List<GameDTO> gamesDeportes= gameDAO.findAllSport(iPosicion.longValue())
    .stream().map(game -> new GameDTO(game)).collect(Collectors.toList());
for (int x=1;x<sports.length;x++){
    if (sports[x]==iNumeroMayor){
        iPosicion = x;
        List<GameDTO> gamesDeportesSame= gameDAO.findAllSport(iPosicion.longValue())
            .stream().map(game -> new GameDTO(game)).collect(Collectors.toList());
        gamesDeportes.addAll(gamesDeportesSame);
    }
}
Map<Long,GameDTO> mapGamesSport=new HashMap<Long, GameDTO>(gamesDeportes.size());
for(GameDTO g : gamesDeportes) {
    mapGamesSport.put(g.getIdGame(), g);
}
for(Entry<Long, GameDTO> g : mapGamesSport.entrySet()) {
    recomendadosLimpia1.addValue(g.getValue());
}

RecomendacionDTO recomendacion1=new RecomendacionDTO();
recomendacion1.setMensaje("Por que te gusta este Deporte");
recomendacion1.setGames(recomendadosLimpia1);
recomendadosFinal.add(1, recomendacion1);
```

Figura 6.3: Algoritmo de Recomendación: Deportes

Recomendación de partidos según localización

Consistirá en recomendarle al usuario partidos de las localizaciones donde él más haya jugado, es decir, si el usuario ha jugado 8 partidos en Bastiagueiro y 2 en el Parque de Oza, el algoritmo de recomendación por localizaciones le recomendará al usuario partidos en la zona de Bastiagueiro, ya que son los que más suele frequentar.

Finalmente haremos lo mismo con las localizaciones. El procedimiento de los partidos recomendados según las localizaciones es el mismo que el de los deportes, solo que estos partidos recomendados se guardarán en la posición dos de la lista de objetos.

```
//Recomendar partido por la ubicacion donde más juega
for(Game game:jugados){
    locations[game.getLocation().getIdLocation().intValue()]=
        (locations[game.getLocation().getIdLocation().intValue()]+1);
}

Integer iNumeroMayorL = locations[1];
Integer iPosicionL = 0;

for (int x=1;x<locations.length;x++){
    if (locations[x]>=iNumeroMayorL){
        iNumeroMayorL = locations[x];
        iPosicionL = x;
    }
}

List<GameDTO> gamesLocations= gameDAO.findAllLocation(iPosicionL.longValue())
    .stream().map(game -> new GameDTO(game)).collect(Collectors.toList());
for (int x=1;<locations.length;x++){
    if (locations[x]==iNumeroMayorL){
        iPosicion = x;
        List<GameDTO> gamesLocationsSame= gameDAO.findAllLocation(iPosicion.longValue())
            .stream().map(game -> new GameDTO(game)).collect(Collectors.toList());
        gamesLocations.addAll(gamesLocationsSame);
    }
}
Map<Long,GameDTO> mapGamesLocations=new HashMap<Long, GameDTO>(gamesLocations.size());
for(GameDTO g : gamesLocations) {
    mapGamesLocations.put(g.getIdGame(), g);
}

for(Entry<Long, GameDTO> g : mapGamesLocations.entrySet()) {
    recomendadosLimpia2.addValue(g.getValue());
}

RecomendacionDTO recomendacion2=new RecomendacionDTO();
recomendacion2.setMensaje("Por que te gusta esta ubicación");
recomendacion2.setGames(recomendadosLimpia2);
recomendadosFinal.add(2, recomendacion2);
```

Figura 6.4: Algoritmo de Recomendación: Localizaciones

```
public class RecomendacionDTO {
```

- ↳ @NotEmpty


```
private String mensaje;
```
- ↳ @NotNull


```
private List<GameDTO> games;
```

Figura 6.5: RecomendacionDTO

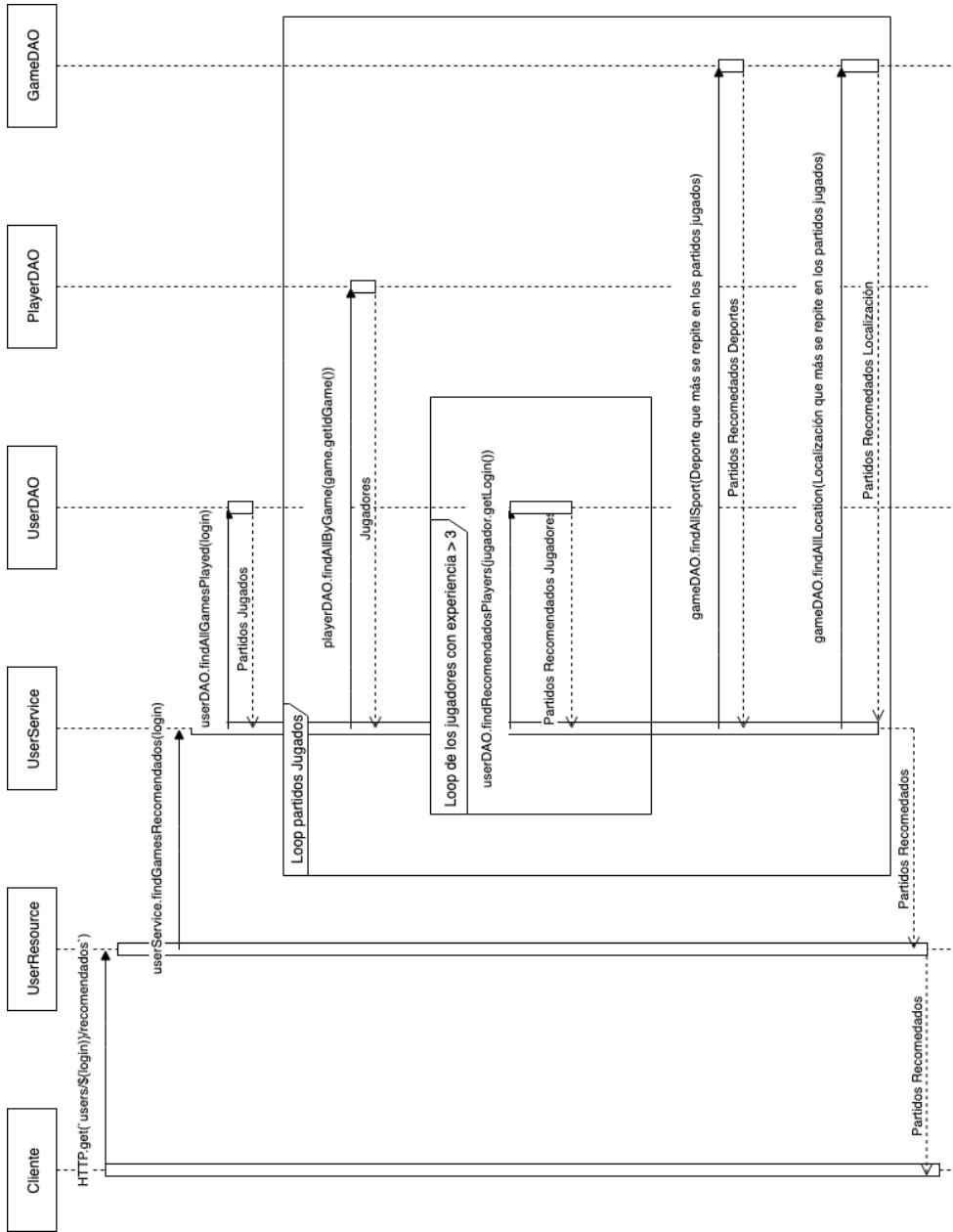


Figura 6.6: Diagrama de Secuencia del Algoritmo de Recomendación

6.1.2 Motor de Renderizado de pantallas

Como se vio previamente, uno de los objetivos principales de la aplicación era adaptar la aplicación para que la adición de nuevos deportes fuera de la forma más sencilla y eficiente posible. En la actualidad la plataforma está preparada para gestionar partidos de fútbol, tenis, pádel y baloncesto, pero en un futuro pueden aparecer más, por lo que tenemos que hacer el código lo más mantenable posible.

Para alcanzar este objetivo cada deporte debe almacenar cual sería el nombre de la plantilla donde se rellenarán los resultados finales del partido y la plantilla para visualizarlos. En el caso de que se cree un deporte nuevo, el Administrador lo único que tendrá que realizar será ir al apartado de “Deportes”, crear el deporte nuevo, y en los valores de “Componente de Entrada” y “Componente de Visualización” introducir el nombre de las plantillas que se utilizarán para mostrar el formulario que servirá para llenar los resultados finales del partido y el componente que mostrará estos resultados finalmente.

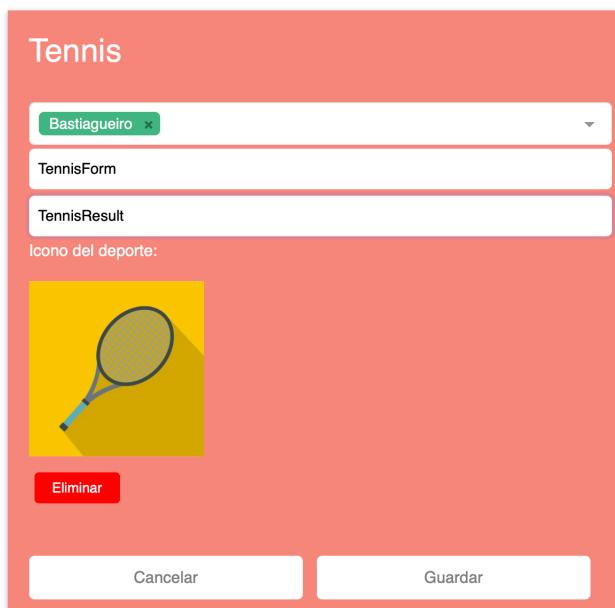


Figura 6.7: Componentes de visualización de resultado según deporte

```
this.$router.replace({ name: this.gameSelect.sport.componenteEntrada,
  params: { id:this.gameSelect.idGame}})
```

Figura 6.8: Cambio de plantilla según valor del componente de entrada

Pero para conseguir esto, el resultado final del partido debe ser enviado al Servidor en un formato que nos permita guardar tanto los sets y juegos de un partido de tenis o pádel como los goles o puntos de un partido de fútbol y baloncesto, por lo que el resultado del partido se decidió enviar desde el Cliente en formato JSON, luego cada plantilla dependiendo de a que deporte vaya dirigida hará un tratamiento diferente de este resultado.

```
{
    "equipoA": {
        "setsGanados": 2,
        "sets": ["2", "6", "6"]
    },
    "equipoB": {
        "setsGanados": 1,
        "sets": ["6", "1", "2"]
    }
}
```

Figura 6.9: Resultado partido de Tenis (JSON)

```
{
    "equipoA": {
        "goles": "2",
        "jugadoresA": [{"id": "sandra", "goles": "2"}]
    },
    "equipoB": {
        "goles": "1",
        "jugadoresB": [{"id": "laura", "goles": "1"}]
    }
}
```

Figura 6.10: Resultado partido de Fútbol (JSON)

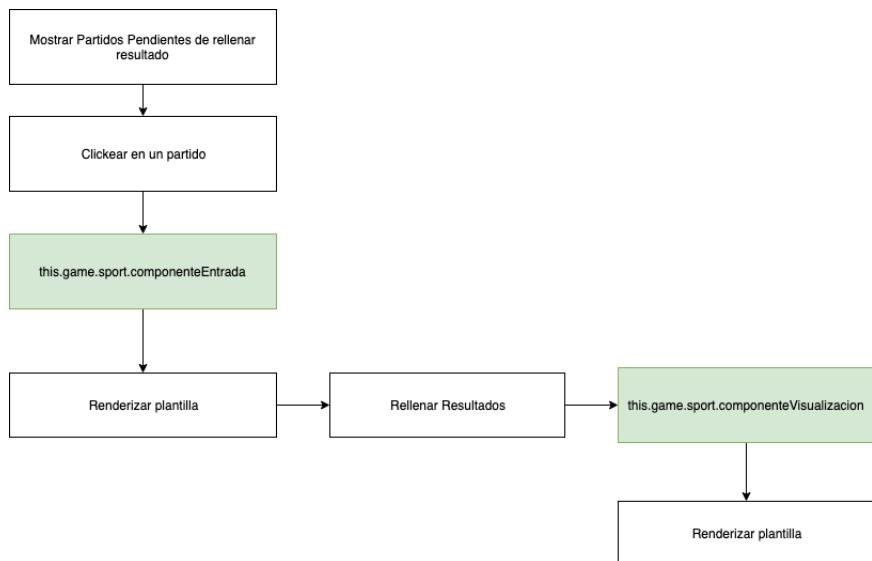


Figura 6.11: Funcionamiento Motor Plantillas

6.2 Pruebas

6.2.1 Pruebas Unitarias

Las pruebas unitarias [30] consisten en comprobar el correcto funcionamiento de un elemento específico del software. Para estas pruebas utilizaremos el soporte de framework utilizando Spring Boot. Para ello añadiremos en el *pom.xml* la dependencia *spring-boot-starter-test* que contiene la mayoría de los elementos necesarios para nuestras pruebas.

Para poder llevar a cabo estas pruebas serán necesarias las siguientes anotaciones:

- *@RunWith(SpringRunner.class)*: Se utiliza para proporcionar un puente entre las funciones de prueba Spring Boot y JUnit.
- *@SpringBootTest*: Le especifica a Spring Boot que vaya y busque la clase de configuración principal, la que tenga la anotación *@SpringBootApplication*, y la use para iniciar un contexto de aplicación Spring.
- *@WithMockUser*: Para ejecutar las pruebas con un usuario específico utilizaremos esta anotación, indicándole el nombre y los roles.

Hemos realizado pruebas destinadas a comprobar el correcto funcionamiento de las funcionalidades expuestas en todos los servicios, desde operaciones tan básicas como crear, eliminar y actualizar como búsquedas más avanzadas, como se muestra en la Figura 6.13, y los casos de prueba en los que la salida no es un determinado valor, sino que se produzca una excepción como se puede observar en la Figura 6.2.1.

CAPÍTULO 6. IMPLEMENTACIÓN Y PRUEBAS

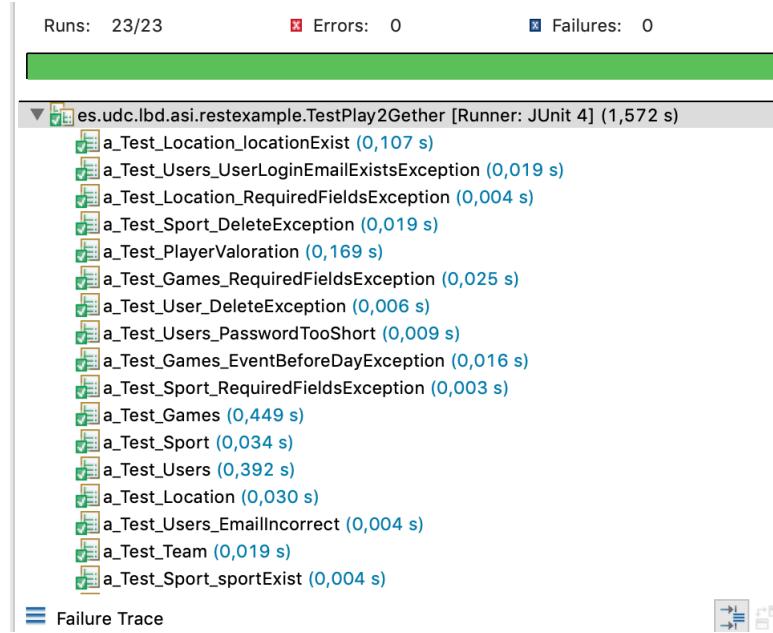


Figura 6.12: Pruebas Realizadas sobre los Servicios

```
//Buscar partidos jugados por ese usuario
assertEquals (1,userService.findGamesPlayed("sandra").size());
assertEquals (0,userService.findGamesPlayed("rober").size());
assertEquals (1,userService.findGamesPlayed("laura").size());
assertEquals (1,userService.findGamesPlayed("lucas").size());

//Buscar partidos creados por ese usuario
assertEquals (3,userService.findGamesCreated("sandra").size());
assertEquals (0,userService.findGamesCreated("rober").size());
assertEquals (1,userService.findGamesCreated("laura").size());
assertEquals (1,userService.findGamesCreated("lucas").size());

//Buscar partidos próximos para ese usuario
assertEquals (2,userService.findGamesNext("sandra").size());
assertEquals (0,userService.findGamesNext("rober").size());
assertEquals (1,userService.findGamesNext("laura").size());
assertEquals (0,userService.findGamesNext("lucas").size());

//Buscar partidos recomendados para ese usuario
assertEquals (1,userService.findGamesRecomendados("sandra").get(0).getGames().size());
assertEquals (1,userService.findGamesRecomendados("sandra").get(1).getGames().size());
assertEquals (2,userService.findGamesRecomendados("sandra").get(2).getGames().size());

//Buscar partidos pendientes de llenar resultado por ese usuario
assertEquals (1,userService.findByCreatorResultado("sandra").size());
assertEquals (0,userService.findByCreatorResultado("laura").size());
assertEquals (0,userService.findByCreatorResultado("rober").size());

//Buscar partidos pendientes de valorar por ese usuario
assertEquals (1,userService.findByValoration("sandra").size());
assertEquals (0,userService.findByValoration("rober").size());
assertEquals (1,userService.findByValoration("laura").size());
```

Figura 6.13: Pruebas Realizadas sobre el Servicio de Usuario

```

    @Test(expected= SportExistsException.class)
    public void a_Test_Sport_sportExist() throws SportExistsException, RequiredFieldsException {
        //Create un deporte con el mismo tipo que otro partido de la BD
        List<SportDTO>listaxeSport= new ArrayList();
        Set <Location>locationsGolf = new <Location>.HashSet();
        Sport sport1=new Sport("Tennis","GolfForm","GolfResult",locationsGolf,"Golf.png");
        SportDTO sport1d= new SportDTO(sport1);
        sportService.save(sport1d);

    }

    @Test(expected= RequiredFieldsException.class)
    public void a_Test_Sport_RequiredFieldsException() throws SportExistsException, RequiredFieldsException {
        //Crear un deporte con el campo tipo a nulo
        List<SportDTO>listaxeSport= new ArrayList();
        Set <Location>locationsGolf = new <Location>.HashSet();
        Sport sport1=new Sport(null,"GolfForm","GolfResult",locationsGolf,"Golf.png");
        SportDTO sport1d= new SportDTO(sport1);
        sportService.save(sport1d);

    }

    @Test(expected= DeleteException.class)
    public void a_Test_Sport_DeleteException() throws DeleteException {
        //Borrar deporte con partidos asociados
        List<SportDTO>listaxeSport= new ArrayList();
        sportService.deleteById(new Long(1));
        listaxeSport= sportService.findAll();
        assertEquals (4,listaxeSport.size());
    }
}

```

Figura 6.14: Pruebas de Excepciones

6.2.2 Pruebas REST

Para comprobar que todas las funcionalidades Rest se ha utilizado PostMan, una herramienta que nos permite realizar peticiones HTTP a cualquier API para comprobar el correcto funcionamiento de nuestros desarrollos. Primero se llevará a cabo una petición *POST* para autenticarnos y recibir de vuelta el *token* necesario para poder realizar luego cualquier petición.

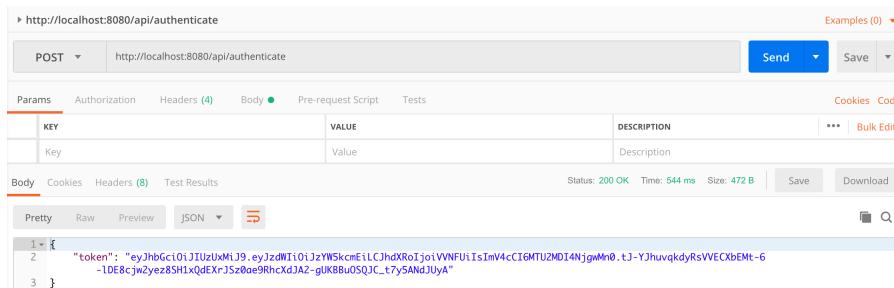


Figura 6.15: Petición POST de autenticación

CAPÍTULO 6. IMPLEMENTACIÓN Y PRUEBAS

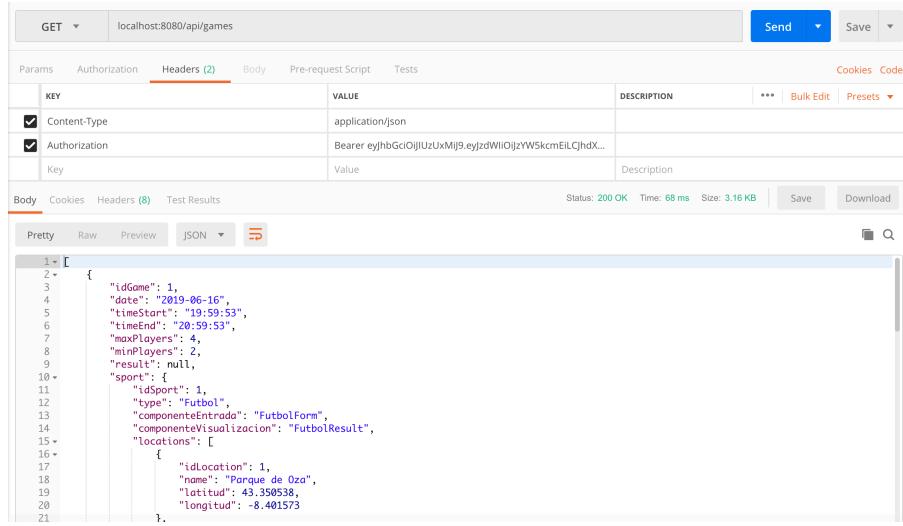


Figura 6.16: Petición GET

6.2.3 Pruebas de integración y aceptación

Estas pruebas se han llevado a cabo mediante la ejecución de los diferentes casos de uso simulando ser cada uno de los usuarios ofrecidos por el sistema. Se revisará que todas las operaciones del sistema funcionan correctamente, comprobando los permisos de cada usuario, creando usuarios, partidos, poniendo en prueba las partes sociales de la aplicación, la carga de imágenes, revisando los correos de las notificaciones, probando la aplicación en distintos navegadores, en resumen, comprobar que toda la aplicación en conjunto funciona correctamente.

Capítulo 7

Solución desarrollada

En este capítulo se mostrará una pequeña guía sobre el funcionamiento básico de la aplicación, centrándonos en las operaciones principales. Comenzaremos con lo que son las páginas de acceso a la aplicación, luego con algunas funcionalidades del administrador y finalmente las operaciones esenciales de los usuarios.

7.1 Acceso a la Aplicación

Hay dos formas de acceder a la aplicación, o logueándose en ella o registrándose llenando los campos obligatorios. Esto es debido a que no podremos acceder a ninguna funcionalidad a no ser que estemos autenticados.

The screenshot shows the login page for the PLAY2GETHER application. At the top, there is a teal header bar with the 'PLAY2GETHER' logo on the left and a 'Login' button on the right. Below the header, the main content area has a light gray background. In the center, the text 'Sign in!' is displayed in a blue font. Below this, there are two input fields: one for 'Username' and one for 'Password'. Each field has a placeholder text: 'Escribe tu login' for the username and 'Escribe tu contraseña' for the password. At the bottom of the form, there are two buttons: a teal 'Log In' button on the left and a red 'Registro' (Registration) button on the right. The footer of the page is a dark gray bar containing the 'PLAY2GETHER' logo and the text 'PLAY2GETHER © 2019'.

Figura 7.1: Página de Login

The screenshot shows the 'Datos Personales' (Personal Data) section of the PLAY2GETHER registration form. The form is divided into two columns. The left column contains fields for Name (Nombre), Surname (Primer Apellido and Segundo Apellido), Date of Birth (Fecha de Nacimiento), and City (Ciudad). The right column contains fields for Email (Email), Login (Login), and Password (Contraseña). A 'Guardar' (Save) button is located at the top right of the form area. At the bottom, there is a 'Foto de perfil' (Profile Photo) input field and a footer with the PLAY2GETHER logo and copyright information.

Figura 7.2: Página de Registro

7.2 Administración

Las funcionalidades básicas que trataremos en esta sección serán la gestión de deportes, localizaciones y usuarios.

Tendremos la posibilidad tanto de visualizar, editar y dar de alta deportes en la Sección *Deportes* de la barra de navegación. También se dispondrá de las mismas funcionalidades para las localizaciones, pero con la ayuda de un mapa para poder seleccionar las coordenadas únicamente clickeando en él. Finalmente en la gestión de usuarios se permitirá al administrador dar de baja usuarios.

CAPÍTULO 7. SOLUCIÓN DESARROLLADA

The screenshot shows the PLAY2GETHER application interface. At the top, there is a navigation bar with the logo 'PLAY2GETHER' and links for 'Deportes', 'Localizaciones', and 'Usuarios'. On the right side of the bar, it says 'Conectado' with a dropdown arrow. Below the navigation bar, there are two main sections. On the left, a light gray sidebar titled 'Deportes' contains a red button labeled 'NUEVO DEPORTE' and a list of sports: 'BALONCESTO', 'FUTBOL', 'PADDLE', and 'TENNIS'. On the right, a red-themed form titled 'Nuevo Deporte' has fields for 'Nombre del deporte' (with 'Parque de Oza' entered), 'Localizaciones' (with 'FutbolForm' and 'FutboResult' selected), 'Nombre del componente de entrada', 'Nombre del componente de visualización', and 'Icono del deporte' (with a placeholder 'Seleccionar archivo' and 'Ningún archivo seleccionado'). At the bottom of the form are 'Cancelar' and 'Guardar' buttons.

Figura 7.3: Página para dar de alta un deporte

The screenshot shows the PLAY2GETHER application interface. At the top, there is a navigation bar with the logo 'PLAY2GETHER' and links for 'Deportes', 'Localizaciones', and 'Usuarios'. On the right side of the bar, it says 'Conectado' with a dropdown arrow. Below the navigation bar, there are two main sections. On the left, a light gray sidebar titled 'Deportes' contains a red button labeled 'NUEVO DEPORTE' and a list of sports: 'BALONCESTO', 'FUTBOL', 'PADDLE', and 'TENNIS'. On the right, a red-themed form titled 'Baloncesto' displays the details for basketball: 'Nombre del deporte' (Parque de Oza), 'Localizaciones' (FutbolForm, FutboResult), and an 'Icono del deporte' (a basketball icon). There is also a 'Eliminar' (Delete) button below the icon. At the bottom of the form are 'Cancelar' and 'Guardar' buttons.

Figura 7.4: Página para editar un deporte

7.2. Administración

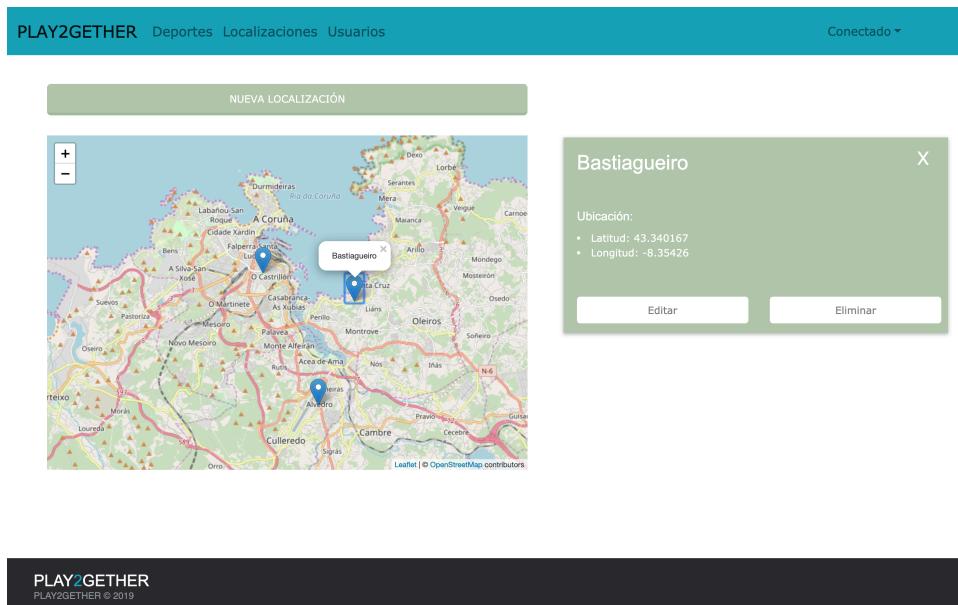


Figura 7.5: Página para visualizar una localización

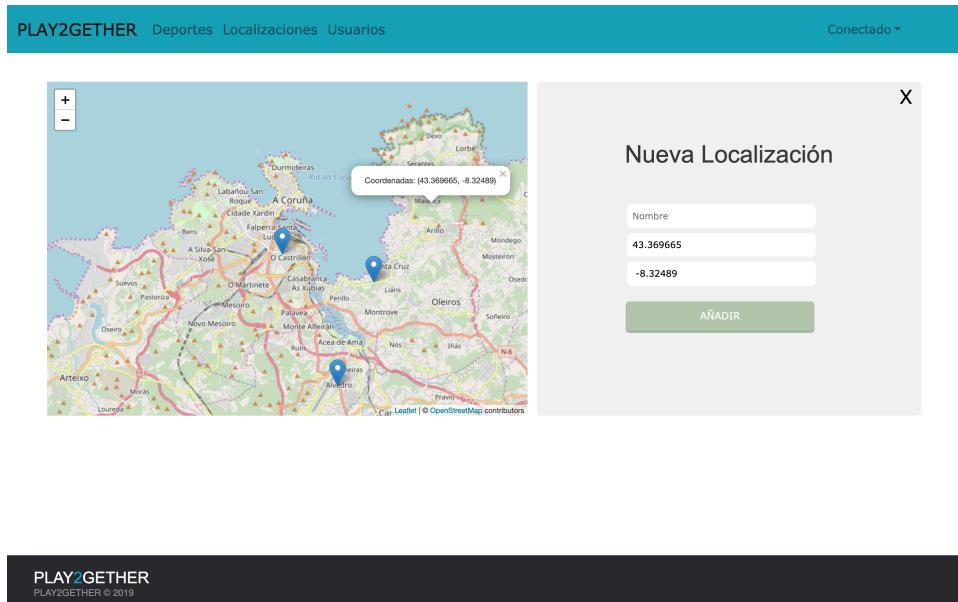


Figura 7.6: Página para dar de alta una localización

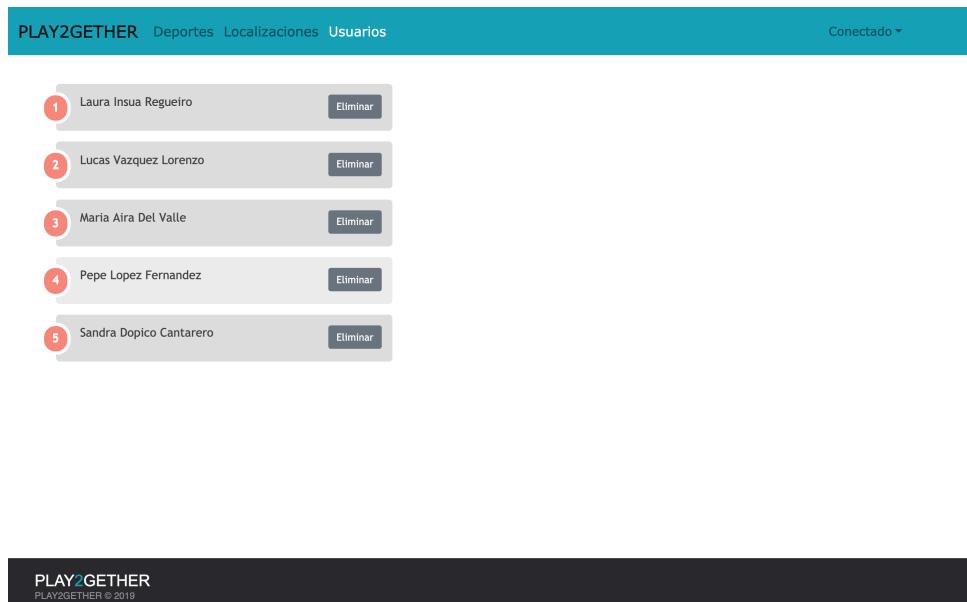


Figura 7.7: Página para dar de baja usuarios

7.3 Usuarios Básicos

En esta sección nos centraremos en las principales funcionalidades, este rol es el que abarca mayores operaciones ya que realmente el administrador se encarga únicamente de la gestión de la aplicación.

Nos permitirá buscar partidos de diferentes formas, tendremos un perfil público, donde podremos ver los diferentes partidos jugados, organizados, a los que me apunté, los partidos que me recomienda la aplicación y los comentarios que me ponen el resto de jugadores después de terminar un partido. Además podremos visualizar el Tablón de Actividades donde podremos observar que hacen las personas a las que seguimos. También disponemos de un apartado de mensajería

7.3. Usuarios Básicos

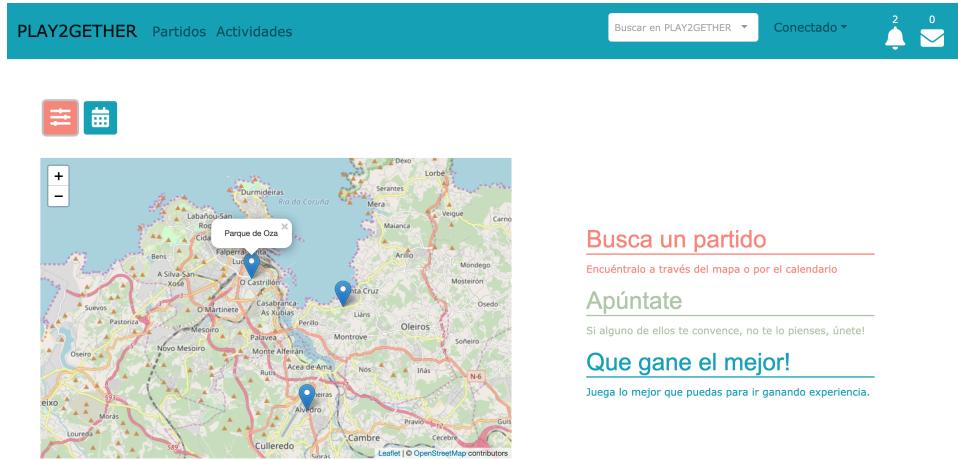


Figura 7.8: Página de búsqueda de partidos: Mapa

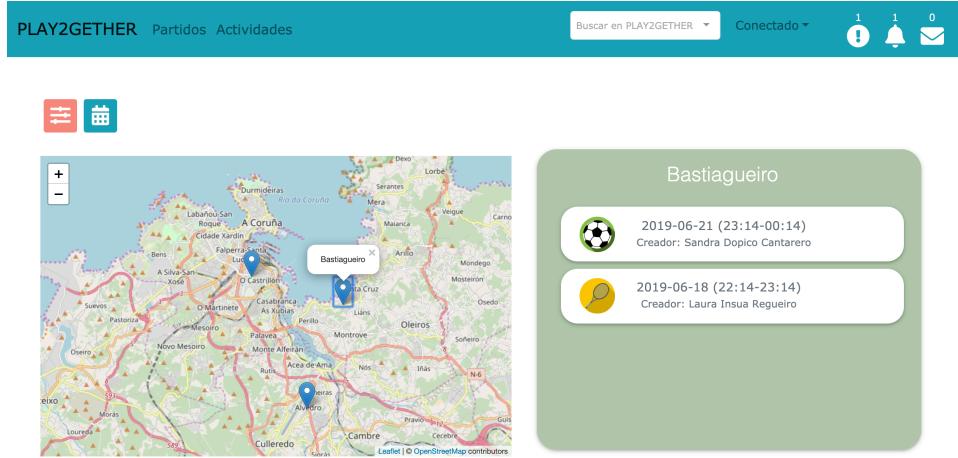


Figura 7.9: Página de búsqueda de partidos: Mapa ubicación seleccionada

CAPÍTULO 7. SOLUCIÓN DESARROLLADA

The screenshot shows the PLAY2GETHER website's calendar interface. At the top, there is a navigation bar with the logo, search bar, and user status (Connected). Below the navigation bar is a header with a location pin icon and a date range: "Jun 2 – 8, 2019". To the right of the date range are buttons for "month", "week", and "day". The main area is a grid calendar for the week of June 2nd. The days of the week are labeled: Sun 6/2, Mon 6/3, Tue 6/4, Wed 6/5, Thu 6/6, Fri 6/7, Sat 6/8. The hours of the day are listed vertically on the left: all-day, 6am, 7am, 8am, 9am, 10am, 11am, 12pm, 1pm. Specific events are highlighted with colored boxes: a yellow box from 8am to 12pm on Tuesday, a red box from 11am to 12pm on Wednesday labeled "Bautizo", and a red box from 10:30am to 11:30am on Friday labeled "Padel Plus". To the right of the calendar, there are three sections: "Busca un partido" (Search for a game), "Apúntate" (Sign up), and "Que gane el mejor!" (The best wins!).

Figura 7.10: Página de búsqueda de partidos: Calendario

The screenshot shows the PLAY2GETHER website's game details page. At the top, there is a navigation bar with the logo, search bar, and user status (Connected). Below the navigation bar is a header with icons for back, cloud, notifications, and messages, and a button for "Detalles Partido" (Game Details). The main content area is divided into two sections: "Información" (Information) on the left and "Participantes" (Participants) on the right. The "Información" section contains the following details: Creador: Lucas Vazquez Lorenzo, Deporte: Padel, Ubicación: Padel Plus, Horario: 21:14-22:14, Fecha: 2019-06-20. The "Participantes" section shows a profile picture of a person named Sandra. At the bottom, there is a footer with the logo and copyright information: "PLAY2GETHER © 2019".

Figura 7.11: Página de detalle del partido

7.3. Usuarios Básicos

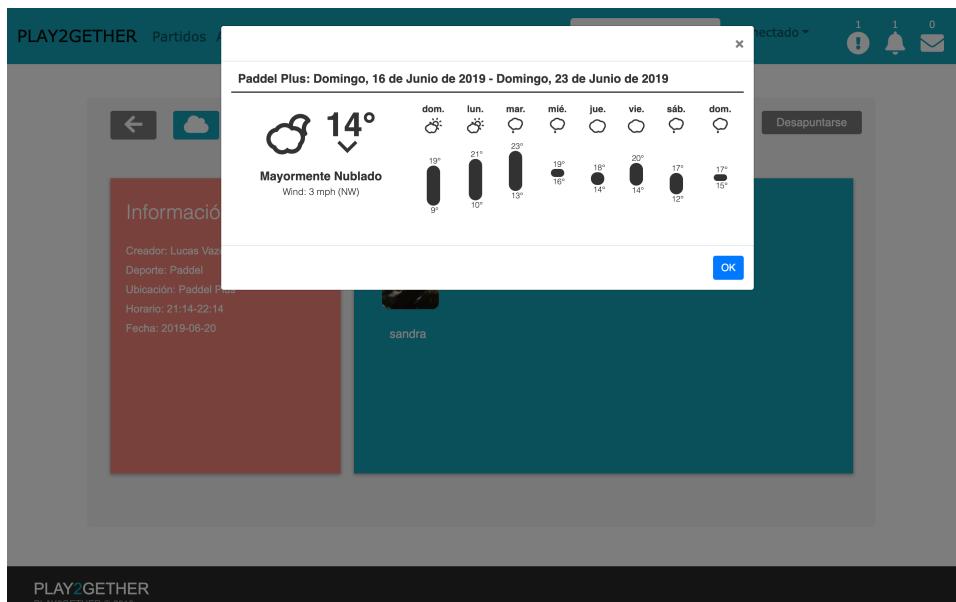


Figura 7.12: Página de detalle del partido: Meteorología

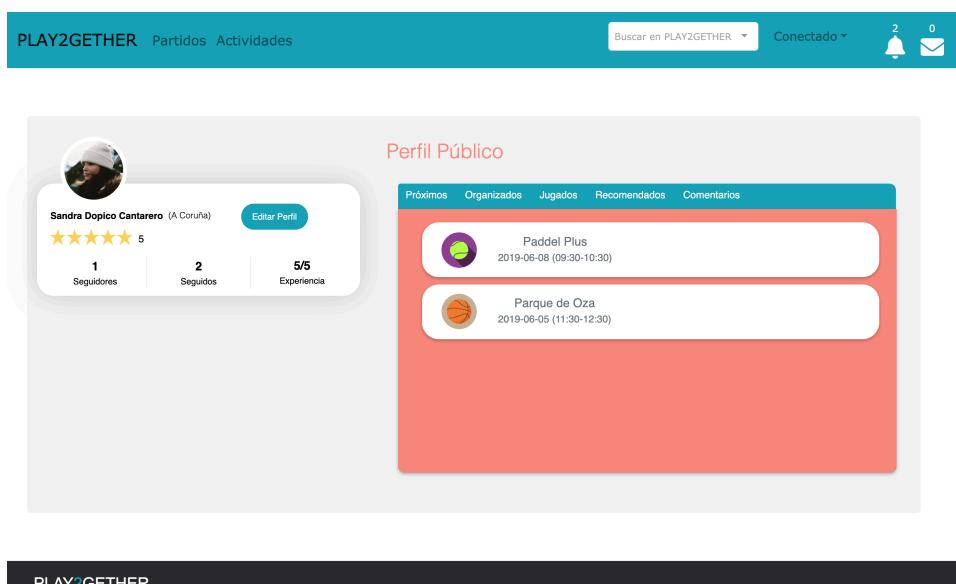


Figura 7.13: Página del Perfil Público

CAPÍTULO 7. SOLUCIÓN DESARROLLADA

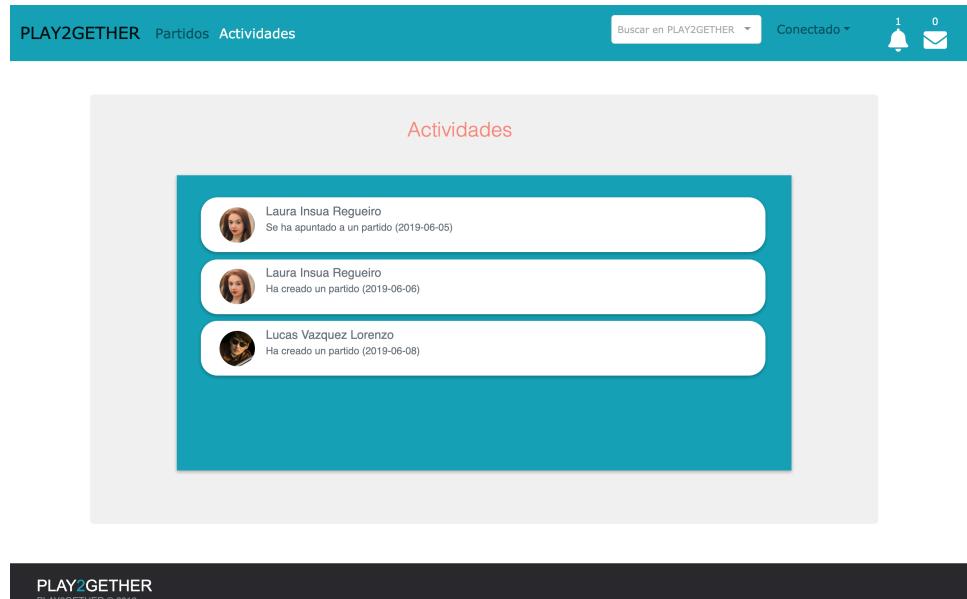


Figura 7.14: Página del Tablón de Actividades

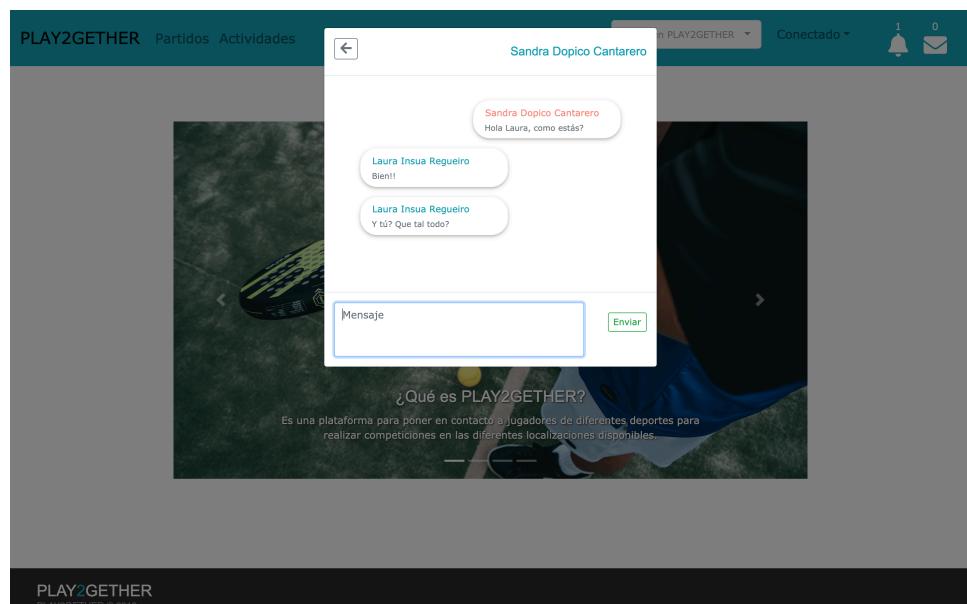


Figura 7.15: Página de Mensajes Directos

7.3. Usuarios Básicos

The screenshot shows the 'Datos Personales' (Personal Data) section of the PLAY2GETHER app. At the top, there is a header bar with the PLAY2GETHER logo, navigation links for 'Partidos' and 'Actividades', and a search bar. On the right side of the header are buttons for 'Buscar en PLAY2GETHER', 'Conectado', a notification bell with 1 notification, and an envelope icon with 0 messages.

The main form is titled 'Datos Personales:' and contains the following fields:

- Nombre:** * (Input: Laura)
- Primer Apellido:** * (Input: Insua)
- Segundo Apellido:** * (Input: Regueiro)
- Fecha de Nacimiento:** (Input: 18/12/1960)
- Ciudad:** (Input: Bilbao)
- Foto de perfil:** (Thumbnail of a woman's face, with a red 'Eliminar' (Delete) button below it).
- Email:** * (Input: guajndos@gmail.com)
- Contraseña:** * (Input field with masked text: *****) and a 'Modificar' (Change) button.
- Equipos favorito:** (Dropdown menu: Deportivo/Futbol)
- Equipos donde juego:** (Dropdown menu: Lakers/Baloncesto)
- Añadir Equipo** (Green button)

Figura 7.16: Página de Datos Personales

Capítulo 8

Conclusiones y trabajo futuro

En este capítulo se hablará del grado en el que se han alcanzado los objetivos planteados al inicio del proyecto y de futuras ampliaciones que se podrían desarrollar en la aplicación.

8.1 Conclusiones

Mediante la realización del Trabajo de Fin de Grado, se han podido sacar varias conclusiones:

- Los objetivos planteados inicialmente en la Sección 1.2 se han cumplido con éxito: todas las funcionalidades planteadas fueron desarrolladas y comprobadas satisfactoriamente.
- En el plano de la formación se consiguió ampliar los conocimientos en tecnologías muy presentes en el entorno empresarial, pudiendo utilizarlas de manera productiva y eficiente. Adquiriéndose experiencia en el manejo de Spring, Hibernate, servidores REST, a crear un cliente completo a través de Vue.js y a relacionar este cliente Web con el servidor Web mediante REST.
- Se ha tomado conciencia de la carga de trabajo y todo lo que supone un proyecto de la envergadura del realizado, dando importancia al uso de una metodología y una planificación para que el desarrollo sea ágil y constante.

8.2 Trabajo Futuro

Si bien la aplicación desarrollada es completa y funcional, se podría ampliar en varios puntos:

- Filtrados más avanzados, como por ejemplo que nos permita filtrar por equipos externos favoritos o de los que forma parte el usuario. El usuario en sus datos personales puede añadir los equipos externos de los que forma parte y sus equipos favoritos, es

decir, un jugador puede pertenecer a un equipo, por ejemplo, al Deportivo, pero sus equipos favoritos son el Ural y el Victoria. Lo que trata de conseguir este filtro es que me devuelva partidos dónde hay más jugadores apuntados que, externamente de la aplicación, juegan en alguno de los equipo federados en los que yo juego o en alguno de los que seleccioné como favoritos. Otro filtro interesante sería por las valoraciones que he dado anteriormente a partidos, intentando buscar un patrón de jugadores que coexisten habitualmente en esos partidos valorados con puntuaciones altas y me saque aquellos partidos vigentes donde coincida este clúster de jugadores.

- Gestión de estadísticas personales a través de GoogleFit. Se podrían almacenar después de un partido los datos estadísticos durante el período del evento deportivo de esa persona, tanto pasos dados, calorías quemadas, la media de velocidad... De esta forma el jugador en sus datos personales podría realizar un seguimiento de sus constantes y ver sus mejorías a través de unos gráficos que ofrecería la aplicación explotando los datos obtenidos a lo largo del tiempo.
- Adaptar la aplicación para poder llevar a cabo competiciones deportivas como por ejemplo una liga por equipos o torneos puntuales.

Apéndices

Apéndice A

Glosario de acrónimos

REST *Representational State Transfer.*

HTML *Cascading StyleSheets.*

CSS *HyperText Markup Language.*

MVCC *Multi Version Concurrency Control.*

IDE *Integrated Development Environment.*

HTTP *HyperText Transfer Protocol.*

API *Application Programming Interface.*

ERS *Especificación de Requisitos de Software.*

MVVM *Model–View–Viewmodel.*

JDBC *Java Database Connectivity.*

DAO *Data Access Object.*

CRUD *Create, Read, Update and Delete.*

URL *Uniform Resource Locator.*

DTO *Data Transfer Object.*

MVC *Model–View–Controller.*

IoC *Inversion Of Control.*

JSON *JavaScript Object Notation.*

BD *Base de Datos.*

HTTPS *HyperText Transfer Protocol Security.*

GIS *Geographic Information System.*

Apéndice B

Patrones

B.1 Patrones

B.1.1 Back-End

DAO

Utilizaremos los DAOs para encapsular todo el acceso a una fuente de datos. La utilidad e importancia de éste reside en la necesidad de las aplicaciones de acceder a almacenes de datos, que pueden ser reemplazados por otros. Para aumentar la mantenibilidad de la aplicación, es preciso, por tanto, separar la lógica de negocio del origen de los datos.

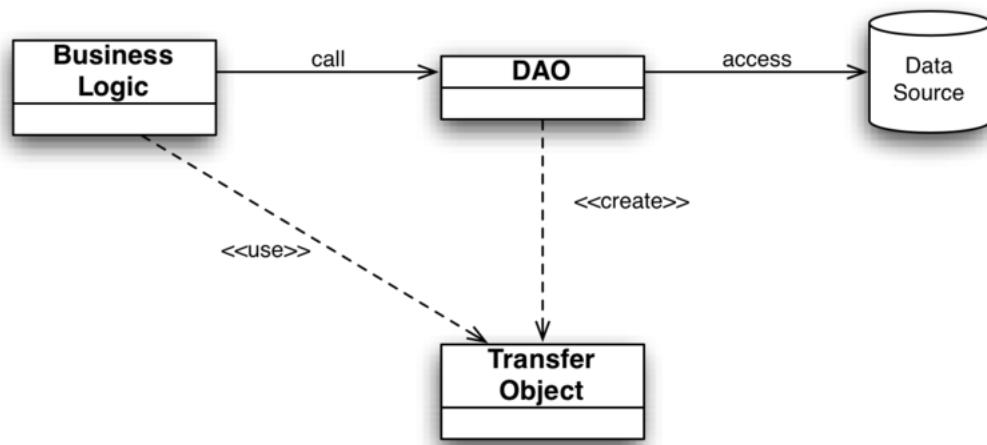


Figura B.1: Diagrama del Patrón DAO [1]

DTO

Se utilizarán los DTOs para transportar los datos desde la base de datos hacia la capa de lógica de negocio y viceversa. La primera razón de su utilización es para ocultar información al cliente, como puede ser en el caso de userDTO, de esta forma la password esta guardada en la BD pero no se pasa al lado cliente ya que es un dato crítico. Otra ventaja de este patrón es que nos permite reducir el número de llamadas remotas trayendo todos los datos necesarios en un único objeto en una única llamada.

Singleton

Garantizará que una clase solo tenga una instancia, y proporcionará un punto de acceso global a ella. Con Spring, el patrón singleton es el ámbito por defecto de un *Bean*. El contador de Spring creara una única instancia compartida de la clase asignada a ese *Bean*, por lo que siempre que se solicite ese Bean, se estará inyectando el mismo objeto. Es decir, nuestras clases con las anotaciones `@Repository` y `@Service` son Singleton (DAOs y services).

Front Controller y MVC

Spring es una implementación del patrón de diseño *FrontController*, seria concretamente el *DispatcherServlet*. También podemos hablar del patrón MVC (modelo-vista-controlador). El funcionamiento de estos 2 patrones en Spring es el siguiente:

- Las peticiones HTTP se canalizan a través del *FrontController*, que no es más que un *servlet*, y a través de la URL sabe a qué controller enviarlo.
- Se llama al controller que ejecuta la lógica de negocio. Llama al servicio que llama a los DAOs que hacen las peticiones a las BD para obtener los datos. Cuando los tiene, lo que hace es devolver los datos al servlet encapsulados en un DTO.
- Finalmente, el *FrontController* redirige la petición hacia la vista. Spring ofrece un servicio Rest al que a través de él se establecerá la conexión con la aplicación cliente (Vue).

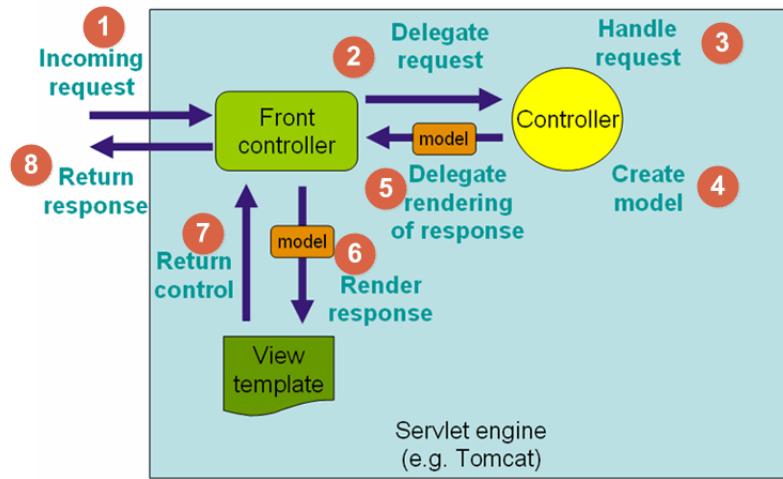


Figura B.2: Diagrama MVC Spring [2]

Fachada

El patrón Facade tiene la característica de ocultar la complejidad proporcionando una interfaz de alto nivel, la cual se encarga de realizar la comunicación con todos los subsistemas necesarios. La fachada es una buena estrategia cuando requerimos interactuar con varios subsistemas para realizar una operación concreta ya que se necesita tener el conocimiento técnico y funcional para saber qué operaciones de cada subsistema tenemos que ejecutar y en qué orden, lo que puede resultar muy complicado cuando los sistemas empiezan a crecer demasiado. Es decir, en resumen, el patrón fachada proporciona una interfaz unificada de comunicación entre subsistemas y cliente.

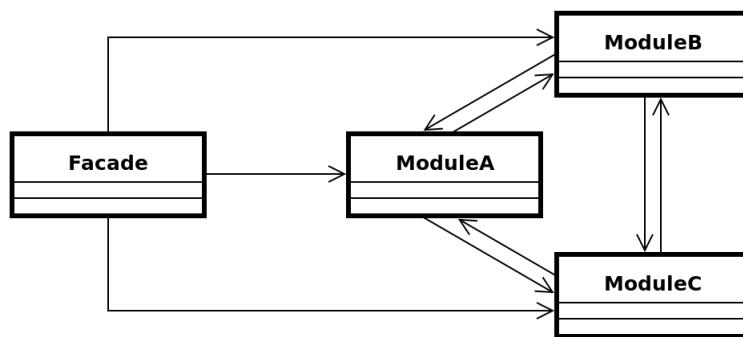


Figura B.3: Diagrama Fachada [3]

Inyección de Dependencias

El patrón de inyección de dependencias (DI, Dependency Injection) es un patrón que implementa el principio de inversión de control (IoC), donde el control se invierte estableciendo dependencias entre objetos. Esto se consigue inyectando objetos en otros objetos mediante un ensamblador, en vez de a través de los objetos mismos. La inyección de dependencias resuelve el problema de hacer que una clase sea independiente de cómo los objetos que requiere son creados.

B.1.2 Front-End

MVVM

Vue.js está basado en el patrón **MVVM (Model-View-ViewModel)**, de esta manera separamos la interfaz de usuario de la lógica de negocio enlazando las propiedades y métodos públicos con la vista.

- **Vista:** Es el HTML y el CSS, es la plantilla o estructura de la interfaz de usuario.
- **Vista-Modelo:** Es el JS asociado a la vista, es una abstracción de la vista, en la que se puede acceder a propiedades y métodos públicos. Representa el estado de los datos del dominio en un momento concreto.
- **Modelo:** El cliente REST.

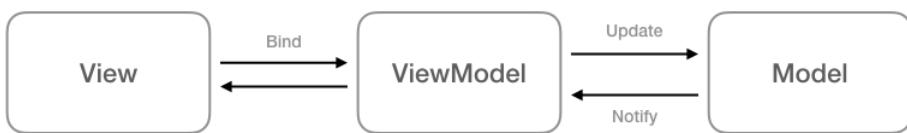


Figura B.4: Patrón Model-View-ViewModel [4]

Callback y Promise

Vue.js hace uso de los patrones **Callback y Promise**. Al incorporar estos patrones, se logra que el cliente pueda seguir trabajando hasta que la información esté disponible. Es decir, se asiste a una función que no puede devolver inmediatamente su resultado y devuelve una promesa de que tendrá el resultado en un futuro, esta promesa será gestionada por un callback.

APÉNDICE B. PATRONES

El cliente hará la petición y esa petición no devuelve inmediatamente la respuesta, así que este sigue trabajando, le promete al cliente que tendrá la respuesta, en el momento que llegue el resultado, se ejecutara el callback.

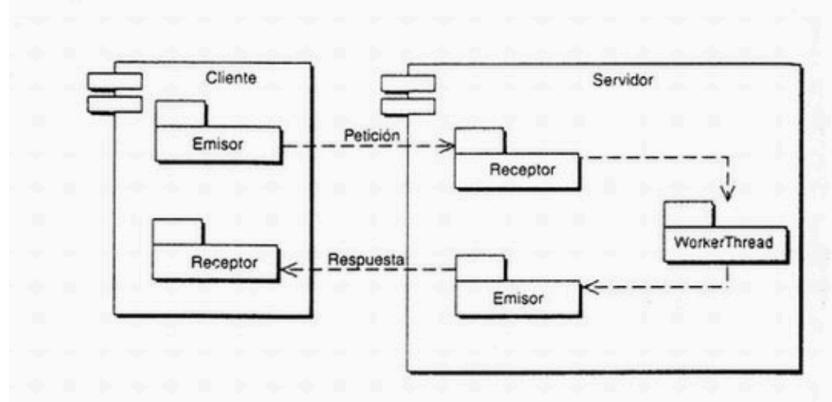


Figura B.5: Patrón Callback [5]

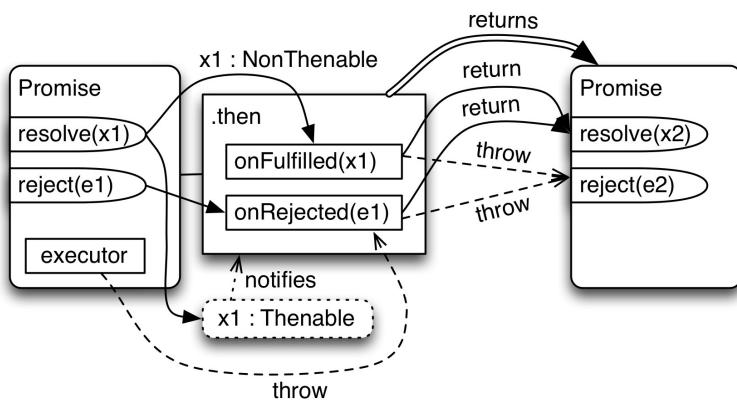


Figura B.6: Patrón Promise [6]

Apéndice C

Manual de Instalación

Para poder desplegar la aplicación en un ordenador es necesario disponer de:

- Una máquina virtual Java versión “1.8”
- Sistema de gestión de base de datos de PostgreSQL.
- Un servidor de aplicaciones, Apache Tomcat
- Node.js versión 8.12
- Apache Maven 3.3.9

Los pasos necesarios para desplegar la aplicación son:

- Crear una bases de datos llamadas tfg creando un usuario llamado asi, con la contraseña asi y que tenga como puerto el 5432. Cabe destacar que la aplicación ya crea automáticamente las tablas como se puede observar en el *application.yml* gracias a la opción *hibernate.ddl-auto: create*
- En *src/main/java/es.udc.lbd.asi.restexample.config/DatabaseLoader.java* se encuentran los datos iniciales que se cargarán en el momento de desplegarse la aplicación.
- Para arrancar el servidor, solo se ha de ir a la carpeta *rest-example-master* y ejecutar el comando *mvn spring-boot:run*
- Para arrancar el cliente, solo se ha de ir a la carpeta *vue-example-master* y ejecutar el comando *npm install* y a continuación *npm run debug* y final mente ir a un navegador y lanzar *http://localhost:1234* para ir a la aplicación. Desde ahí si vamos al apartado de Login podemos loguearnos como administrador: contraseña y usuario *pepe* o como usuario básico, contraseña y usuario *sandra*

Apéndice D

Contenido del CD

La memoria contendría un CD que contendrá:

- **DopicoCantareroSandraTFG2019.pdf:** La memoria del TFG en formato PDF.
- **DopicoCantareroSandraTFG2019resumo.pdf:** El resumen del proyecto en formato PDF.
- **DopicoCantareroSandraTFG2019anexo.zip:** Zip que contiene los archivos necesarios para el despliegue de la aplicación junto con un pequeño manual. Por un lado tenemos la carpeta *rest-example-master*, que contiene el código fuente del servidor, y por otro lado *vue-example-master*, que contiene le código fuente del cliente.

Apéndice E

Mockups

A continuación se incorporarán todos los mockups realizados en la Fase Previa, explicada en la Sección 3.1. Cabe comentar que estas pantallas son una idea previa de lo que se quería realizar para poder planificar el trabajo del proyecto y tener una base por la cual empezar, por lo que habrá cambios respecto a las pantallas de la aplicación final.



Figura E.1: Iniciar Sesión.

The image shows a registration form titled "Registrarse" (Register) on a website. The form includes fields for NAME, SURNAME, CITY, BIRTH DATE, and EMAIL. It also features fields for LOGIN and PASSWORD, and a profile picture input with a plus sign for adding a photo. A "CREAR PERFIL" (Create Profile) button is at the bottom right.

PLAY2GETHER

Registrarse

NOMBRE:

APELLIDOS:

CIUDAD:

FECHA NACIMIENTO:

CORREO ELECTRÓNICO:

LOGUIN:

CONTRASEÑA:

CREAR PERFIL

Figura E.2: Registro.



Figura E.3: Búsqueda de Partidos: Mapa.

APÉNDICE E. MOCKUPS

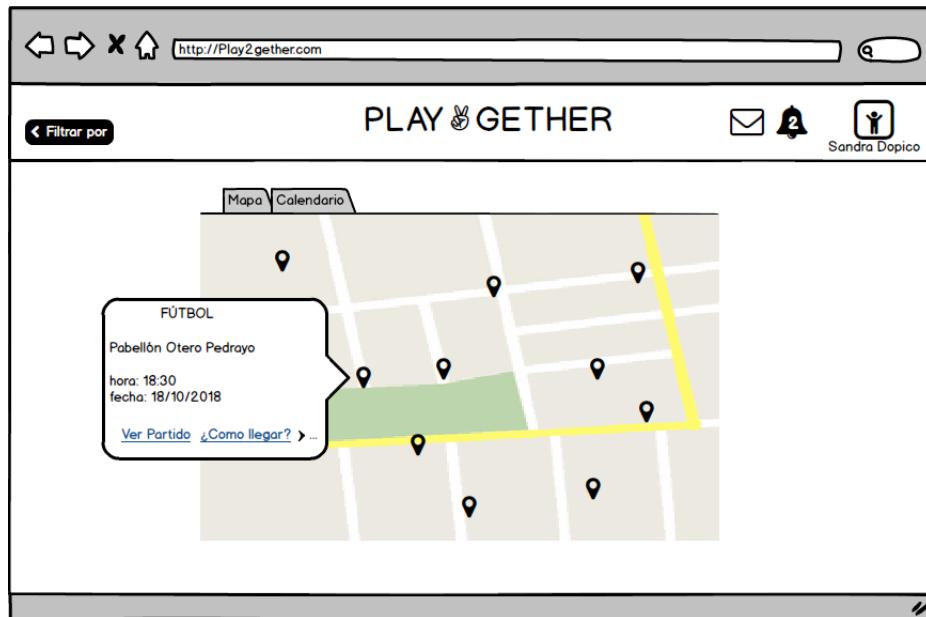


Figura E.4: Búsqueda de Partidos: Selección en el Mapa.

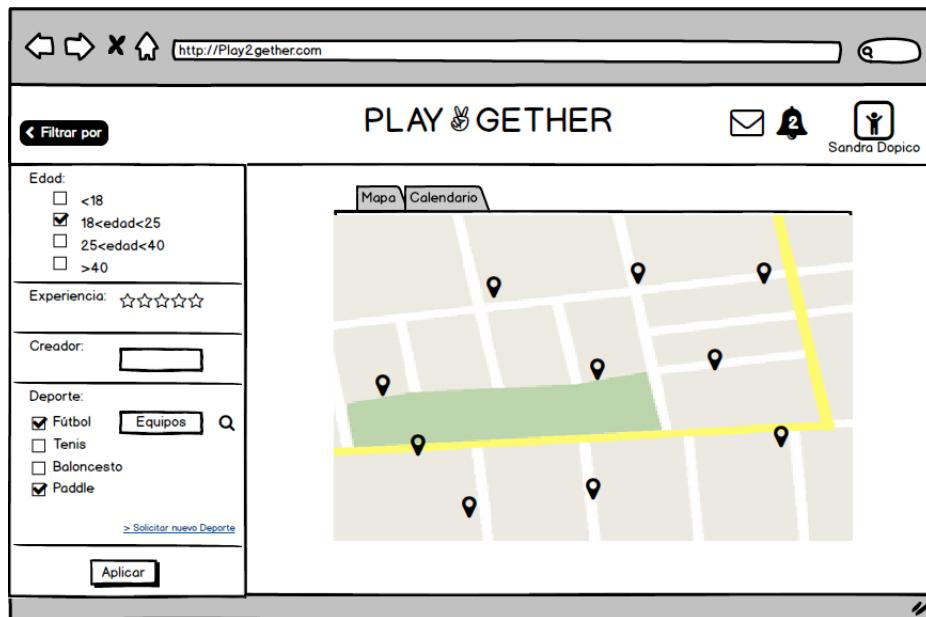


Figura E.5: Búsqueda de Partidos: Filtros.

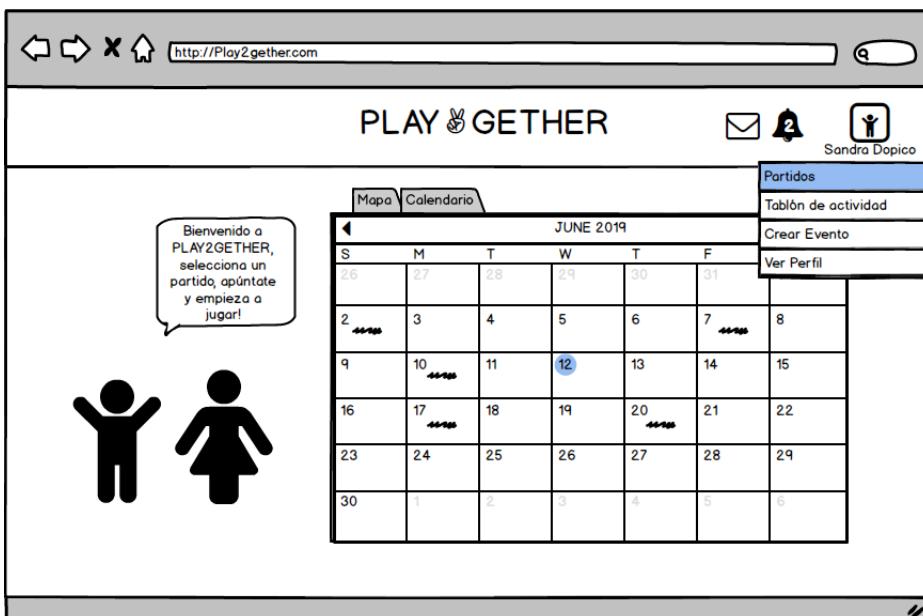


Figura E.6: Búsqueda de Partidos: Calendario.

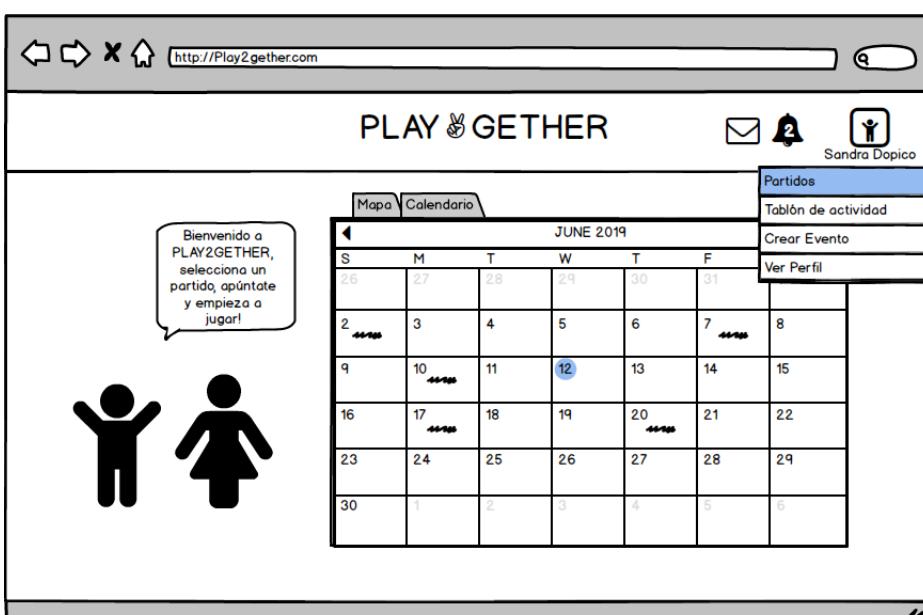


Figura E.7: Búsqueda de Partidos: Calendario.

APÉNDICE E. MOCKUPS



Figura E.8: Crear Partido: Seleccionar Fecha.



Figura E.9: Crear Partido: Seleccionar Hora.



Figura E.10: Crear Partido: Consultar Metereología.

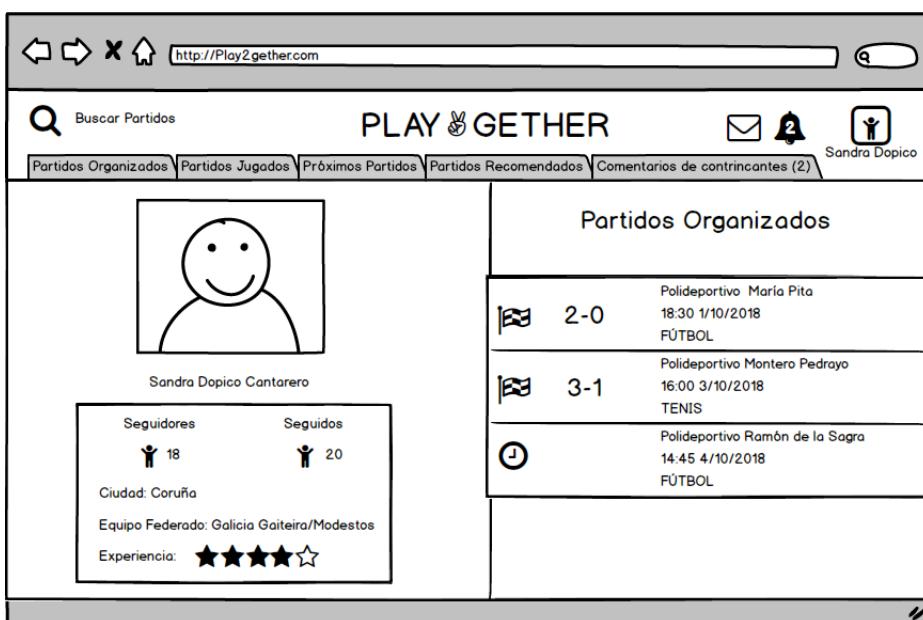


Figura E.11: Perfil Usuario: Partidos Organizados.

APÉNDICE E. MOCKUPS

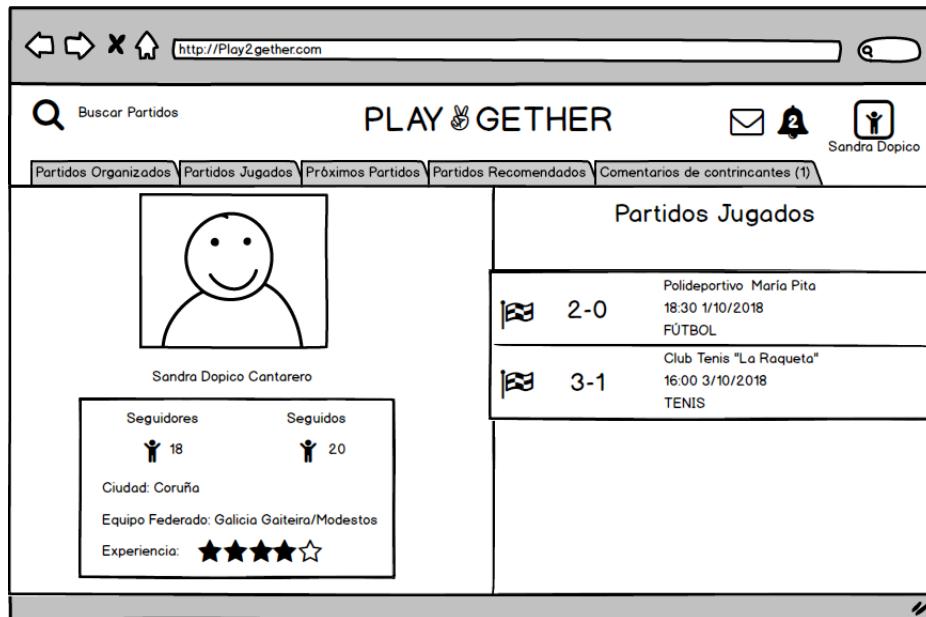


Figura E.12: Página Perfil Usuario: Partidos Jugados.

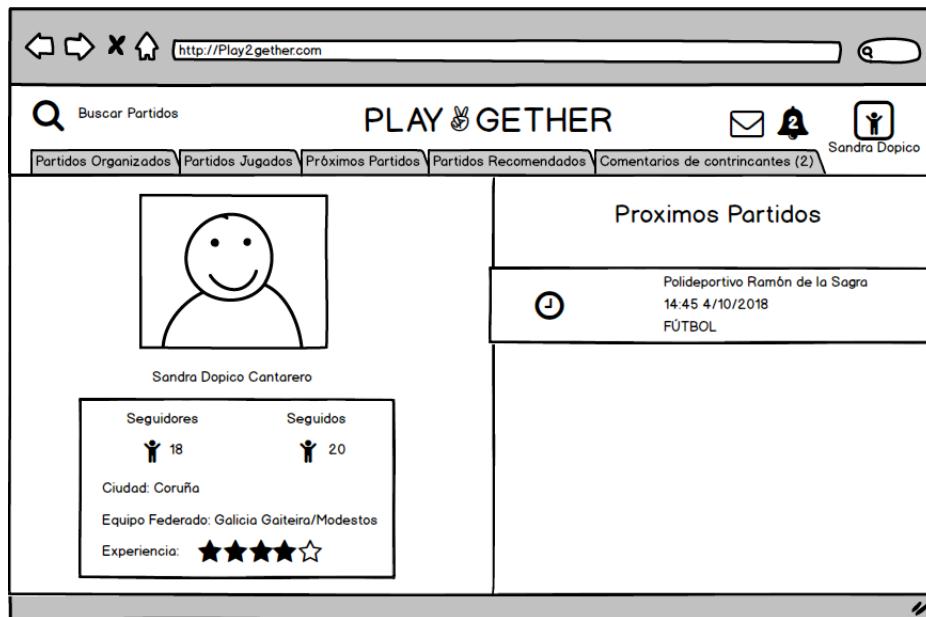


Figura E.13: Perfil Usuario: Partidos Próximos

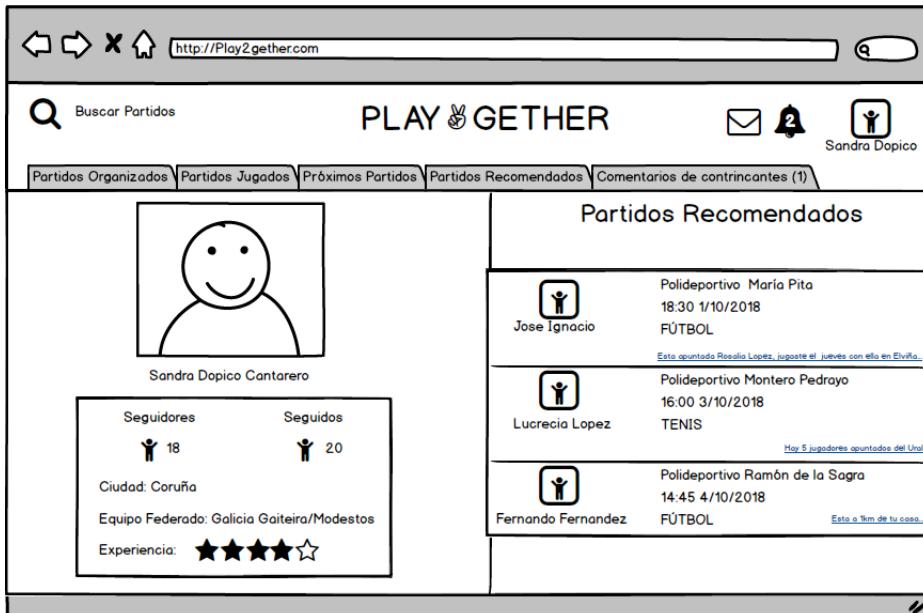


Figura E.14: Perfil Usuario: Partidos Recomendados

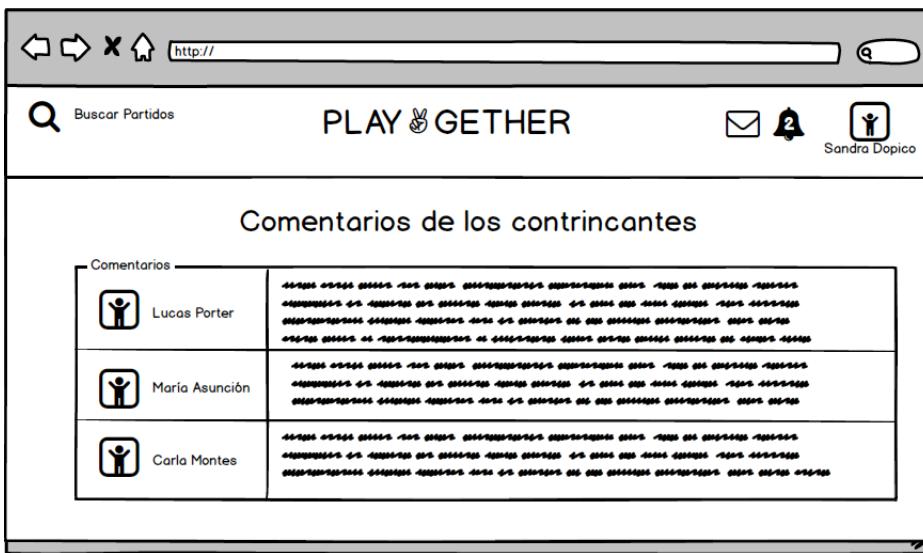


Figura E.15: Perfil Usuario: Comentarios Coontrincantes

APÉNDICE E. MOCKUPS

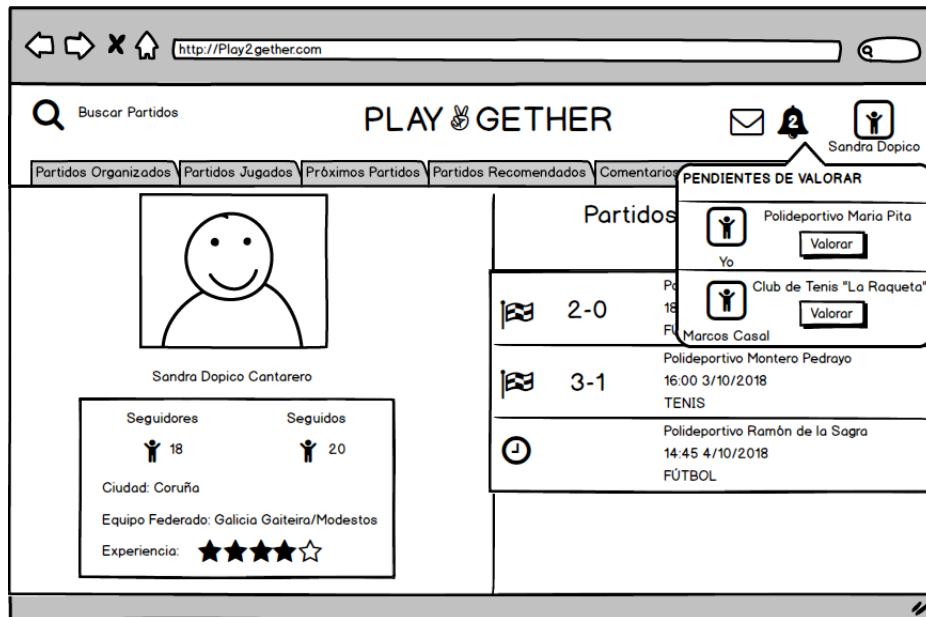


Figura E.16: Perfil Usuario: Notificaciones.

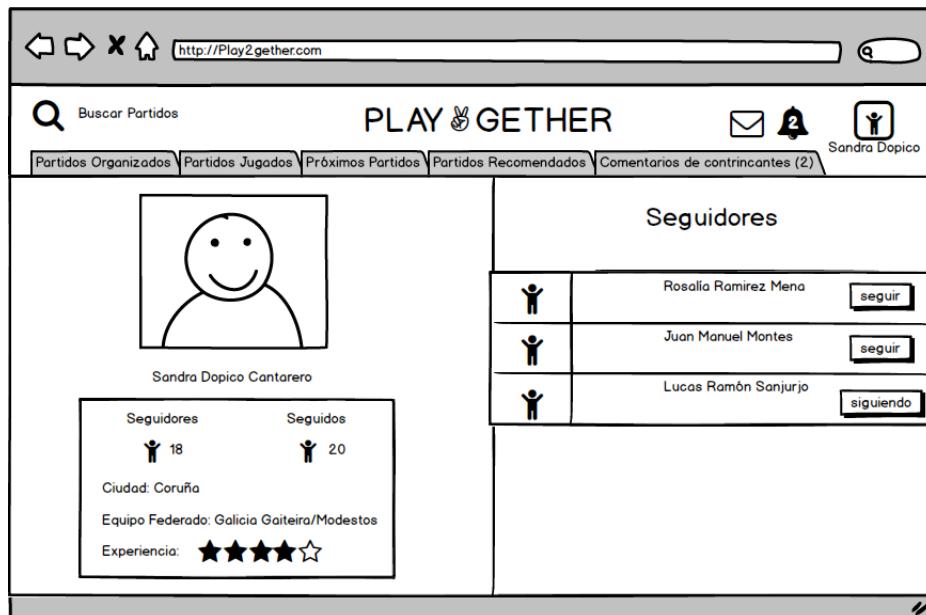


Figura E.17: Perfil Usuario: Seguidores.

The screenshot shows a user profile for 'Sandra Dopico'. At the top right, there are icons for messaging, notifications, and a user profile. The main content area has a header 'PLAY GETHER'. Below it, a search bar says 'Buscar Partidos'. A navigation bar includes 'Partidos Organizados', 'Partidos Jugados', 'Próximos Partidos', 'Partidos Recomendados', and 'Comentarios de contrincantes (1)'. On the left, there's a large profile picture of Sandra Dopico, her name, and follower counts (18 Seguidores, 20 Seguidos). Below that are her location ('Ciudad: Coruña'), team ('Equipo Federado: Galicia Gaiteira/Modestos'), and experience rating ('Experiencia: ★★★★☆'). On the right, a section titled 'Seguidos' lists three users she follows: Rosalia Ramirez Mena, Juan Pereiro Montes, and Lucas Ramón Sanjurjo, each with a 'siguiendo' button.

Figura E.18: Perfil Usuario: Seguidos.

This screenshot shows the same user profile for 'Sandra Dopico'. The top navigation bar and sidebar information are identical to Figure E.18. The main content area now features a section titled 'Partidos Organizados' on the right. It lists four organized matches with their details:

- Polideportivo María Pita, 2-0, 18:30 1/10/2018, FÚTBOL
- Polideportivo Montero Pedrayo, 3-1, 16:00 3/10/2018, TENIS
- Urbaniación Os Regos, 115-103, 15:30 5/10/2018, BALONCESTO
- Polideportivo Ramón de la Sagra, 14:45 4/10/2018, FÚTBOL

Figura E.19: Perfil Amigo.

APÉNDICE E. MOCKUPS



Figura E.20: Detalles Partido Fútbol.

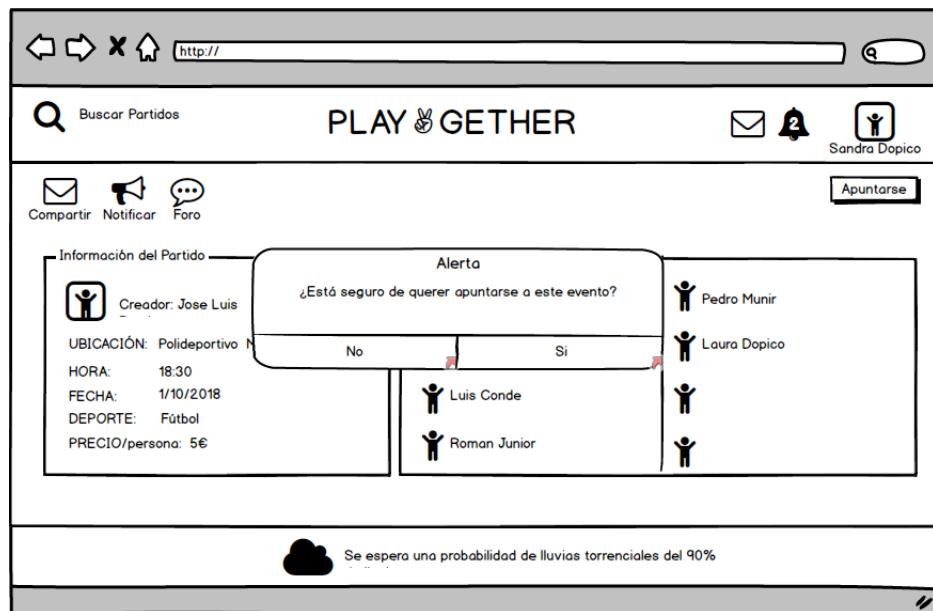


Figura E.21: Detalles Partido Fútbol: Apuntarse.

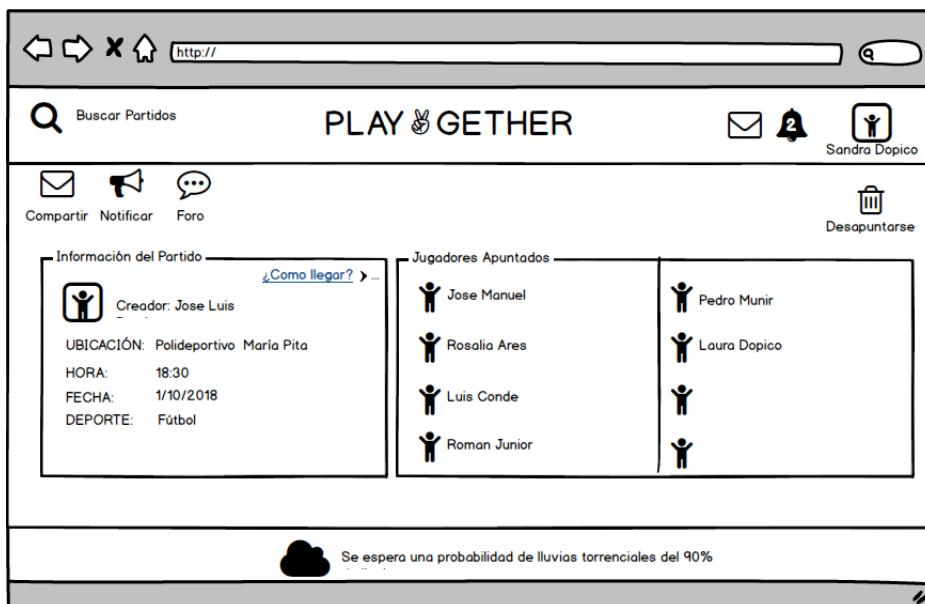


Figura E.22: Detalles Partido Fútbol: Apuntado.

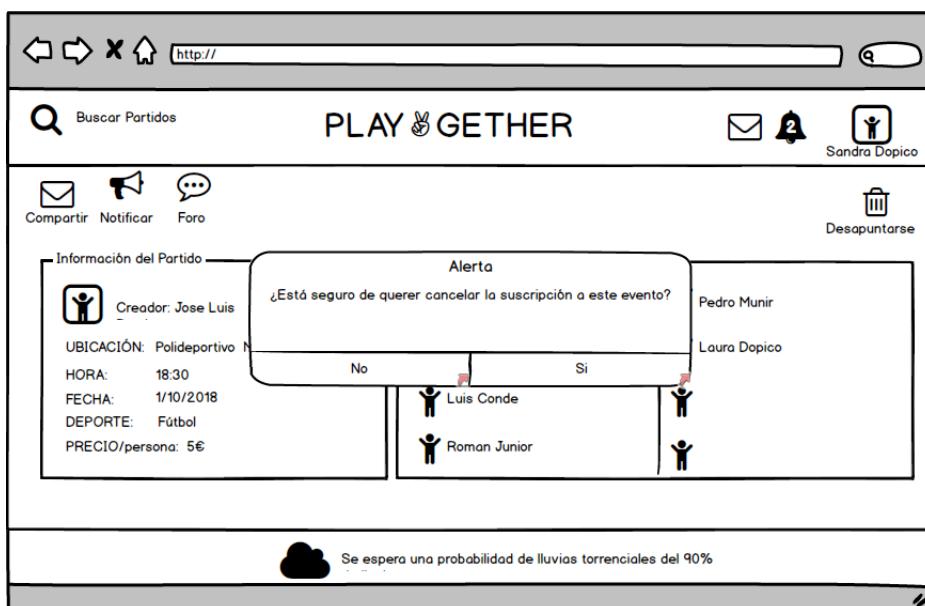


Figura E.23: Detalles Partido Fútbol: Desapuntarse.

APÉNDICE E. MOCKUPS

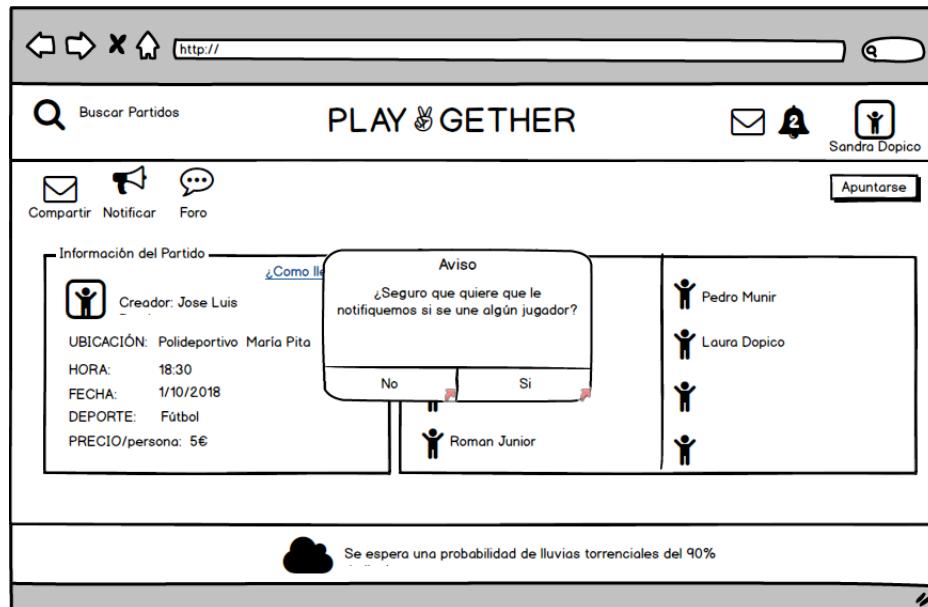


Figura E.24: Detalles Partido Fútbol: Notificación.



Figura E.25: Detalles Partido Fútbol: Notificaciones.

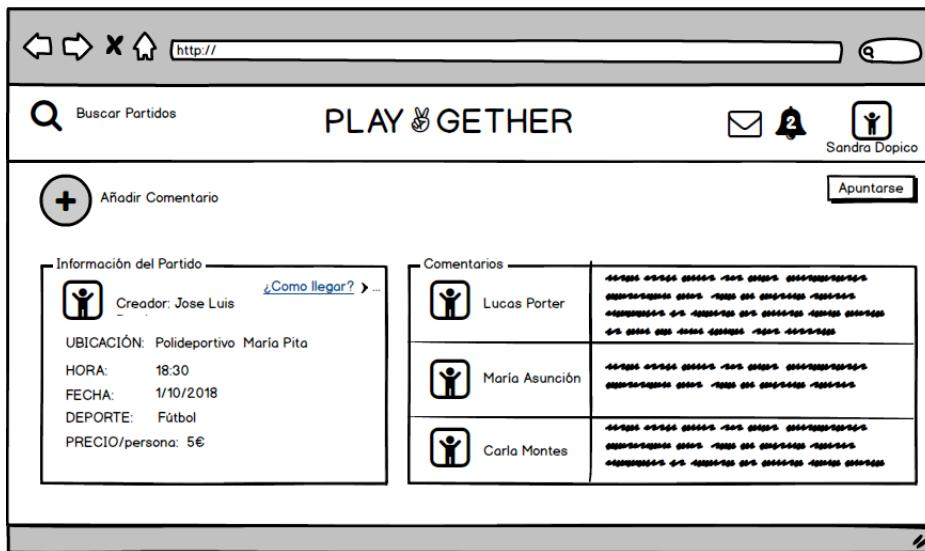


Figura E.26: Detalles Partido Fútbol: Comentarios.

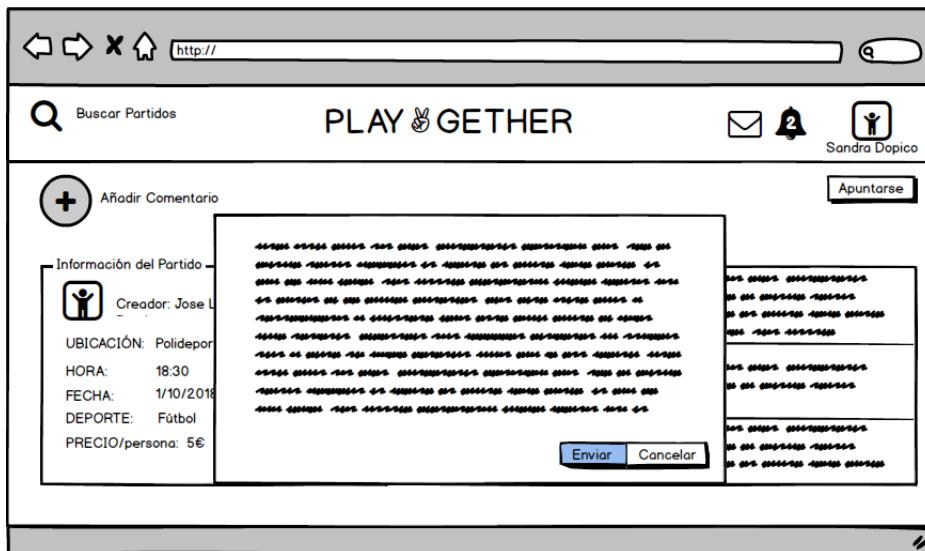


Figura E.27: Detalles Partido Fútbol: Escribir Comentario.

APÉNDICE E. MOCKUPS

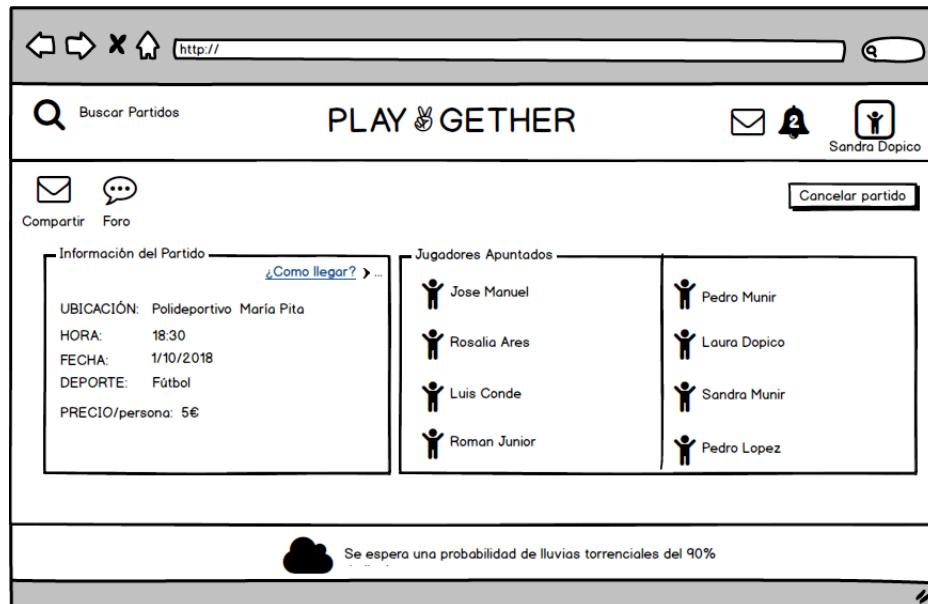


Figura E.28: Detalles Partido Fútbol: Organizado.



Figura E.29: Detalles Partido Fútbol: Rellenar Resultados.

The screenshot shows a web browser window with the URL <http://> in the address bar. The main content is a page titled "PLAY GETHHER". At the top right, there are icons for a search bar ("Buscar Partidos"), notifications ("2"), and user profile ("Sandra Dopico"). Below the title, there are buttons for "Compartir", "Notificar", and "Foro".

Información del Partido:

- Creador: Jose Luis
- UBICACIÓN: Club de Tenis "Laraqueta"
- HORA: 18:30
- FECHA: 1/10/2018
- DEPORTE: Tenis
- PRECIO/persone: 5€

Puntos Jugadores:

	1	2	3	4	5
Jose Manuel					
Roman Junior					
Pedro Munir					

Previsión Meteorológica: Se espera una probabilidad de lluvias torrenciales del 90%.

Figura E.30: Detalles Partido Tenis.

The screenshot shows a web browser window with the URL <http://> in the address bar. The main content is a page titled "PLAY GETHHER". At the top right, there are icons for a search bar ("Buscar Partidos"), notifications ("2"), and user profile ("Sandra Dopico"). Below the title, there are buttons for "Compartir" and "Foro".

RESULTADO: 3-1

Información del Partido:

- UBICACIÓN: Club de Tenis "Laraqueta"
- HORA: 18:30
- FECHA: 1/10/2018
- DEPORTE: Tenis
- VALORACIÓN: ★★★★★

Puntos Jugadores:

	1	2	3	4	5
Jose Manuel	★☆☆☆☆	6	6	2	6
Roman Junior	★☆☆☆☆				
Pedro Munir	★☆☆☆☆	2	4	6	1
Laura Dopico	★★★★★				

Figura E.31: Detalles Partido Finalizado Tenis

APÉNDICE E. MOCKUPS

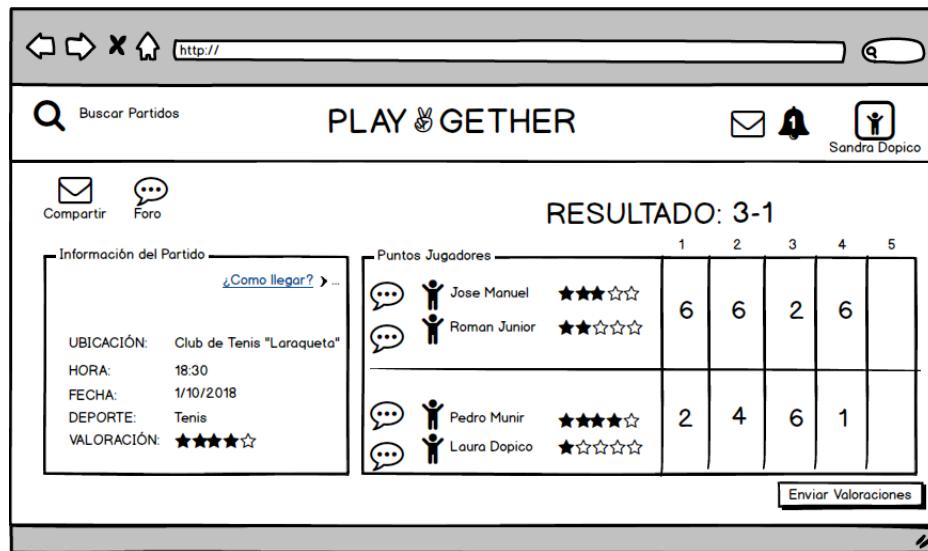


Figura E.32: Detalles Partido Finalizado Tenis: Valorar.

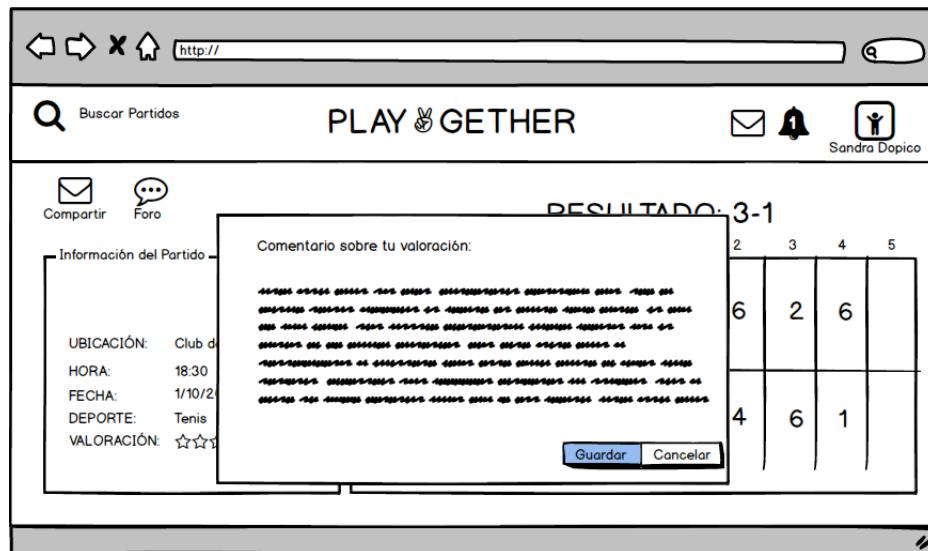


Figura E.33: Detalles Partido Finalizado Tenis: Añadir comentario a una valoración de un jugador.

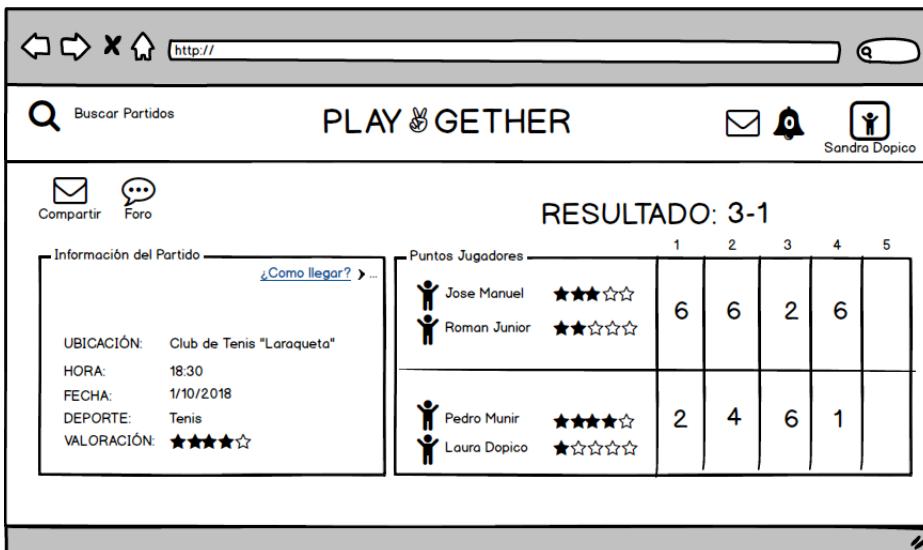


Figura E.34: Detalles Partido Finalizado Tenis: Valorado.



Figura E.35: Detalles Partido Organizado Baloncesto.

APÉNDICE E. MOCKUPS

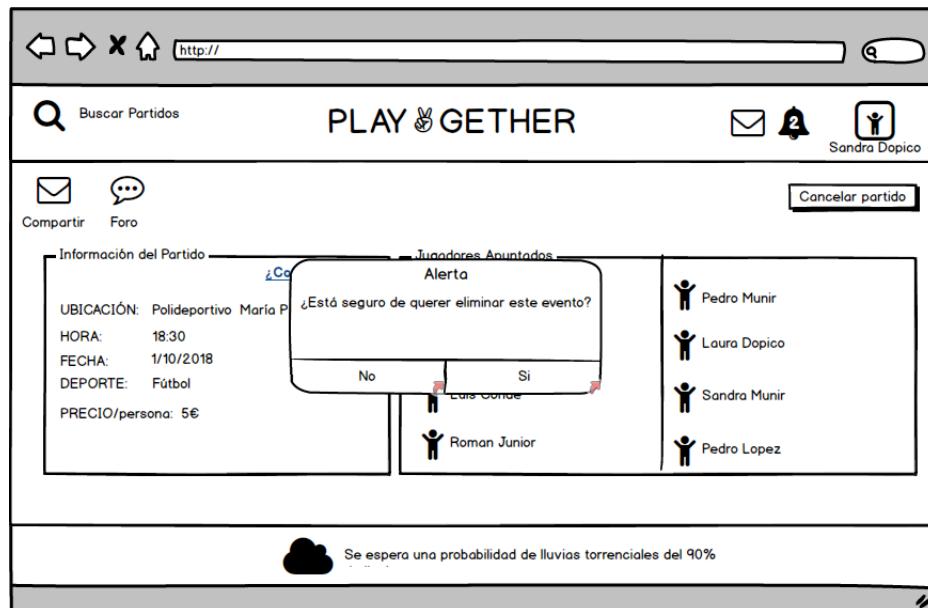


Figura E.36: Detalles Partido Organizado Baloncesto: Cancelar.



Figura E.37: Detalles Partido Finalizado Baloncesto.

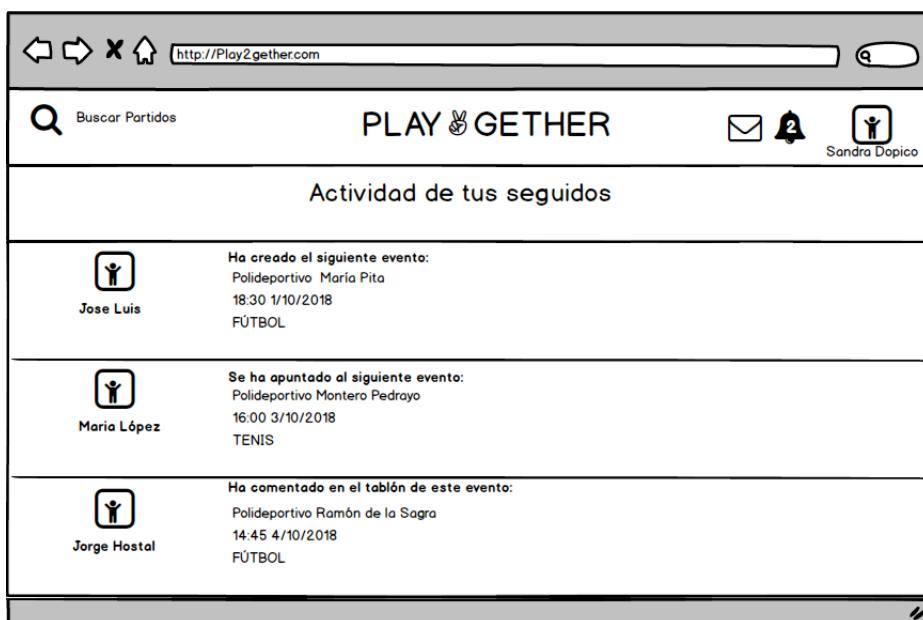


Figura E.38: Tablón de Actividades.

Bibliografía

- [1] “Página web del Diagrama del Patrón DAO,” (Consultado en Junio 2019). [Online]. Available: https://gl.wikipedia.org/wiki/Data_access_object
- [2] “Página web del Diagrama del Patrón MVC,” (Consultado en Junio 2019). [Online]. Available: <http://www.jtech.ua.es/j2ee/publico/spring-2012-13/sesion03-apuntes.html>
- [3] “Página web del Diagrama del Patrón Fachada,” (Consultado en Junio 2019). [Online]. Available: [https://es.wikipedia.org/wiki/Facade_\(patr%C3%B3n_de_dise%C3%BAo\)](https://es.wikipedia.org/wiki/Facade_(patr%C3%B3n_de_dise%C3%BAo))
- [4] “Página web del Diagrama del Patrón MVVM,” (Consultado en Junio 2019). [Online]. Available: <https://medium.com/flawless-app-stories/how-to-use-a-model-view-viewmodel-architecture-for-ios-46963c67be1b>
- [5] “Página web del Diagrama del Patrón Callback,” (Consultado en Junio 2019). [Online]. Available: <https://www.monografias.com/trabajos37/call-back/call-back2.shtml>
- [6] “Página web del Diagrama del Patrón Promise,” (Consultado en Junio 2019). [Online]. Available: <https://es.stackoverflow.com/questions/64265/qu%C3%A9-es-una-promesa-en-javascript#answer-64403>
- [7] “Página web de Timpik,” (Consultado en Junio 2019). [Online]. Available: <http://www.timpik.com/>
- [8] “Página web de Fubles,” (Consultado en Junio 2019). [Online]. Available: <https://www.fubles.com/es/>
- [9] Evan You, “Guía sobre Vue.js,” 2013, (Consultado en Febrero 2019). [Online]. Available: <https://vuejs.org/v2/guide/>
- [10] ——, “Documentación sobre Vue-Select,” 2013, (Consultado en Febrero 2019). [Online]. Available: <https://vue-select.org>

- [11] ——, “Documentación sobre Vue-MultiSelect,” 2013, (Consultado en Febrero 2019). [Online]. Available: <https://vue-multiselect.js.org>
- [12] ——, “Documentación sobre Vue-Route,” 2013, (Consultado en Febrero 2019). [Online]. Available: <https://router.vuejs.org>
- [13] Michael Thiessen, “Documentación sobre el Renderizado de Vue,” 2018, (Consultado en Febrero 2019). [Online]. Available: <https://michaelnthiessen.com/force-re-render/>
- [14] Ryan Dahl, “Página web de Node.js,” 2009, (Consultado en Febrero 2019). [Online]. Available: <https://nodejs.org/es/docs/guides/getting-started-guide/>
- [15] Vladimir Agafonkin, “Página web de Leaflet,” 2011, (Consultado en Marzo 2019). [Online]. Available: <https://leafletjs.com/>
- [16] Mark Otto, “Página web de Bootstrap,” 2010, (Consultado en Marzo 2019). [Online]. Available: <https://getbootstrap.com/docs/4.3/getting-started/introduction/>
- [17] Christian Bauer,Gavin King, and Gary Gregory, “Página web de Hibernate,” 2001, (Consultado en Febrero 2019). [Online]. Available: <https://hibernate.org>
- [18] Baeldung, “Página web sobre la herencia en Hibernate,” 2018, (Consultado en Febrero 2019). [Online]. Available: <https://www.baeldung.com/hibernate-inheritance>
- [19] CommunityDocumentation, “Página web sobre el mapeo en Hibernate,” 2004, (Consultado en Marzo 2019). [Online]. Available: <https://docs.jboss.org/hibernate/orm/3.6/reference/es-ES/html/collections.html>
- [20] Pivotal Software, “Página web de Spring,” 2002, (Consultado en Marzo 2019). [Online]. Available: <https://spring.io/projects/spring-boot>
- [21] ——, “Página web de Spring,” 2002, (Consultado en Abril 2019). [Online]. Available: <https://www.spring.io/projects/spring-security/>
- [22] Michael Stonebraker, “Página web de PostgreSQL,” 1996, (Consultado en Febrero 2019). [Online]. Available: <https://www.postgresql.org/>
- [23] “Página web de Latex,” (Consultado en Junio 2019). [Online]. Available: <https://www.latex-project.org>
- [24] “Página web de PostMan,” (Consultado en Marzo 2019). [Online]. Available: <https://www.getpostman.com/>
- [25] “Página web de Eclipse,” (Consultado en Febrero 2019). [Online]. Available: <https://www.eclipse.org>

BIBLIOGRAFÍA

- [26] “Página web de Sublime,” (Consultado en Febrero 2019). [Online]. Available: <https://www.sublimetext.com>
- [27] “Página web de MagicDraw,” (Consultado en Noviembre 2019). [Online]. Available: <https://www.nomagic.com/products/magicdraw>
- [28] “Página web de Balsamiq.” (Consultado en Noviembre 2019). [Online]. Available: <https://balsamiq.com/wireframes/>
- [29] “Página web de Draw.io,” (Consultado en Junio 2019). [Online]. Available: <https://www.draw.io>
- [30] “Página web sobre Testing en Sping Boot,” (Consultado en Junio 2019). [Online]. Available: <https://www.baeldung.com/spring-boot-testing>

