Universidad de Valladolid

ESCUELA DE INGENIERÍA INFORMÁTICA DE VALLADOLID

Grado en Ingeniería Informática de Software

Deporty

Alumno: Cristian Peñas Arias

Tutor/a/es: César Llamas Bello

Deporty, una aplicación de eventos deportivos

Cristian Peñas Arias

Índice general

Índice de figuras

Índice de cuadros

Quiero dedicar este trabajo de fin de carrera a mi madre, padre y hermano por siempre estar ahí cuando les he necesitado y por darme la oportunidad de hacer lo que me gusta. Por todo esto muchas gracias.

Agradecimientos

Muchas gracias a todos los compañeros de universidad y a todos los profesores que me han enseñado todo lo que se y no hubiera llegado hasta aquí sin muchos de ellos.

Resumen

El presente trabajo de fin de grado consiste en el desarrollo del sistema Deporty. Este desarrollo pasa por varias fases, que son: Análisis, diseño, implementación y realización de pruebas. Además cubrirá el desarrollo de los tres componentes fundamentales del proyecto: el modelo de datos, el acceso de datos y la interfaz de usuario. La finalidad principal de Deporty es permitir presenciar y retransmitir información referente a partidos de un deporte.
Es una aplicación que aporta ciertas herramientas a las personas que quieren conocer los diferentes partidos de cualquier deporte que se encuentra cerca de su ubicación. Se podrán ver tanto los partidos que se están jugando en ese momento como los que se jugarán en un futuro, además proporciona a los espectadores el resultado actual de dicho partido así como los cambios en el resultado en tiempo real.

Para ello se utiliza un bot de telegram que facilita dicha labor, y una aplicación web para poder introducir nuevos deportes con mayor facilidad. El trabajo se ha implementado utilizando MEAN stack y el API de Telegram.

Palabras clave: Aplicación web, análisis, diseño, implementación, deporte, Bot, MEAN Stack, Node.js, Express, MongoDB, Angular, Front-end, Back-end.

Abstract

The present end-of-degree project consists in the development of the Deporty system: Analysis, Design, Implementation and testing are the phases of this development. It will also cover the development of the three fundamental components of the project: the model of the data, the access to the data and the user interface. The main purpose of the Deporty is to allow the presence and the retransmission in relation to the matches of a sport.
It is an application that provides certain tools to people who want to know the different parts of any sport that is close to their location. You can see the matches that are being played at that moment as it is played or you can see the matches that will be played in the future, as well as the spectators you can see the current result of that match as well as the changes in the result in real time.

for it, a telegram bot is used to facilitate this task, and a web application to introduce new sports or teams easily. The work has been implemented using the MEAN stack and the Telegram API.

Keywords: Web aplication, amalysis,design, implementation, Sport, Bot, MEAN Stack, Node.js, Express, MongoDB, Angular, Front-end, Back-end.

Presentación

Introducción

El presente documento describe el trabajo realizado en el trabajo de fin de grado preparado para la obtención del título de Grado en Ingenieria. consiste en el desarrollo del sistema Deporty: Análisis, Diseño, Implementación y realización de pruebas. Además cubrirá el desarrollo de los tres componentes fundamentales del proyecto: el modelo de datos, el acceso de datos y la interfaz de usuario. El trabajo se ha implementado utilizando MEAN stack y el API de Telegram. MEAN Stack se utilizó para el desarrollo de la aplicación web y el API de Telegram junto con el back-end de MEAN Stack se utilizó para la creación del bot de Telegram. Haciendo uso de este sistema se pueden presenciar partidos y retransmitirlos de diferentes tipos de deportes.
Deporty, es una aplicación que aporta ciertas herramientas a las personas que quieren conocer los diferentes partidos de cualquier deporte cuya ubicación es cercana de tal forma que a parte de seguir el partido desde la aplicación se puede seguir en directo obteniendo la ubicación del partido. Se podrán ver tanto los partidos que se están jugando en ese momento, como los que se jugarán en un futuro, además proporciona a los espectadores la posibilidad de conocer el resultado actual de dicho partido así como los hacer una suscripción al partido y poder saber los cambios en el resultado en tiempo real.

Para ello se utiliza un bot de telegram que facilita dicha labor, y una aplicación web para poder introducir nuevos datos como deportes, categorías, equipos y partidos con mayor facilidad.

Los locutores serán los encargados de crear y actualizar estos partidos para que sus espectadores estén informados. También podrá cambiar el estado del partido en cualquier momento para poder informar a sus espectadores de los diferentes cambios.

Los espectadores a parte de buscar partidos, puede conocer todos los detalles del partido como las reglas que tiene dicho deporte, las categorías existentes e incluso la ubicación para verlo en directo.

Organización de la memoria

La memoria está dividida en tres partes.
La primera parte comprende los capítulos 1,2,3 y 4. Corresponde con la memoria del proyecto, se definirán objetivos, se valorará el estado del arte de la tecnología, se hará una

valoración de la metodología que se va a utilizar, de la planificación del proyecto así como de sus conclusiones finales con los resultados obtenidos. La segunda parte comprende los capítulos 5,6,7,8. Corresponde con toda la parte técnica del proyecto, es decir del análisis de la aplicación, de su posterior diseño, de su implementación y de las diferentes pruebas que se han hecho para comprobar que la aplicación funciona como se espera. La tercera y ultima parte comprende los capítulos 9 y 10. En este caso se trata de una parte en la que se explican a través de diferentes manuales la instalación para poder desarrollar el proyecto y para poder desplegarlo en caso de que haya cambios importantes, y del uso de la aplicación para los diferentes usuarios definidos en el análisis.

Capítulo 1

Descripción del proyecto

Se tratará brevemente de explicar cómo se organiza la Memoria del Trabajo Fin de Grado (TFG), del posible contenido de cada uno de los capítulos y secciones, así como de contenidos mínimos exigibles y algunas recomendaciones prácticas.

1.1. Introducción

El motivo de la elección de este tema para el trabajo de fin de grado es que siempre he sido deportista, y siempre me ha gustado la informática, entonces decidí elegir este trabajo para poder unir esos gustos en uno. Así este trabajo no solo representa mi formación durante los años sino que me representa a mi como persona y como profesional.

El propósito de este trabajo es crear facilidades a aquellas personas que les gusta ver deportes y no necesitan instalar mas aplicaciones para ello ya que con telegram instalada sera suficiente. La idea es combinar un bot de Telegram dentro del back-end de una aplicación web completa, de esta forma, Deporty dispondrá de bastantes facilidades para los locutores a la hora de crear partidos a través del servicio web. Por último, se dará una visión general de los contenidos de la Memoria.

1.2. Objetivos del trabajo

El objetivo de este trabajo como se dijo anteriormente es proporcionar facilidades a personas que les gusta observar partidos, independientemente del deporte y de si son profesionales ya que los partidos que no son profesionales no se pueden espectar a través de los medios. De esta forma podrán saber donde encontrar estos partidos, cuando se jugarán y el resultado en el caso de que haya empezado.

Deberá dar la opción a los espectadores de suscribirse a un partido y de esta forma cuando el locutor cambie cualquier dato del partido llegará una notificación a cada uno de estos espectadores con los cambios que ha habido en el partido. De esta forma se pueden saber todos los cambios de esos partidos a distancia.

Debe proporcionar una aplicación web para facilitar a los locutores la creación de los partidos ya que a través del bot sería muy complicado y podría repercutir en el número de usuarios que lo utilizarían.

Debe permitir a los locutores cambiar el estado y el resultado del partido que está transmitiendo en este caso a través del bot.

Esta aplicación podrá ser usada por cualquier persona que tenga instalado la aplicación de Telegram en su teléfono móvil y en un principio para personas que sepan Español ya que es el único lenguaje dentro de la aplicación.

Se explicará también el alcance de la aplicación, sus restricciones y limitaciones, así como las perspectivas del trabajo.

1.3. Entorno de aplicación

1.3.1. Introducción

Un deporte es una actividad reglamentada, normalmente de carácter competitivo. La Real Academia Española define un deporte como una «actividad física ejercida como juego o competición, cuya práctica supone entrenamiento y sujeción a normas».

Un bot es un programa informático que efectúa tareas repetitivas a través de Internet. Los usos mas comunes para utilizar un bot son para rastrear información en la web, dar respuestas rápidas, mantener conversaciones con los usuarios y simular tráfico en Internet y redes sociales.

La primera versión del la api del bot de Telegram fue anunciado en el año 2015. Se incorporó en la versión 3.0 de Telegram para móviles Android. En la segunda versión del bot cuyo año de lanzamiento fue 2016 mejoraron la interfaz e incluyeron un teclado dinámico. Tambien incluyeron geolocalización y la posibilidad de hacer llamadas de callback.

Hay muchos lenguajes de programación para hacer un Bot de telegram como puede ser Java, Node.js, Python, c++.

1.3.2. Aplicaciones similares

Este proyecto une los deporte con sus seguidores, de tal forma que cada vez que cambie el resultado, la aplicación avisará de los cambios a los espectadores del partido. En nuestro caso, el locutor es un usuario que puede registrarse.

Para entender el entorno se analizarán algunos sistemas que hacen una función parecida a la de esta aplicación.

Mis marcadores

Mis Marcadores es una aplicación para móviles gratuita publicada en el año 2014, en la que podrás obtener los resultados deportivos de forma rápida y precisa. Tiene varios deportes y varias ligas y torneos de todo el mundo.

Mis marcadores a parte de tener una sección resumen del partido en el que detallan los eventos del partido mas importantes, también tiene otras secciones como estadísticas del partido y alineaciones. En nuestro caso solo nos interesa la sección de resumen.

Idioma: Castellano.

Donde encontrarlo: `https://play.google.com/store/apps/details?id=eu.livesport.MisMarcadores_com`

También tiene aplicación web y su dirección es:También tiene aplicación web y su dirección es: `https://www.mismarcadores.com/`

SofaScore

SofaScore es otra aplicación para móviles gratuita publicada en el año 2015, en la que podrás obtener los resultados deportivos de forma rápida y precisa. Tiene varios deportes y varias ligas y torneos de todo el mundo.

También tiene sección de estadísticas, para poder ver ciertos datos de interés para los seguidores de los partidos.

Se pueden seguir partidos de tal forma que si cambia el resultado te envía una notificación.

Idioma: Castellano.

Donde encontrarlo: `https://play.google.com/store/apps/details?id=com.sofascore.results`

También tiene aplicación webm y su dirección es: `https://www.sofascore.com/es/`

Capítulo 2

Metodología

En este capítulo se detallarán las cuestiones metodológicas, es decir, las metodologías y herramientas que se han utilizado para plantear el trabajo.

2.1. Proceso de desarrollo

Se utilizará una metodología en cascada con varias etapas: Análisis de requisitos: Aclaración de las distintas funcionalidades de la aplicación. Diseño de software: Descomposición de los distintos elementos que deberá tener la aplicación y la comunicación entre ellos. Implementación: Implementación del diseño en código. Pruebas: Comprobación de las diferentes funcionalidades de la aplicación con usuarios reales.

2.2. Definición de siglas y abreviaturas

En esta sección se muestra una tabla con el significado de varias palabras que son siglas y con su significado.

MVC	Patrón modelo-vista-controlador se fundamenta en la separación del código en tres capas diferentes.
HTTP	Significa *Hypertext Transfer Protocol* significa protocolo de transferencia de hipertexto y permite la comunicación de información en la WWW.
WWW	Significa World Wide Web y en español red informática mundial, es un sistema de distribución de documentos en hipertexto o hipermedia interconectados y accesibles a través de Internet.
TFG	Significa trabajo de fin de grado.
NPM	Significa *The Node Package Manager*, en español el gestor de paquetes de node.
JS	Significa Javascript, es un lenguaje de programación.
CU	Significa caso de uso, es una descripción de los pasos o actividades que de deberán realizar para llevar a cabo un proceso.

UML	Significa *Unified Modeling Language*, en español lenguaje unificado de modelado y es el lenguaje de modelado de sistemas de software más conocido.
API	Significa *Application Programming Interface*, en español interfaz de Programación de Aplicaciones. El concepto hace referencia a los procesos, funciones y método que brinda una determinada biblioteca de programación.
MEAN	Significa *MongoDB, Express, Angular y NodeJS*, es una abreviatura para referirse al conjunto de estos lenguajes y librerías.
POO	Significa programación orientada a objetos.
HTML	Significa *HyperText Markup Language*, en español lenguaje de marcas de hipertexto y se utiliza para el desarrollo de páginas de Internet.
CSS	Significa *Cascading Style Sheets*, en español hojas de estilo en cascada y es un lenguaje que describe la presentación de los documentos estructurados en hojas de estilo.
SSH	Significa *Secure Shell*, es un protocolo que facilita la comunicación de forma segura entre dos sistemas.

Cuadro 2.1: Tabla de siglas y abreviaturas.

Capítulo 3

Planificación

En este capítulo se aboradarán las cuestiones relativas a la planificación del trabajo.

3.1. Estimación del esfuerzo

Para la planificación se realizará en primer lugar una estimación del coste en horas/-persona. Se hará dividiendo los diferentes procesos entre las diferentes fases, de esta forma, podremos hacer una estimación del tiempo que nos llevara cada etapa del desarrollo de la aplicación.

3.1.1. Análisis

Actividad 1.1	**Requisitos**
Actividades	Realizar los requisitos funcionales, no funcionales y de información.
Encargados	Analista.
Fecha de inicio	18/06/2018.
Fecha de fin	20/06/2018.
Esfuerzo	12 Horas/Persona.

Cuadro 3.1: Estimación de requisitos.

Actividad 1.2	**Casos de Uso**
Actividades	1. Diagrama de casos de uso. 2.Especificación de casos de uso
Encargados	Analista.
Fecha de inicio	1. 21/06/2018. 2. 21/06/2018

Fecha de fin	1. 22/06/2018. 2. 25/06/2018
Esfuerzo	1. 6 Horas/Persona. 2. 10 Horas/Persona

Cuadro 3.2: Estimación de casos de Uso.

Actividad 1.3	**Diagrama de clases**
Actividades	Realizar el diagrama de clases.
Encargados	Analista.
Fecha de inicio	26/06/2018.
Fecha de fin	27/06/2018.
Esfuerzo	6 Horas/Persona.

Cuadro 3.3: Diagrama de clases.

Actividad 1.4	**Diagramas de secuencia**
Actividades	Realizar los diagramas de secuencia.
Encargados	Analista.
Fecha de inicio	28/06/2018.
Fecha de fin	29/06/2018.
Esfuerzo	8 Horas/Persona.

Cuadro 3.4: Diagrama de actividades.

3.1.2. Diseño

Actividad 2.1	**Diagrama entidad-relación (Base de datos)**
Actividades	Realizar el diagrama de la base de datos
Encargados	Analista-Diseñador.
Fecha de inicio	02/07/2018
Fecha de fin	03/07/2018
Esfuerzo	6 Horas/Persona.

Cuadro 3.5: Estimación del diagrama entidad-relación.

Actividad 2.2	**Diagrama de clases**
Actividades	Diagrama de clases.
Encargados	Analista.
Fecha de inicio	04/07/2018.
Fecha de fin	05/07/2018.
Esfuerzo	6 Horas/Persona.

Cuadro 3.6: Diagrama de clases.

Actividad 2.3	**Diagrama de dependencias**
Actividades	Realizar el diagrama de dependencias en las diferentes partes de las aplicaciones
Encargados	Diseñador.
Fecha de inicio	06/07/2018
Fecha de fin	06/07/2018
Esfuerzo	4 Horas/Persona.

Cuadro 3.7: Estimación del diagrama de dependencias.

Actividad 2.4	**Diagramas de secuencia**
Actividades	Realizar los diferentes diagramas de secuencia para los diferentes casos de uso
Encargados	Analista-Diseñador.
Fecha de inicio	09/07/2018
Fecha de fin	11/07/2018
Esfuerzo	13 Horas/Persona.

Cuadro 3.8: Estimación de los diagramas de secuencia.

3.1.3. Implementación

Actividad 3.1	**Formacion de equipos**

Actividades	Formar al equipos en los diferentes lenguajes y frameworks.
Encargados	Programador.
Fecha de inicio	12/07/2018
Fecha de fin	23/07/2018
Esfuerzo	26 Horas/Persona.

Cuadro 3.9: Estimación de la formación de equipos.

Actividad 3.2	**Implementación del backend**
Actividades	Realizar la implementación del backend (Servicio web).
Encargados	Programador.
Fecha de inicio	24/07/2018
Fecha de fin	30/07/2018
Esfuerzo	20 Horas/Persona.

Cuadro 3.10: Estimación de la implementación del backend.

Actividad 3.3	**Implementación del frontend**
Actividades	Realizar la implementación del frontend así como de todos sus componentes.
Encargados	Programador.
Fecha de inicio	06/08/2018
Fecha de fin	16/08/2018
Esfuerzo	36 Horas/Persona.

Cuadro 3.11: Estimación de la implementación del frontend.

Actividad 3.4	**Implementación del bot de telegram**
Actividades	Realizar la implementación del bot de telegram.
Encargados	Programador.
Fecha de inicio	17/08/2018
Fecha de fin	23/08/2018
Esfuerzo	20 Horas/Persona.

Cuadro 3.12: Estimación de la implementación del bot de telegram.

3.1.4. Pruebas

Actividad 4.1	**Realización de la batería de pruebas**
Actividades	Realizar una batería de pruebas extensa para probar todas las funcionalidades de la aplicación, tanto de la aplicación web como del bot de Telegram.
Encargados	Programador.
Fecha de inicio	24/08/2018
Fecha de fin	28/08/2018
Esfuerzo	10 Horas/Persona.

Cuadro 3.13: Estimación de la batería de pruebas.

3.2. Planificación temporal

En la planificación de tareas se tendrán el cuenta los objetivos y requisitos de la aplicación, así como la estimación del esfuerzo, y también se tendra en cuenta de forma mas especifica la distinción en tareas de análisis, diseño, implementación y pruebas, asi como el coste estimado de cada una de ellas.

La parte del análisis llevará en total desde el día 18 de Junio hasta el 29 de Junio, es decir un total de 12 días. Y serán en total 42 Horas/Persona. La parte del diseño de la aplicación llevará en total desde el día 2 de Julio hasta el 11 de Julio. En total llevará 29 Horas/Persona. La Parte de la implementación durará desde el día 12 de Julio hasta el 23 de Agosto. En total llevará un 102 Horas/Persona. La parte de las pruebas durara 10 Horas/Persona y solo llevará desde el 24 de Agosto hasta el 28//

Como se puede observar la mayor parte del tiempo estará dedicado a la implementación de la aplicación con 102 horas siendo mayor que el resto de fases por las que pasará el desarrollo. El total de horas del proyecto de forma estimada deberá costar 183 horas

3.3. Presupuesto económico

Para la estimación del presupuesto se tendrán en cuenta las herramientas utilizadas (hardware y software, con los factores de impacto que correspondan a la duración del proyecto), junto con los recursos humanos necesarios, según la planificación de tareas, y el tipo de rol (analista, programador, etc) correspondiente a cada tarea. Para ello a parte de explicar estos gastos en secciones se hará una tabla con un resumen de lo que costaría la realización del proyecto.

3.3.1. Hardware y software

Todos sabemos que para crear una aplicación necesitamos disponer del *hardware* y del *software* necesario. En esta sección se habla del coste que va a suponer dichos elementos.

Para el hardware se necesta utilizar un ordenador de trabajo, para poder crear el código necesario para la aplicación. En este caso con un portatil Acer Aspire 5 A515-51G puede valer. Tiene un coste de 610€.

Para el servidor en el que alojaremos la web, en teoría este presupuesto no se debería contemplar a la hora de crear una aplicación pero igualmente se hablará de el. Amazon dispone de 1 año gratuito para aquellas personas que estén empezando y no hayan utilizado uno de sus servidores con anterioridad. En nuestro caso utilizaremos un servidor de uso general t3.small que es la nueva generación de servidores que tiene Amazon. Cuando nuestro periodo gratuito acabase tendríamos que pagar 0,02$ la hora, es decir unos 0.48$ al día o 175$ al año. Pero para reducir coste se utilizará un servidor en DigitalOcean ya que tan solo nos costaría 5€al mes, es decir 60€al año por lo que se reduciría este coste un 65 %.

Para el software utilizaremos varias cosas dependiendo de la fase en la que estemos. En la fase de análisis y diseño utilizaremos un programa de diseño que utilice el lenguaje UML. Se eligió Astah [1] dado que la licencia para estudiantes es gratuita ademas de que la universidad de Valladolid tiene licencia a disposición de los alumnos. Por lo tanto sera gratuito para la realización de este proyecto.

En la fase de implementación se utilizará visual studio code para hacer la aplicación, ya que es un *software* gratuito y muy cómodo para trabajar con MEAN. Se utilizará un servidor de MongoDB dado que proporciona software gratuito al tener una cuenta para la gestión de las bases de datos.
Se deberá tener también una licencia de Google Maps para poder utilizar su api. Esta es gratuita pero solamente para los 10 primeros usos del día.

3.3.2. Recursos humanos

Se necesita otro tipo de recursos para la creación de una aplicación, personas. Estas personas dependiendo del trabajo pueden tomar un rol, en la primera fase del proyecto se necesitará a un analista. Este analista deberá trabajar en la parte de análisis, es decir, en realizar los requisitos de la aplicación y los casos de uso. Este trabajo equivaldría de forma estimada a unas 42 horas lo que supondría un total de 630€.

En la segunda etapa necesitaremos a una persona que haga de diseñador. Este diseñador deberá trabajar en la parte de diseño, en concreto en hacer el diagrama de clases y el diseño de la base de datos, lo que haría un total de 29 horas, es decir unos 435€.

[1] http://astah.net/

La tercera etapa es la mas costosa en cuanto a tiempo y se necesita un programador. Este, deberá de hacer una formación previa para poder conocer las tecnologías que se utilizarán, y después hacer la implementación tanto de la aplicación web como del bot de Telegram. Este trabajo está estimado en 102 horas, equivaldría a 1530€.

En la ultima fase se necesitará a un *tester*. Esta persona se encargará de realizar varias pruebas para comprobar que la aplicación cumple con sus objetivos. En caso de no cumplir informará de los posibles *bugs* para su posterior corrección. Este trabajo esta estimado en 10 horas, lo que supondría un total de 150€. En resumen los recursos humanos supondrían un total de 2745€.

3.3.3. Presupuesto total del desarrollo

Este es el presupuesto teórico del proyecto deporty. Los unicos valores que pueden variar son los recursos humanos, en función de las horas que lleve realizar todas las funcionalidades.

	Coste / Unidad (€)	**Número de unidades**	**Coste estimado (€)**
1. Recursos humanos			2745
Salario de empleados	15	146	2745
2. Hardware			610
Ordenador de trabajo	15	1	610
3. Software			0
Herramientas de desarrollo (Mean stack)	0	1	0€
Herramientas de diseño (Astah licencia uva)	0	1	0
Servidor base de datos (MongoDB free version)	0	1	0
Licencia google maps (free version)	0	1	0
4. Otros materiales			150
Consumibles informático (Wadgets)			30
Materiales de oficina			20
Suministros generales (luz,agua,teléfono,wifi)			100
Presupuesto total			3505

Para poner desplegar la aplicación necesitaremos varias herramientas que tienen varios costes. Se tendrá en cuenta el presupuesto teniendo en cuenta los costes de mantenimiento de la aplicación y los servidores durante un año.

	Coste / Unidad (€)	**Número de unidades**	**Coste estimado (€)**
1. Recursos humanos			1500
Salario de empleados	15	100	1500
2. Hardware			68
Servidor VPS	5	12	60
Servidor MongoDB	0	12	0
Dominio DNS	8	1	8
Almacenamiento repositorio	7	1	84
Presupuesto del despliegue en producción			1652

Este presupuesto puede aumentar si hay mucho tráfico en los servidores y necesita aumentar la capacidad de estos.

Total de coste del desarrollo del proyecto, junto con el despliegue y el mantenimiento del mismo seria de 5157€.

Capítulo 4

Análisis

4.1. Funcionalidades

Las diferentes funcionalidades de la aplicación son las siguientes:
Para un usuario no registrado(UNR):

- Identificarse (UNR-CU01)
- Registrarse (UNR-CU02)
- Espectar partido (UNR-CU03)
- Consultar datos del deporte (UNR-CU04)
- Consultar equipos que participan en un deporte (UNR-CU05)
- Consultar reglas de un deporte (UNR-CU06)
- Consultar datos de un partido cercano (UNR-CU07)

Para un usuario registrado con rol de administrador(URA):

- Crear equipo (URA-CU01)
- Modificar equipo (URA-CU02)
- Crear deporte (URA-CU03)
- Modificar deporte (URA-CU04)
- Introducir regla a deporte (URA-CU05)
- Modificar regla (URA-CU06)
- Añadir categoría a deporte (URA-CU07)

- Modificar categoría (URA-CU08)
- Eliminar deporte (URA-CU09)
- Eliminar equipo (URA-CU10)
- Eliminar regla (URA-CU11)
- Eliminar categoría (URA-CU12)

Para un usuario registrado con rol de espectador(URE):

- Consultar lista de deportes (URE-CU01)
- Consultar lista de equipos (URE-CU02)

Para un usuario registrado con rol de espectador(URL):

- Crear partido (URL-CU01)
- Cambiar estado de un partido (URL-CU02)
- Introducir resultado del partido (URL-CU03)
- Introducir equipo a partido (URL-CU04)
- Eliminar partido (URL-CU05)
- Modificar datos de un partido (URL-CU06)
- Eliminar equipo de un partido (URL-CU07)

La Figura 4.1 representa estas funcionalidades en un diagrama.

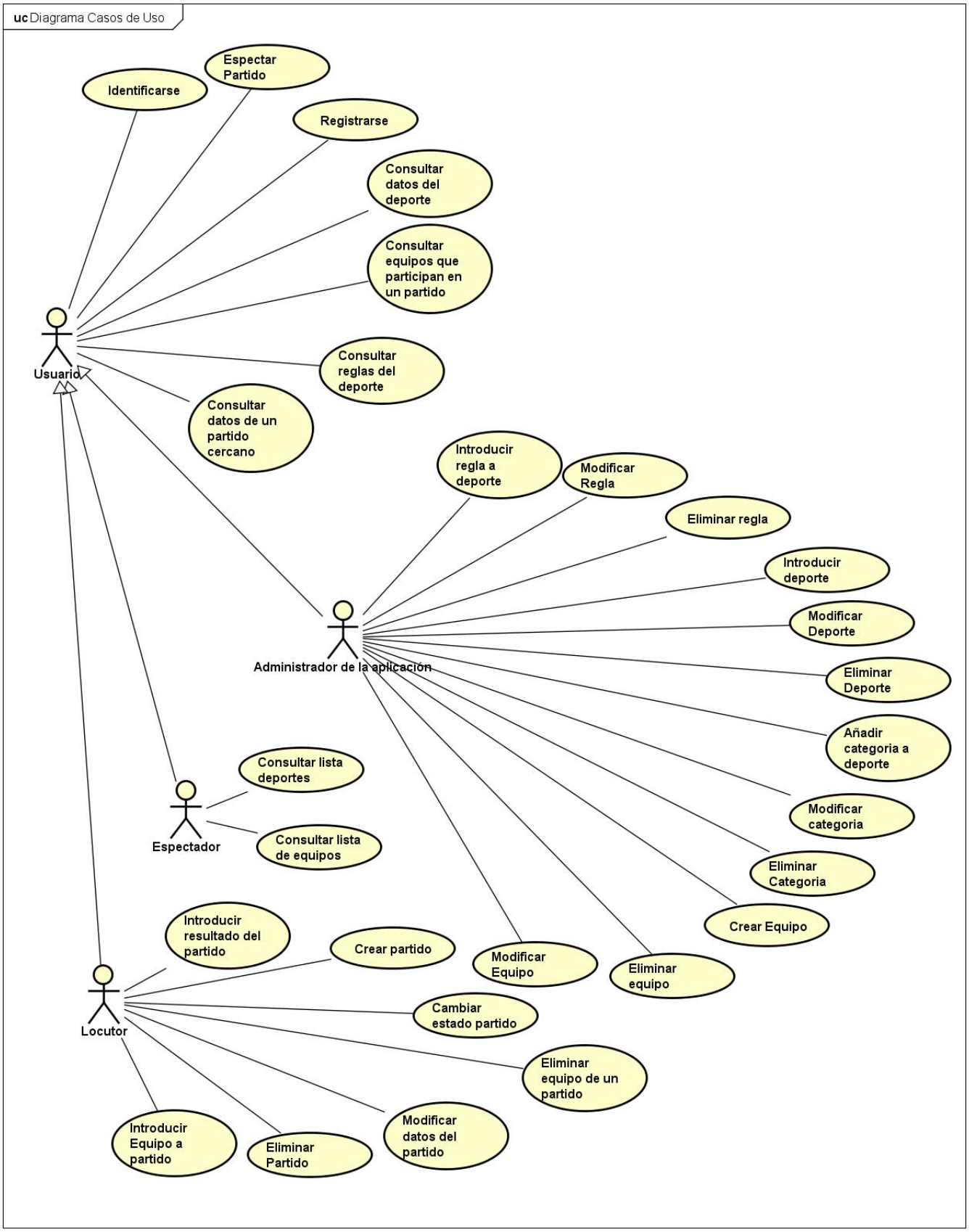


Figura 4.1: Diagrama de casos de uso.

4.2. Requisitos

En este apartado se describen los requisitos del sistema que se va a desarrollar: requisitos funcionales, de interfaz de usuario, de información, etc. Asimismo, se especificará si la aplicación debe ser o no multiplataforma (o multilingüe), si se establecen restricciones de uso o de otro tipo (por ejemplo heredadas del entorno organizativo en el que se integrará), etc.

Para una mejor descripción de los requisitos, se usan los llamados casos de uso, basados en la identificación de actores y tareas.

Funcionales

Los Requisitos funcionales describen el funcionamiento del sistema y los servicios que han de proporcionar. En este apartado, se definirán este tipo de requisitos indicando las diferentes situaciones que el sistema deberá permitir. Las siguientes tablas definen los requisitos funcionales básicos que tiene que cumplir la aplicación.

RF-01	**Gestión de acceso**
Descripción	El sistema deberá gestionar el acceso al sistema teniendo en cuenta la seguridad de los usuarios y del propio sistema.
Estabilidad	Alta.
Comentarios	La gestión del acceso al sistema se controla mediante cuentas de usuario con su respectivo correo electrónico y su contraseña y del control de sesiones.

Cuadro 4.1: Gestión de acceso.

RF-02	**Alta de usuarios**
Descripción	El sistema deberá permitir la creación de nuevos usuarios.
Estabilidad	Alta.
Comentarios	Una persona sin usuario podrá registrarse en el sistema teniendo que introducir sus datos como correo electrónico y su contraseña.

Cuadro 4.2: Alta de usuarios.

RF-03	**Modificación de los datos de usuarios**
Descripción	El sistema deberá permitir modificar los datos de usuario.
Estabilidad	Alta.

Comentarios	Una persona solo podrá modificar los datos de su usuario.

Cuadro 4.3: Modificación de usuarios.

RF-04	**Alta de partidos**
Descripción	El sistema deberá permitir la creación de partidos a los usuarios con el rol locutor.
Estabilidad	Alta.
Comentarios	Para dar de alta un partido se necesitan diferentes datos como la ubicación, el día y la hora en el que empieza y el resultado.

Cuadro 4.4: Alta de partidos.

RF-05	**Modificación de un partido**
Descripción	El sistema deberá permitir la modificación de partidos activos a los usuarios que estén transmitiendo el partido con el rol locutor.
Estabilidad	Alta.
Comentarios	Para modificar un partido se necesitan diferentes datos como la ubicación, el día y la hora en el que empieza y el resultado.

Cuadro 4.5: Modificación de un partido.

RF-06	**Eliminación de un partido**
Descripción	El sistema deberá permitir la eliminación de partidos a usuarios que estén transmitiendo el partido con el rol locutor.
Estabilidad	Alta.
Comentarios	Al eliminar el partido se eliminarán también las participaciones de los equipos y los espectadores.

Cuadro 4.6: Eliminación de un partido.

RF-06	**Consultar datos de un partido**
Descripción	El sistema deberá permitir consultar la información de un partido a cualquier usuario.
Estabilidad	Alta.

Comentarios	Ninguno.

Cuadro 4.7: Consta de datos de un partido.

RF-07	**Alta de un deporte**
Descripción	El sistema deberá permitir la creación de un deporte si el usuario tiene el rol de administrador.
Estabilidad	Alta.
Comentarios	Para dar de alta un deporte se necesitaran datos como nombre, numero máximo de equipos y descripción.

Cuadro 4.8: Alta de un deporte.

RF-08	**Modificación de un deporte**
Descripción	El sistema deberá permitir la modificación de un deporte si el usuario tiene el rol de administrador.
Estabilidad	Alta.
Comentarios	Para modificar un deporte se necesitaran datos como nombre, numero máximo de equipos y descripción.

Cuadro 4.9: Modificación de un deporte.

RF-09	**Eliminación de un deporte**
Descripción	El sistema deberá permitir la eliminación de un deporte si el usuario tiene el rol de administrador.
Estabilidad	Alta.
Comentarios	Se borrarán también las reglas, las categorías y los partidos.

Cuadro 4.10: Eliminación de un deporte.

RF-10	**Consulta de datos de un deporte**
Descripción	El sistema deberá permitir consultar los datos de cualquier deporte a cualquier usuario.
Estabilidad	Alta.
Comentarios	Ninguno.

Cuadro 4.11: Consulta de datos de un deporte.

RF-11	**Añadir una categoría a un deporte**
Descripción	El sistema deberá permitir añadir categorías de los diferentes deportes si el usuario tiene el rol de administrador.
Estabilidad	Alta.
Comentarios	Una categoría de un deporte describe las diferentes clasificaciones en las que se puede dividir un deporte.

Cuadro 4.12: Añadir una categoría a un deporte.

RF-11	**Eliminación de una categoría de un deporte**
Descripción	El sistema deberá permitir eliminar categorías de los diferentes deportes si el usuario tiene el rol de administrador.
Estabilidad	Alta.
Comentarios	Ninguno.

Cuadro 4.13: Eliminación de una categoría de un deporte.

RF-12	**Añadir una reglas a un deporte**
Descripción	El sistema deberá permitir añadir reglas a cualquier deporte si el usuario tiene el rol de administrador.
Estabilidad	Alta.
Comentarios	Una regla de un deporte describe que tipo de acciones están permitidas y cuales no, así como más tipos de información.

Cuadro 4.14: Añadir una regla a un deporte.

RF-13	**Eliminación de una reglas de un deporte**
Descripción	El sistema deberá permitir eliminar una regla de cualquier deporte si el usuario tiene el rol de administrador.
Estabilidad	Alta.
Comentarios	Ninguno

Cuadro 4.15: Eliminación de una regla de un deporte.

RF-14	**Alta de equipos**
Descripción	El sistema deberá permitir crear un equipo si el usuario tiene el rol de administrador.
Estabilidad	Alta.
Comentarios	Para dar de alta a un equipo se deberán tener los datos de nombre,tag y deporte.

Cuadro 4.16: Alta de equipos.

RF-15	**Modificación de equipos**
Descripción	El sistema deberá permitir modificar un equipo si el usuario tiene el rol de administrador.
Estabilidad	Alta.
Comentarios	Para modificar un equipo se deberán tener los datos de nombre,tag y deporte.

Cuadro 4.17: Modificación de equipos.

RF-16	**Eliminación de equipos**
Descripción	El sistema deberá permitir eliminar un equipo si el usuario tiene el rol de administrador.
Estabilidad	Alta.
Comentarios	Ninguno.

Cuadro 4.18: Eliminación de equipos.

RF-17	**Inscribir un equipo a un partido**
Descripción	El sistema deberá permitir al usuario locutor inscribir a un equipo a un partido del deporte que tiene asignado dicho equipo.
Estabilidad	Alta.

Comentarios	Para inscribir un equipo a un partido se hará a través de la aplicación web.

Cuadro 4.19: Inscribir un equipo a un partido.

RF-18	**Cancelación de una inscripción de un equipo a un partido**
Descripción	El sistema deberá permitir al usuario locutor cancelar una inscripción de un equipo a un partido en el que este inscrito.
Estabilidad	Alta.
Comentarios	Para cancelar una inscripción de un equipo a un partido se hará a través de la aplicación web.

Cuadro 4.20: Cancelación de una inscripción de un equipo a un partido.

RF-19	**Buscar partidos en la zona**
Descripción	El sistema deberá permitir buscar partidos cercanos conociendo la ubicación de un usuario.
Estabilidad	Alta.
Comentarios	Para buscar partidos en la zona se utilizará el comando /partidosCercanos a través del bot de Telegram.

Cuadro 4.21: Buscar partidos en la zona.

RF-20	**Consultar ubicación de un partido**
Descripción	El sistema deberá permitir a cualquier usuario consultar la ubicación de un partido.
Estabilidad	Alta.
Comentarios	Para conocer la ubicación de un partido se utilizará el comando /ubicacionPartido a través del bot de Telegram o través de la página web.

Cuadro 4.22: Consultar ubicación de un partido.

RF-21	**Presenciar un partido**

Descripción	El sistema deberá permitir a cualquier usuario presenciar un partido desde la aplicación de Telegram mediante un bot.
Estabilidad	Alta.
Comentarios	Para presenciar un partido se utilizará el comando /espectarPartido a través del bot de Telegram.

Cuadro 4.23: Presenciar un partido.

RF-22	**Dejar de presenciar un partido**
Descripción	El sistema deberá permitir a cualquier usuario dejar de presenciar un partido desde la aplicación de Telegram mediante un bot.
Estabilidad	Alta.
Comentarios	Para dejar de presenciar un partido se utilizará el comando /desespectarPartido a través del bot de Telegram.

Cuadro 4.24: Dejar de presenciar un partido.

RF-23	**Consultar espectadores de un partido**
Descripción	El sistema deberá permitir al usuario locutor consultar los espectadores del partido que tiene activo desde la aplicación de Telegram mediante un bot.
Estabilidad	Alta.
Comentarios	Para consultar los espectadores de un partido se utilizará el comando /consultarEspectadores a través del bot de Telegram.

Cuadro 4.25: Consultar espectadores de un partido.

RF-24	**Asociar cuenta de Telegram a un usuario**
Descripción	El sistema deberá permitir a un usuario registrado en el sistema enlazar su cuenta de Telegram con la del sistema a través de la generación de un *token*.
Estabilidad	Alta.
Comentarios	El *token* se generará desde la aplicación web y mandará un mensaje con su valor al id de Telegram introducido, si se introduce correctamente quedarán enlazados.

Cuadro 4.26: Asociar cuenta de Telegram a un usuario.

RF-25	Desasociar cuenta de Telegram a un usuario
Descripción	El sistema deberá permitir a cualquier usuario de Telegram desasociar su cuenta de Telegram con la del usuario que tenía asociada la cuenta del sistema, a través del bot.
Estabilidad	Alta.
Comentarios	Se utilizará el comando /reiniciarId para eliminar los datos del id de Telegram en el sistema.

Cuadro 4.27: Desasociar cuenta de Telegram a un usuario.

RF-26	Autorizar el envío mensajes desde la aplicación web
Descripción	El sistema deberá permitir a cualquier usuario de telegram autorizar al sistema para que desde la aplicación web pueda enviar mensajes al id de Telegram correspondiente a traves del bot.
Estabilidad	Alta.
Comentarios	Se utilizará el comando /autorizaToken para permitir al sistema el envió de mensajes al id de Telegram desde la web.

Cuadro 4.28: Autorizar el envío mensajes desde la aplicación web.

RF-27	Desautorizar el envío mensajes desde la aplicación web
Descripción	El sistema deberá permitir a cualquier usuario de Telegram desautorizar al sistema para que desde la aplicación web pueda enviar mensajes al id de Telegram correspondiente a traves del bot.
Estabilidad	Alta.
Comentarios	Se utilizará el comando /desautorizaToken para permitir al no permitir el envió de mensajes al id de Telegram desde la web.

Cuadro 4.29: Desautorizar el envío mensajes desde la aplicación web.

No funcionales

Los requisitos no funcionales son, requisitos que especifican criterios para juzgar la calidad de una operación de un sistema. Por tanto, se refieren a todos los requisitos que no describen información, ni funciones, sino características de funcionamiento o restricciones.

Las siguientes tablas definen los requisitos no funcionales que tiene que cumplir el sistema.

RNF-01	**Acceso concurrente a bases de datos**
Descripción	El sistema deberá asegurar que múltiples accesos se puedan realizar de forma concurrente sin que estas puedan causar lentitud notable en el sistema. Los procesos deben progresar constantemente y sin interrupción, así como las acciones de intercambio de información.
Estabilidad	Media.
Comentarios	Ninguno.

Cuadro 4.30: Acceso concurrente a bases de datos.

RNF-02	**Usabilidad y accesibilidad**
Descripción	El sistema deberá hacer que los objetos, acciones y opciones sean visibles siempre y cuando corresponda. El usuario no tiene por qué recordar información. El usuario siempre debe saber exactamente qué es lo que el sistema está haciendo. El sistema debe hablar siempre en el lenguaje del usuario. Las frases, palabras y conceptos deben de ser familiares para el usuario, haciendo que la información aparezca en un orden lógico y natural.
Estabilidad	Media.
Comentarios	Ninguno.

Cuadro 4.31: Usabilidad y accesibilidad.

RNF-03	**Disponibilidad**
Descripción	El sistema deberá garantizar que cuando un usuario quiera conectarse con el sistema este, deberá estar disponible al menos el 95 % del tiempo.
Estabilidad	Media.
Comentarios	Ninguno.

Cuadro 4.32: Disponibilidad.

RNF-04	**Seguridad**

Descripción	El sistema deberá garantizar la seguridad de los datos de los diferentes usuarios, para ello la información sensible como las contraseñas irán cifradas.
Estabilidad	Alta.
Comentarios	Ninguno.

Cuadro 4.33: Seguridad.

De información

IRQ-01	**Usuario**
Descripción	El sistema deberá almacenar, gestionar y proteger la información de los usuarios, así como los datos personales requeridos para el sistema.
Estabilidad	Alta.
Datos	Datos personales, contraseñas, correos, id de Telegram y sus partidos asociados.

Cuadro 4.34: Usuario.

IRQ-02	**Tipo de rol**
Descripción	El sistema deberá almacenar, gestionar y proteger la información de los diferentes tipos de roles dentro del sistema.
Estabilidad	Alta.
Datos	Espectador, locutor y administrador.

Cuadro 4.35: Tipo de rol.

IRQ-03	**Partido**
Descripción	El sistema deberá almacenar, gestionar la información de los partidos dentro del sistema.
Estabilidad	Alta.
Datos.	Ubicación, días de inicio y fin, el resultado y el estado del partido.

Cuadro 4.36: Partido.

IRQ-04	Equipo
Descripción	El sistema deberá almacenar, gestionar la información de los equipos dentro del sistema, incluidos los partidos en los que participan.
Estabilidad	Alta.
Datos	Nombre, tag o siglas, imagen y descripción.

Cuadro 4.37: Equipo.

IRQ-05	Estado del partido
Descripción	El sistema deberá almacenar, gestionar la información del estado de los partidos dentro del sistema.
Estabilidad	Alta.
Datos	Los diferentes estados son: Sin empezar, en curso, anulado y terminado.

Cuadro 4.38: Estado del partido.

IRQ-06	Deporte
Descripción	El sistema deberá almacenar, gestionar la información del estado de los deportes dentro del sistema, incluidas sus reglas y sus diferentes categorías.
Estabilidad	Alta.
Datos	Nombre, número máximo de equipos, imagen y descripción.

Cuadro 4.39: Deporte.

4.3. Diagrama de clases

En este apartado se describen las clases que se deberán utilizar independientemente del lenguaje que utilizaremos en la implementación. En la siguiente figura, podemos ver el diagrama de clases realizado en el análisis:

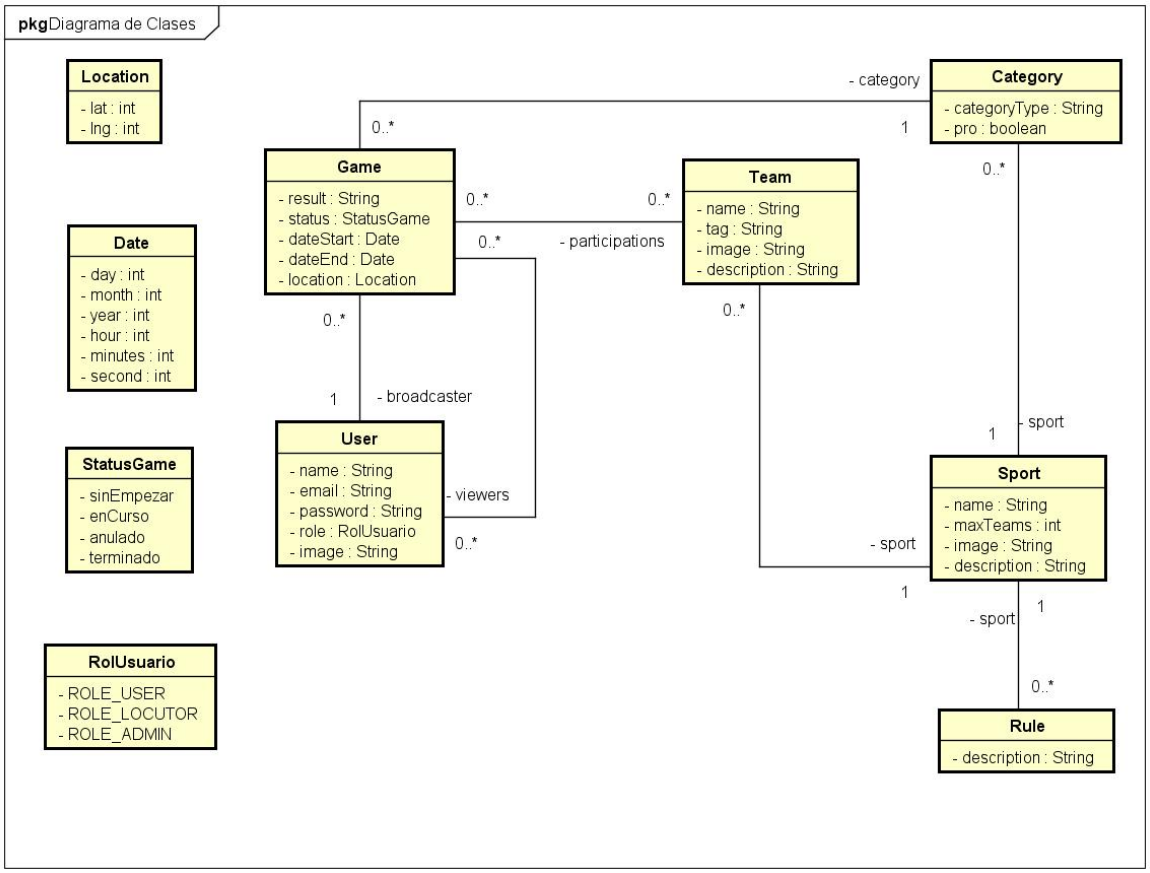


Figura 4.2: Diagrama de clases realizado en el análisis.

User representa los datos de acceso al sistema de cada usuario. Un usuario podrá obtener estas credenciales de forma única una vez registrado en el sistema.

Game representa es la mas importante ya que en esta aplicación se basa en partidos, esta relacionada con el usuario ya que hay un usuario que se encarga de transmitir el partido que sera el broadcaster y ademas se vuelve a relacionar con el para indicar que un partido puede tener cero o muchos espectadores. También se relaciona con la categoría ya que un partido es de una categoría específica.

Sport también es una de las mas importantes ya que cada partido tiene un deporte asociado a través de su categoría. Representa los datos de un deporte en el sistema así como el número máximo de equipos que permite ese partido.

Category representa la categoría de un deporte y si es profesional o no. Esta relacionado con Sport dado que una categoría siempre pertenece a un deporte.

Rule representa una regla específica de un deporte, por eso esta relacionada con un deporte y además, un deporte puede tener varias reglas.

Team representa los datos de un equipo en el sistema. Esta relacionado con un deporte, para que esos equipos solo puedan participar en partidos que tengan una categoría para ese deporte.

StatusGame representa el estado en el que esta un partido en ese momento.

RolUsuario representa el rol de un usuario dentro de la aplicación. Cada tipo de usuario puede acceder a diferentes funcionalidades dependiendo del valor de esta entidad.

Date representa el día y la hora.

Location representa la latitud y la longitud, es decir coordenadas.

4.4. Diagramas de secuencia

En este apartado se describen las secuencias de algunos de los casos de uso:

Cualquier usuario

Como indica la Figura 4.3 cualquier usuario puede identificarse y hasta que no introduzca un email y una contraseña que correspondan con un usuario no podrán realizar mas acciones.

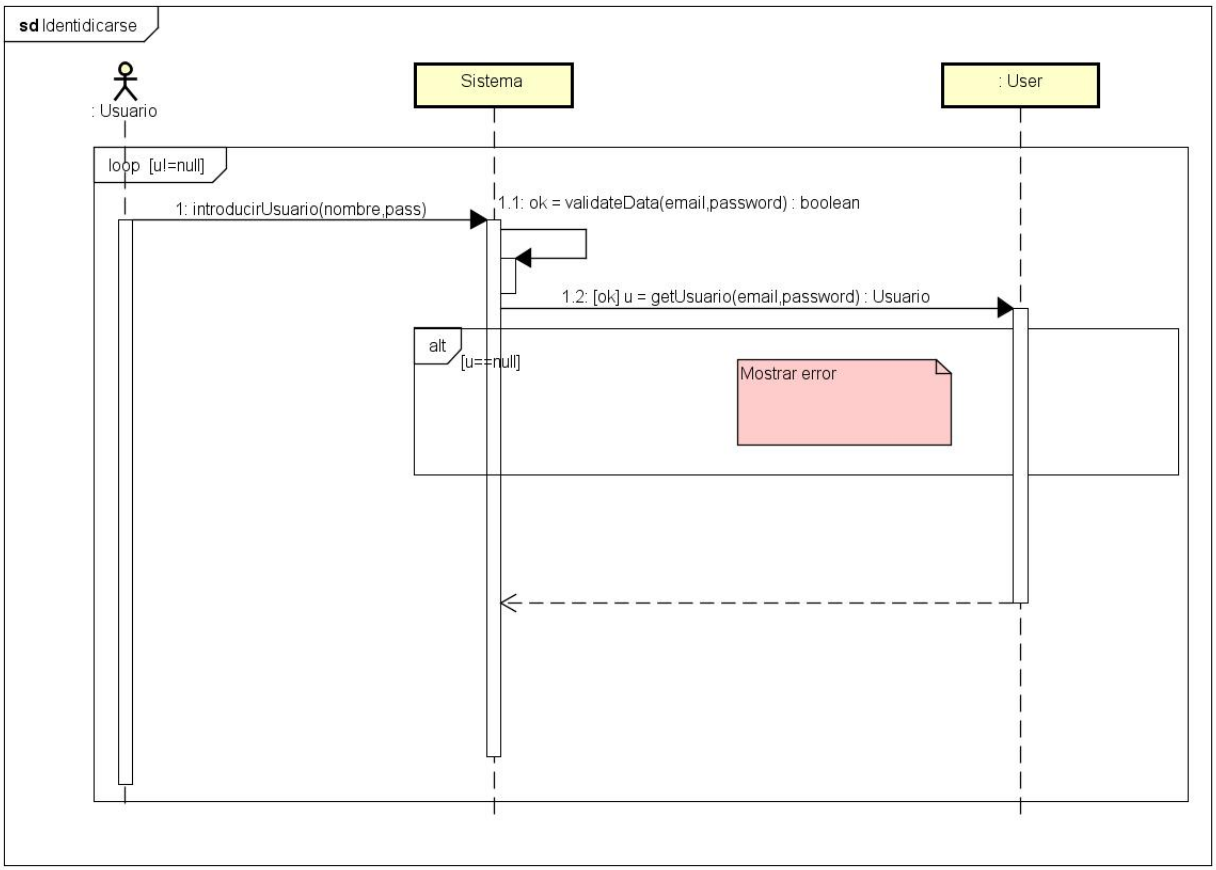


Figura 4.3: Diagrama de secuencia del CU identificarse.

Cualquier usuario puede presenciar un partido.

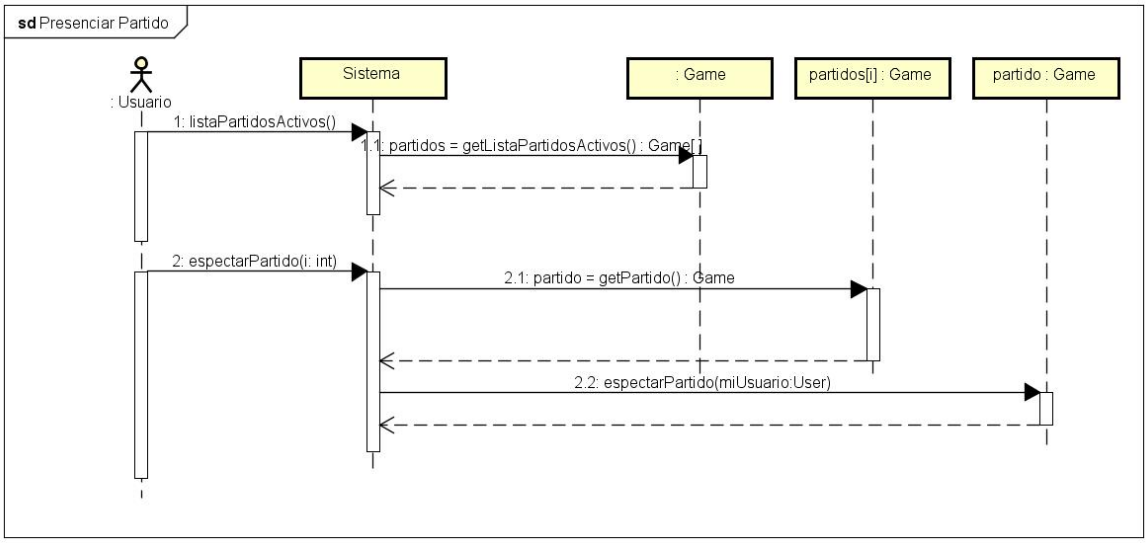


Figura 4.4: Diagrama de secuencia presenciar partido.

Cualquier usuario puede consultar los datos del partido que esta presenciando. Antes el usuario debe de haber presenciado un partido.

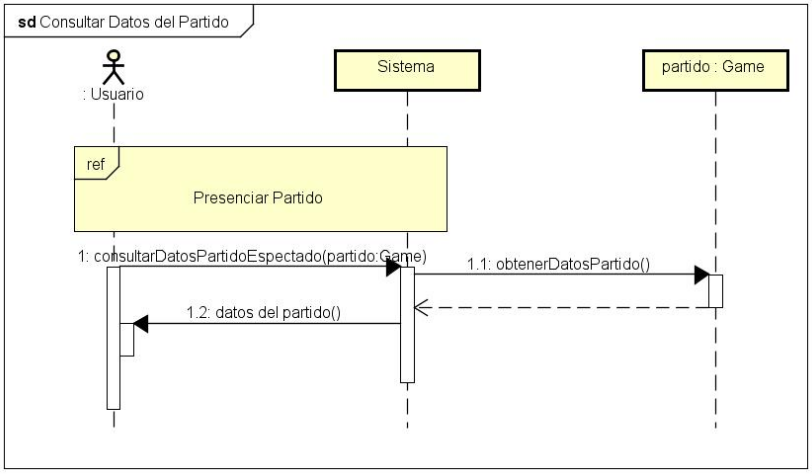


Figura 4.5: Diagrama de secuencia consultar datos de un partido.

Cualquier usuario puede consultar las reglas de un deporte de un partido que estén presenciando. Antes el usuario debe de haber presenciado un partido.

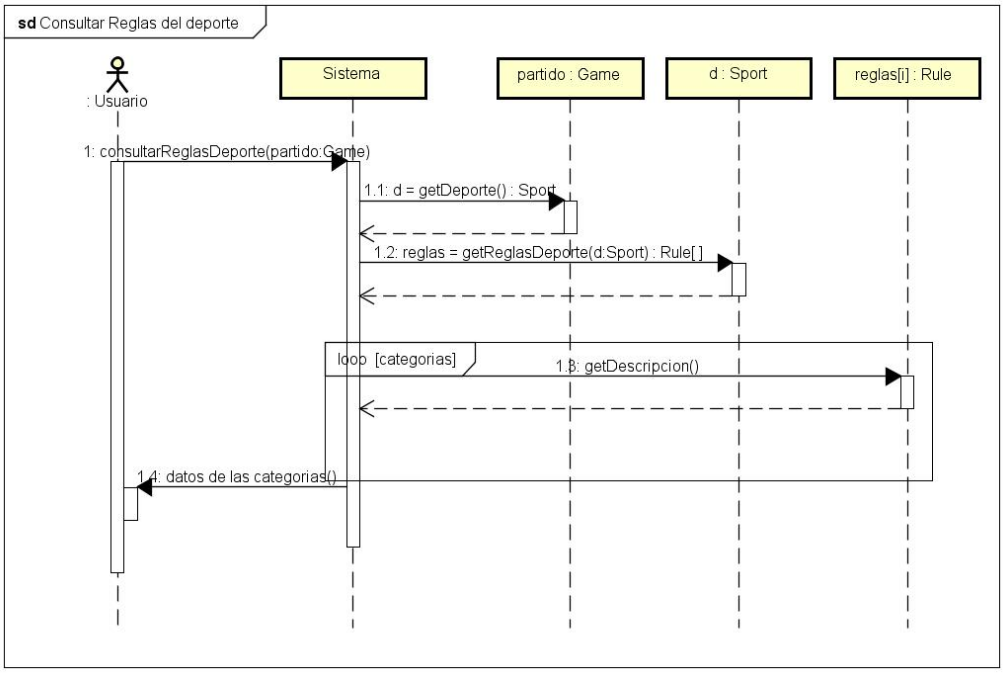


Figura 4.6: Diagrama de secuencia consultar reglas de un deporte de un partido que se este espectando.

Usuario administrador

Es necesario haberse registrado en el sistema y ademas tener el rol de administrador para poder introducir el deporte.

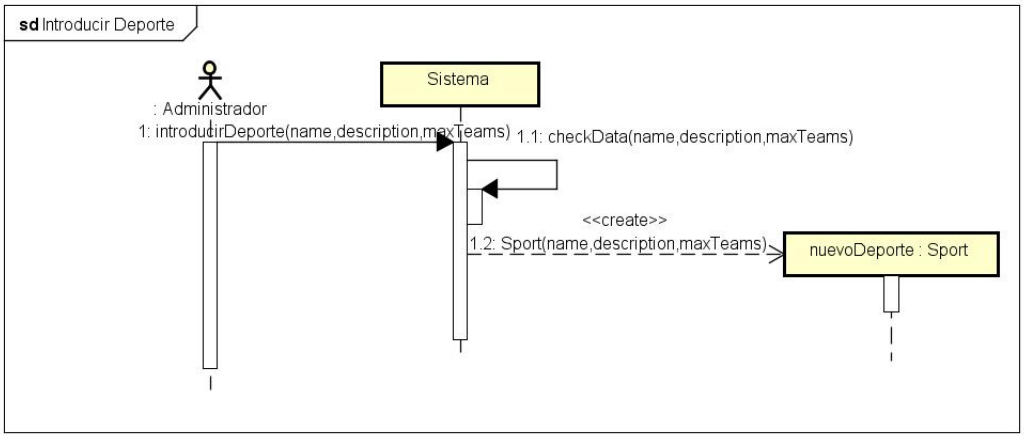


Figura 4.7: Diagrama de secuencia del CU introducir deporte.

Es necesario haberse registrado en el sistema y ademas tener el rol de administrador

para poder modificar el deporte.

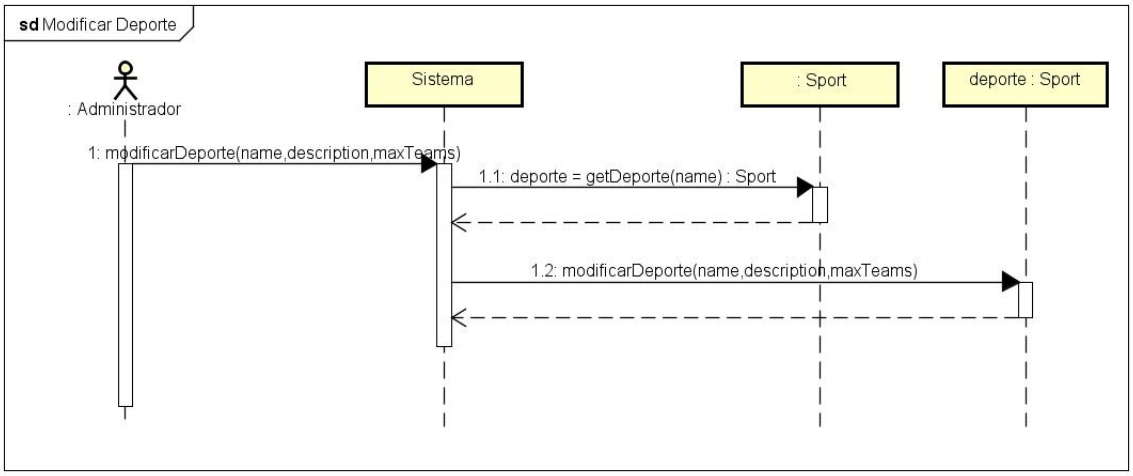


Figura 4.8: Diagrama de secuencia del CU modificar deporte.

Es necesario haberse registrado en el sistema y ademas tener el rol de administrador para poder eliminar el deporte. Cuando se elimina el deporte primero se deben eliminar las categorías y las reglas asociadas a ese deporte para no dejar inconsistencia de datos.

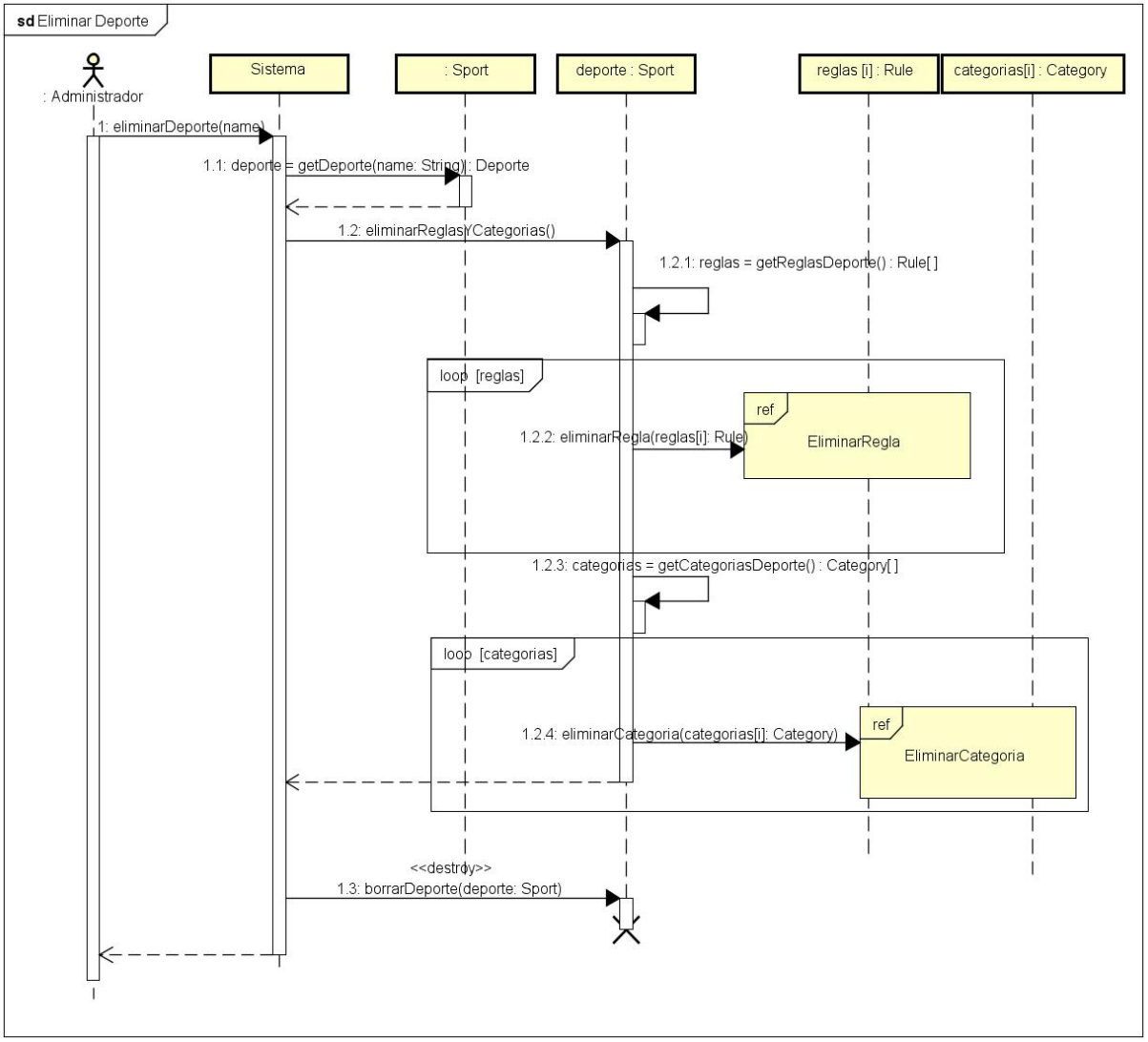


Figura 4.9: Diagrama de secuencia del CU eliminar deporte.

Usuario Locutor

Es necesario haberse registrado en el sistema y tener el rol locutor para poder crear un partido.

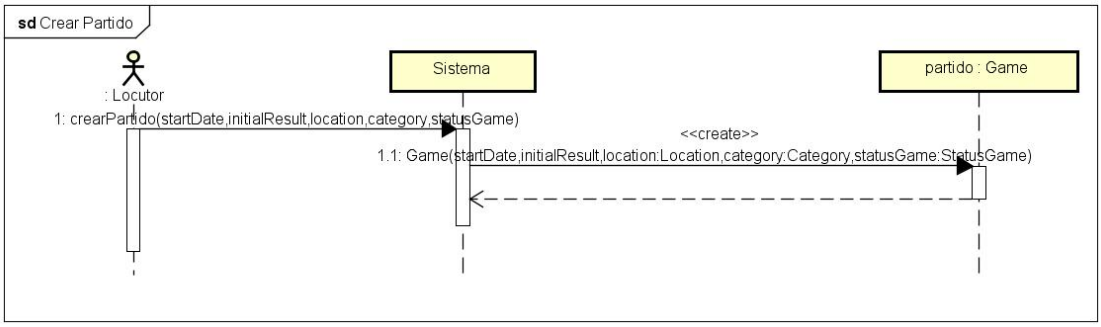


Figura 4.10: Diagrama de secuencia crear partido.

Para cambiar el resultado de un partido es necesario tener un partido activo.

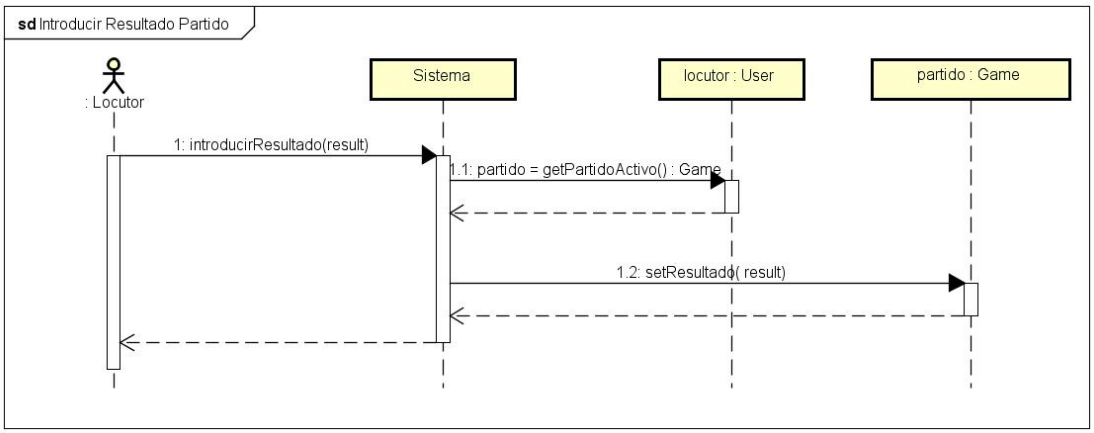


Figura 4.11: Diagrama de secuencia cambiar resultado de un partido.

Para introducir un equipo a un partido es necesario tener un partido activo.

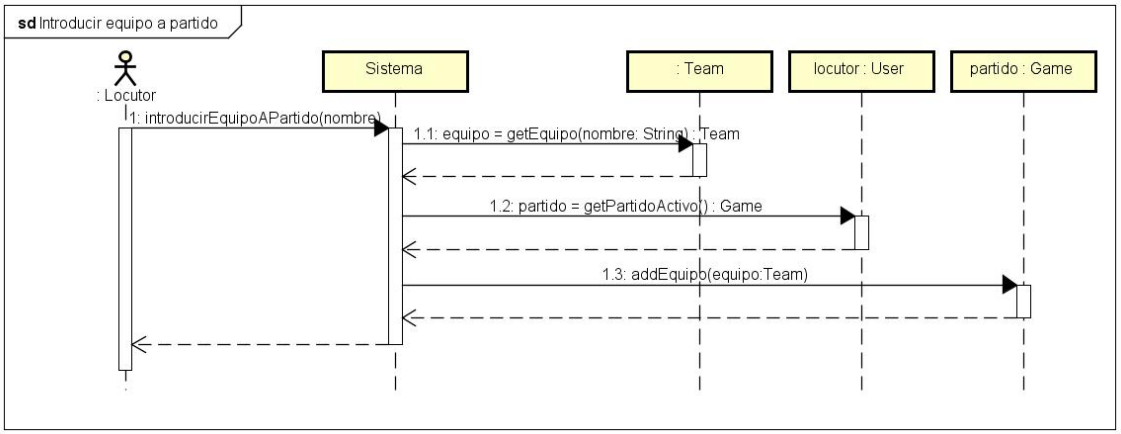


Figura 4.12: Diagrama de secuencia introducir equipo a partido.

Para eliminar un partido es necesario tener un partido activo ya que será este el que borraremos.

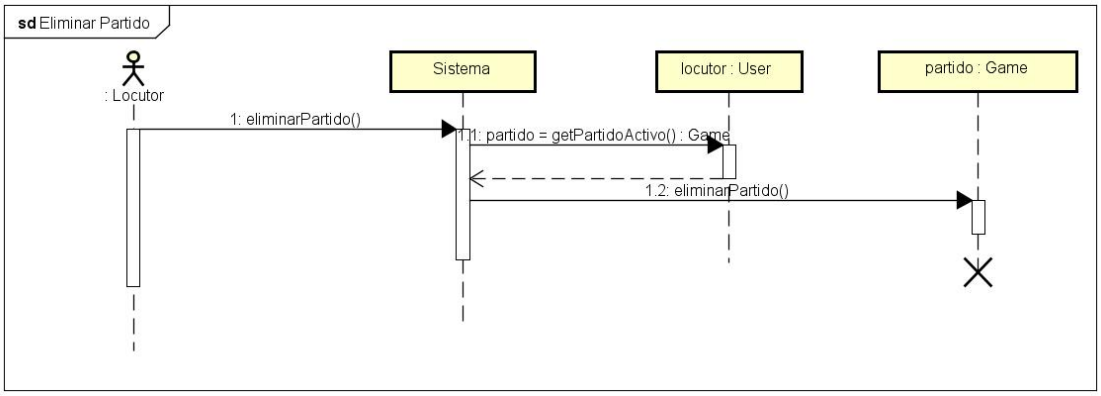


Figura 4.13: Diagrama de secuencia eliminar partido.

Capítulo 5

Diseño

5.1. Diseño de datos

En primer lugar podemos ver el modelo de datos que se usará para diseñar la base de datos como podemos ver en la Figura 5.1

User representa los datos de acceso al sistema de cada usuario, un usuario podrá obtener estas credenciales de forma única una vez registrado en el sistema, con los cuales podrá abrir sesión en la aplicación web de Deporty y navegar en el sistema.

IdAuthorized representa los datos que asocian a un usuario de la aplicación web con el usuario de Telegram y ademas es el encargado de permitir que el bot pueda mandar mensajes desde la aplicación web si el usuario lo ha autorizado. Se relaciona con los datos del usuario y solo habrá un registro de IdAuthorized por cada usuario como mucho.

Game representa es la mas importante ya que en esta aplicación se basa en partidos, esta relacionada con el usuario ya que hay un usuario que se encarga de transmitir el partido que sera el broadcaster. También se relaciona con la categoría ya que un partido es de una categoría específica.

Spectacle representa la expectación por parte de un usuario de Telegram a través de su id con un partido para poder ver los cambios desde Telegram cuando los datos del partido cambien.

Sport también es una de las mas importantes ya que cada partido tiene un deporte asociado a través de su categoría. Representa los datos de un deporte en el sistema así como el número máximo de equipos que permite ese partido.

Category representa la categoría de un deporte y si es profesional o no. Esta relacionado con Sport dado que una categoría siempre pertenece a un deporte.

Rule representa una regla específica de un deporte, por eso esta relacionada con un deporte.

Team representa los datos de un equipo en el sistema. Esta relacionado con un deporte, para que esos equipos solo puedan participar en partidos que tengan una categoría para ese deporte.

Participation representa una participación de un equipo a un partido, incluyendo el número de la participación del equipo para que en caso de ser importante se pueda conocer

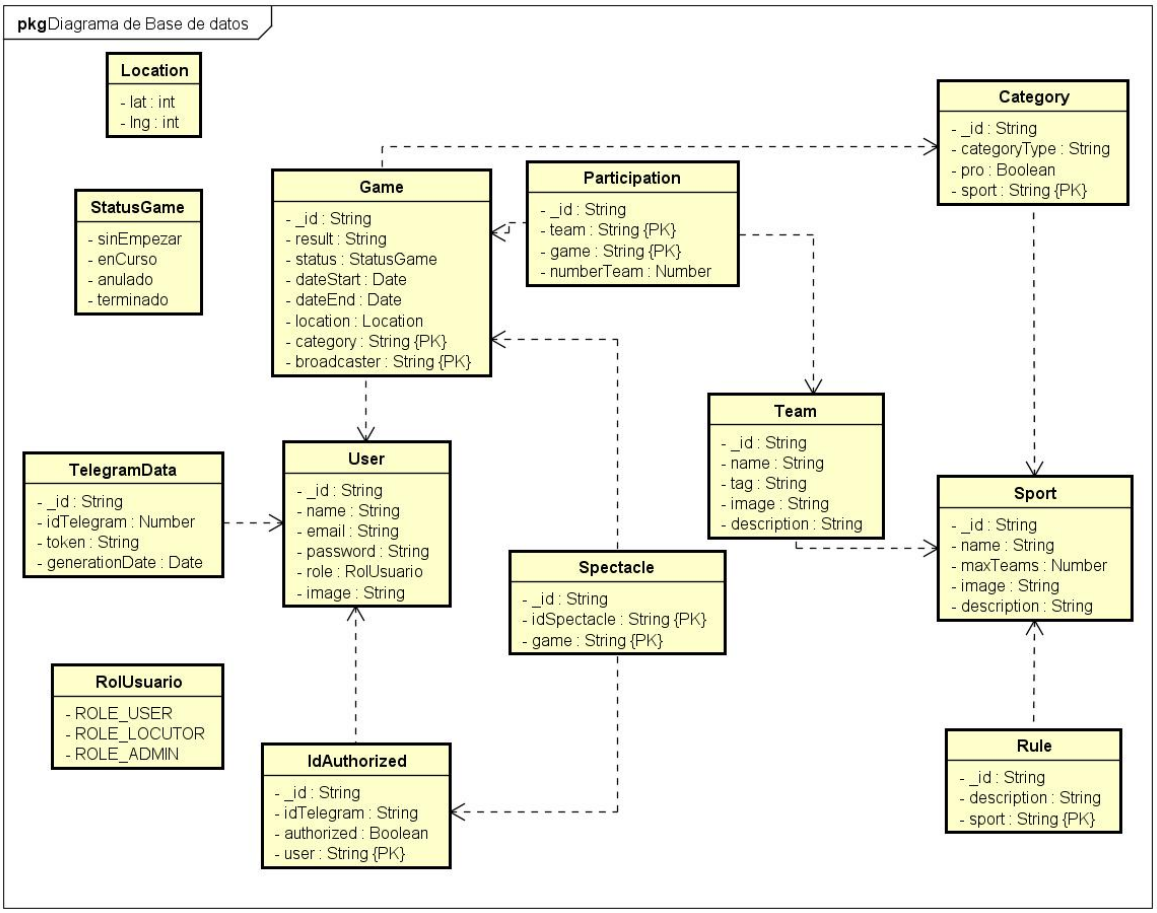


Figura 5.1: Modelo entidad-relación.

quien es local y quien es visitante.

StatusGame representa el estado en el que esta un partido en ese momento.

RolUsuario representa el rol de un usuario dentro de la aplicación. Cada tipo de usuario puede acceder a diferentes funcionalidades dependiendo del valor de esta entidad.

TelegramData representa la generación del token. Esta entidad esta formada por el dia en la que se generó para que se pueda comprobar si ha caducado, el token, el id de Telegram al que pertenece dicho token y el usuario que lo ha generado en la aplicación.

5.2. Diagramas de clase

En este apartado se describen las clases que se deberán utilizar para la implementación de forma mas detallada. En la siguiente figura, podemos ver el diagrama de clases realizado en el

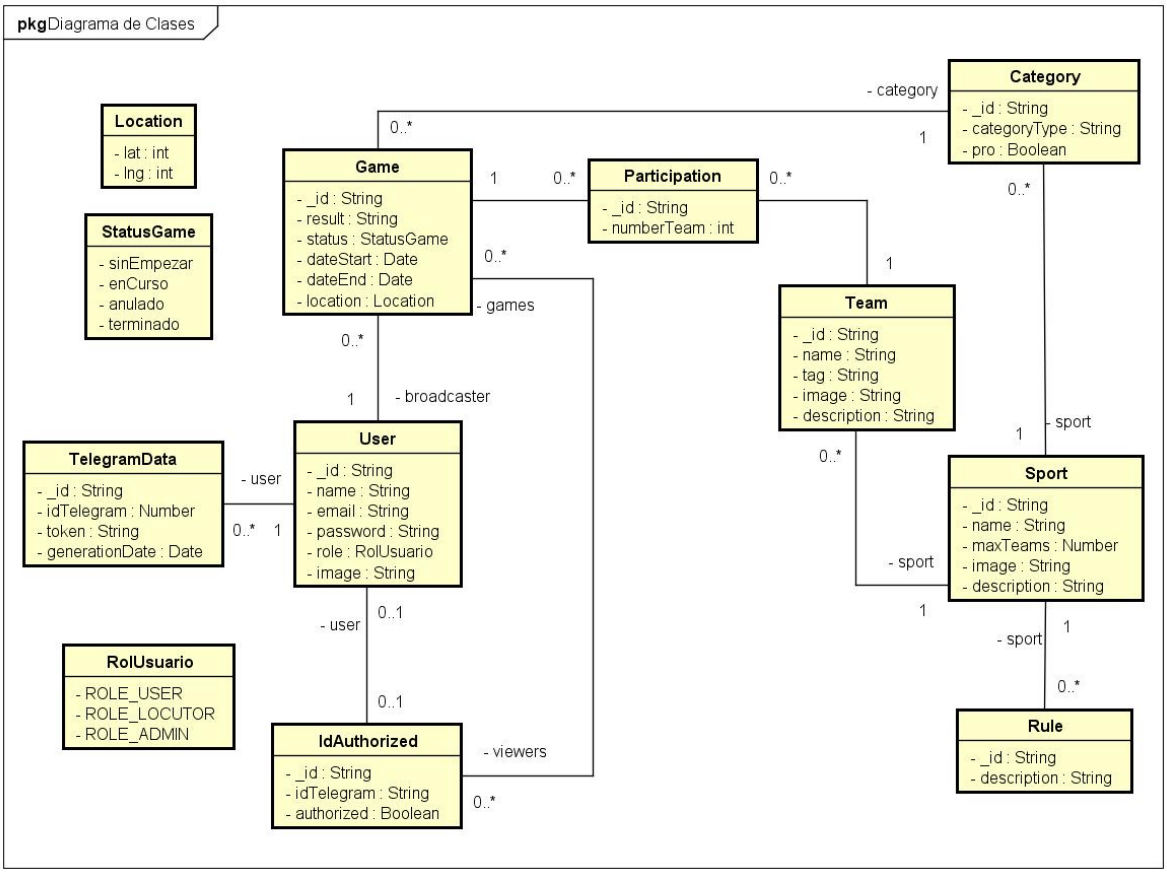


Figura 5.2: Diagrama de clases realizado en el diseño.

User representa los datos de acceso al sistema de cada usuario, un usuario podrá obtener estas credenciales de forma única una vez registrado en el sistema, con los cuales podrá abrir sesión en la aplicación web de Deporty y navegar en el sistema.

IdAuthorized representa los datos que asocian a un usuario de la aplicación web con el usuario de Telegram y ademas es el encargado de permitir que el bot pueda mandar mensajes desde la aplicación web si el usuario lo ha autorizado. Se relaciona con los datos del usuario y solo habrá un registro de IdAuthorized por cada usuario como mucho.

Game representa es la mas importante ya que en esta aplicación se basa en partidos, esta relacionada con el usuario ya que hay un usuario que se encarga de transmitir el partido que sera el broadcaster. También se relaciona con la categoría ya que un partido es de una categoría específica. Para el tipo de dato Date se usará el que tiene javascript. Tambien se relaciona con IdAuthorized porque no será necesario tener un usuario para presenciar un partido, pero si hará falta tener idTelegram, por lo que quien presenciará el partido sera esta clase.

Sport también es una de las mas importantes ya que cada partido tiene un deporte asociado a través de su categoría. Representa los datos de un deporte en el sistema así como el número máximo de equipos que permite ese partido.

Category representa la categoría de un deporte y si es profesional o no. Esta relacionado con Sport dado que una categoría siempre pertenece a un deporte.

Rule representa una regla específica de un deporte, por eso esta relacionada con un deporte y además, un deporte puede tener varias reglas.

Team representa los datos de un equipo en el sistema. Esta relacionado con un deporte, para que esos equipos solo puedan participar en partidos que tengan una categoría para ese deporte.

Participation representa una participación de un equipo a un partido, incluyendo el número de la participación del equipo para que en caso de ser importante se pueda conocer quien es local y quien es visitante.

StatusGame representa el estado en el que esta un partido en ese momento.

RolUsuario representa el rol de un usuario dentro de la aplicación. Cada tipo de usuario puede acceder a diferentes funcionalidades dependiendo del valor de esta entidad.

TelegramData representa la generación del token. Este token es individual para cada usuario en la aplicación, es decir, ningún otro usuario puede enlazar las cuentas con ese token ya que lo dará por inválido.

5.3. Herramientas utilizadas

En esta sección se entrará más en detalle en las tecnologías especiíficas que se han empleado para desarrollar la aplicación: lenguajes de programación empleados, gestor de bases de datos, herramientas usadas en la planficación y en la generación de documentación.

Se utilizará la tecnología llamada MEAN Stack, que se basa en utilizar Angular en el front-end y en el back-end se utilizará *Express* y *Node.JS*. Como gestor de base de datos se utilizará mongoDB.

El principal motivo de la elección de MEAN Stack para hacer la aplicación web y de Node.Js para hacer el bot de Telegram es que se puede incluir el bot dentro del Back-end de la aplicación lo que facilita su integración así como la comunicación de ambos.

MEAN Stack tiene varias ventajas, las mas importantes son:

1. Permite hacer la aplicación web de principio a fin utilizando JavaScript
2. MongoDB proporciona escalabilidad y no requiere de potentes recursos para trabajar por lo que se pueden ejecutar en un hardware de bajo precio.
3. Express permite el uso de middlewares en las peticiones HTTP
4. El desarrollo en Node.Js es más rápido.
5. Las aplicaciones son mucho mas rápidas, lo que implica que la experiencia de usuario es mejor.

6. Al igual que MongoDB, Node.js necesita de menor coste en el hardware ya que necesita menos memoria para soportar las mismas conexiones que otras tecnologías actuales.

7. Es bastante flexible.

5.3.1. Front-End

Debido a que Deporty tendrá una aplicación web se necesitará una interfaz web. Las tecnologías que se utilizarán para esto serán: HTML5, CSS, TypeScript, Angular, Bootstrap, Jquery.

- **HTML5** Es una nueva versión del lenguaje HTML, es un lenguaje de programación que se utiliza para el desarrollo de páginas web.

- **CSS** Es un lenguaje de diseño gráfico para definir y crear el estilo de las páginas web.

- **TypeScript** Es un lenguaje de programación de código abierto con herramientas de POO. Se le llama comunmente el superset de JavaScript de tal forma que el navegador nunca sabrá si estamos utilizando JavaScript o este lenguaje

- **Angular** Es un framework de Javascript que se basa en el patrón modelo vista controlador y se utilizará de la lógica de la aplicación.

- **Bootstrap** Es una biblioteca o conjunto de herramientas para el diseño de sitios web y se utilizará para diseñar las distintas páginas.

- **JQuery** Es una biblioteca de JavaScript que permite interactuar de manera simplificada con los Documentos HTML. En este caso solo se utilizará porque es necesario para poder usar Bootstrap

5.3.2. Back-End

Dentro del back-end necesitaremos varias herramientas dado que a parte de hacer la parte del servidor web tendremos que hacer también la del bot. Las herramientas que se utilizarán serán: JavaScript, Express, Node.JS, node-telegram-bot-api y otras librerías para diversas cosas.

- **JavaScript** es un lenguaje de programación que se utilizará en la totalidad del back-end.

- **Node.js** Es un framework de código abierto que permite la ejecución de programas hechos en Javascript en el lado del servidor. Por lo tanto, permite que tanto el servidor como los clientes se comuniquen por medio de JS de forma rápida.

- **Express** Es un framework de aplicaciones web, flexible escrito en JS para Node.js y servicios Rest. Es decir es el que se encarga de recibir las peticiones del cliente y de responder a estas.

- **node-telegram-bot-api** Es una api que permite la comunicación de Telegram con Node.js y proporciona varios métodos para ello.

Hay otras librerias que se utilizarán en el back-end. Por ejemplo un cifrador de contraseñas, un generador de tokens para la sesión o mongoose para la conexión para la base datos.

5.4. Arquitectura

Los patrones de diseño son herramientas de modelado para solucionar problemas de diseño de software que se repiten. Su finalidad es permitirnos afrontar problemas de diseño encajando nuestro proyecto en un modelo. En esta aplicación se utilizó una arquitectura cliente-servidor donde los clientes son las computadoras de los diferentes usuarios y los servidores son otras computadoras que proveen los datos.

La Figura 5.3 representa la estructura del Stack de programación, es decir como va el flujo en MEAN Stack.

Las partes de las que se compone este tipo de sistemas por lo tanto el cliente que es el encargado de enviar peticiones al servidor y espera a que le llegue una respuesta y el servidor que a través de diferentes puertos responde a esas peticiones, es decir provee un servicio.

Este es el esquema de MEAN Stack. Se aprecian tres capas, la primera es la que se utilizará desde el navegador o desde el bot de telegram que será la capa del cliente. La segunda capa será la del servidor web que recibirá las llamadas de dichos clientes, conectara con la tercera capa, que es el servidor de la base de datos y responderá al cliente después de hacer las operaciones pertinentes.

El funcionamiento será el siguiente: El cliente le mostrará al usuario la primera vista, cuando el usuario decide que hacer esos datos serán recogidos por la interfaz, serán preparados y enviados al servidor web mediante un protocolo HTTP. Una vez obtenidos estos datos, el servidor web se encargará de realizar todas las operaciones asociadas a la acción del usuario, ya sea insertar, modificar o consultar datos a través de una consulta al

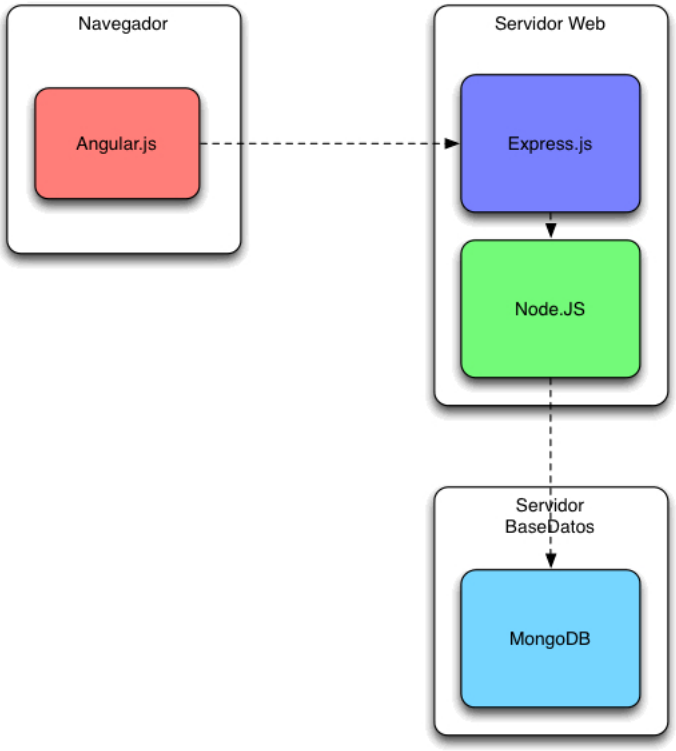


Figura 5.3: Estructura del Stack de programación.

servidor de la base de datos. Una vez realizadas las operaciones el servidor web devolverá una respuesta al cliente y este mostrará los datos necesarios para realizar completar esa petición.

En este caso se podía tener un servidor para la aplicación web y otro para el bot de Telegram. En caso de tener muchas peticiones por ambas partes y que estas interfieran en la velocidad de respuesta por parte del servidor se podrían separar estas dos funcionalidades en dos servidores. Para ahorrar costes y al no ser necesario por el momento se tomo la decisión de tener ambas partes en un único servidor. Incluso el front-end y el back-end se podrían tener en servidores separados dado que el back-end y el front-end se pueden ejecutar de forma independiente.

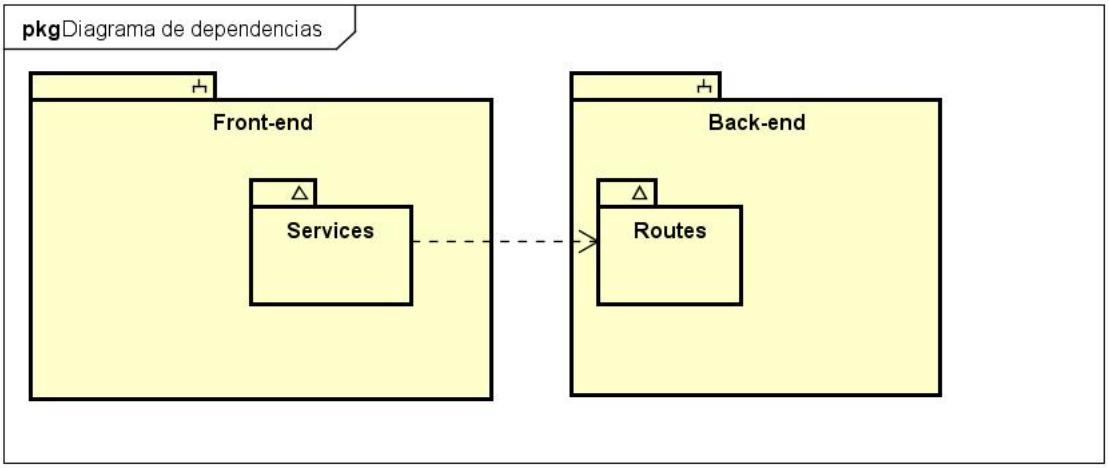


Figura 5.4: Diagrama de dependencias entre el front-end y el back-end.

Desde el paquete services del back-end se comunica con el front-end a traves del paquete routes. El paquete service tiene clases que su única función es conocer la ruta del back-end a la que quiere conectarse y lógicamente hacer esta conexión. El paquete routes del back-end tiene clases que lo único que hacen es abrir una ruta de conexión y enviar esa petición al controlador para que se pueda procesar dicha petición.

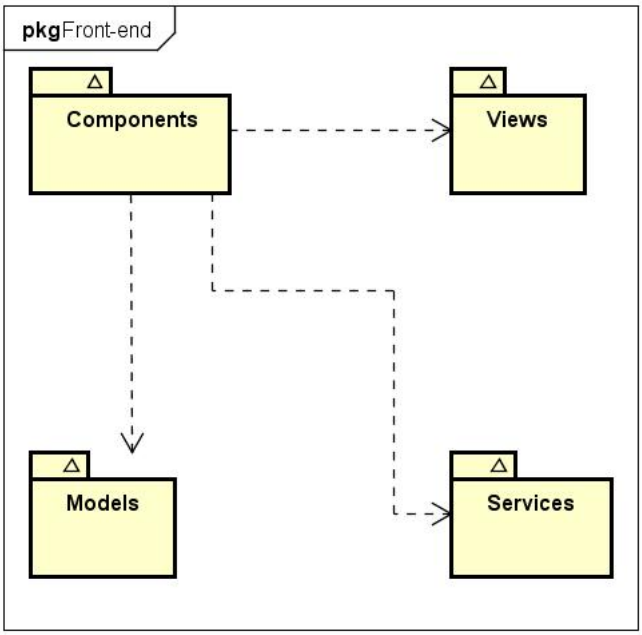


Figura 5.5: Diagrama de dependencias entre paquetes del front-end.

Como podemos ver en la figura 5.5, el front-end tiene varios paquetes, pero solo uno

de ellos usa el resto que es el paquete de componentes. En este paquete estarán los controladores de los CU, utilizará los modelos que necesiten para ello y se encargarán también de actualizar la vista de forma dinámica cuando llegue la respuesta del back-end a través del paquete de servicios.

Podemos ver como en el front-end están las capas de presentación que son las vistas que hay en el paquete views, y la capa de negocio, que tiene el modelo y los controladores. La capa de servicios es una capa separada de las otras mencionadas anteriormente.

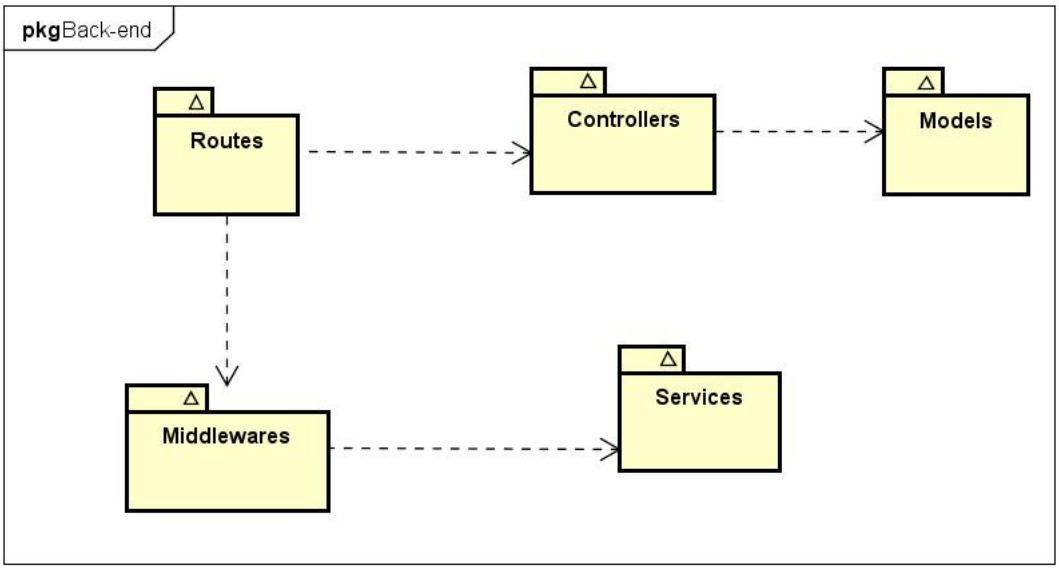


Figura 5.6: Diagrama de dependencias entre paquetes del back-end.

Como podemos ver en la figura 5.6, el back-end esta compuesto por varios paquetes, como ya se explicó anteriormente el paquete de rutas es el encargado de enviar las peticiones recibidas a los controladores y ademas puede utilizar middlewares en el proceso. Los controladores en este caso son los gestores encargados de obtener los datos a partir de las *entities* de la capa Models y si es necesario hacer operaciones con ellos y devolver una respuesta para que el front-end pueda conocer si ha habido un error, si se ha producido todo correctamente o para recibir los datos que se precisaban en la petición.
El back-end exceptuando los middlewares y el paquete de servicios se correspondería con la capa de acceso a datos.

Con esta arquitectura hemos conseguido separar la aplicación en varias capas cuya responsabilidad está bien definida. La capa de presentación será la encargada de recibir las peticiones del usuario así como mostrar los datos para que pueda verlos. La capa de negocio será la encarga de presentar la información a las vista, interactuar con el back-end de la aplicación mediante servicios y capturar eventos que puedan ser gestionados dentro del flujo de la aplicación sin necesidad de contactar con el back-end.

5.5. Diagramas de secuencia

En esta sección se mostrarán algunos de los diagramas de secuencia de algunos casos de uso. Se mostrará en caso de ser necesario el diagrama del CU primero el front-end y despues se mostrará el diagrama de seciencia del back-end.

CU Identificarse

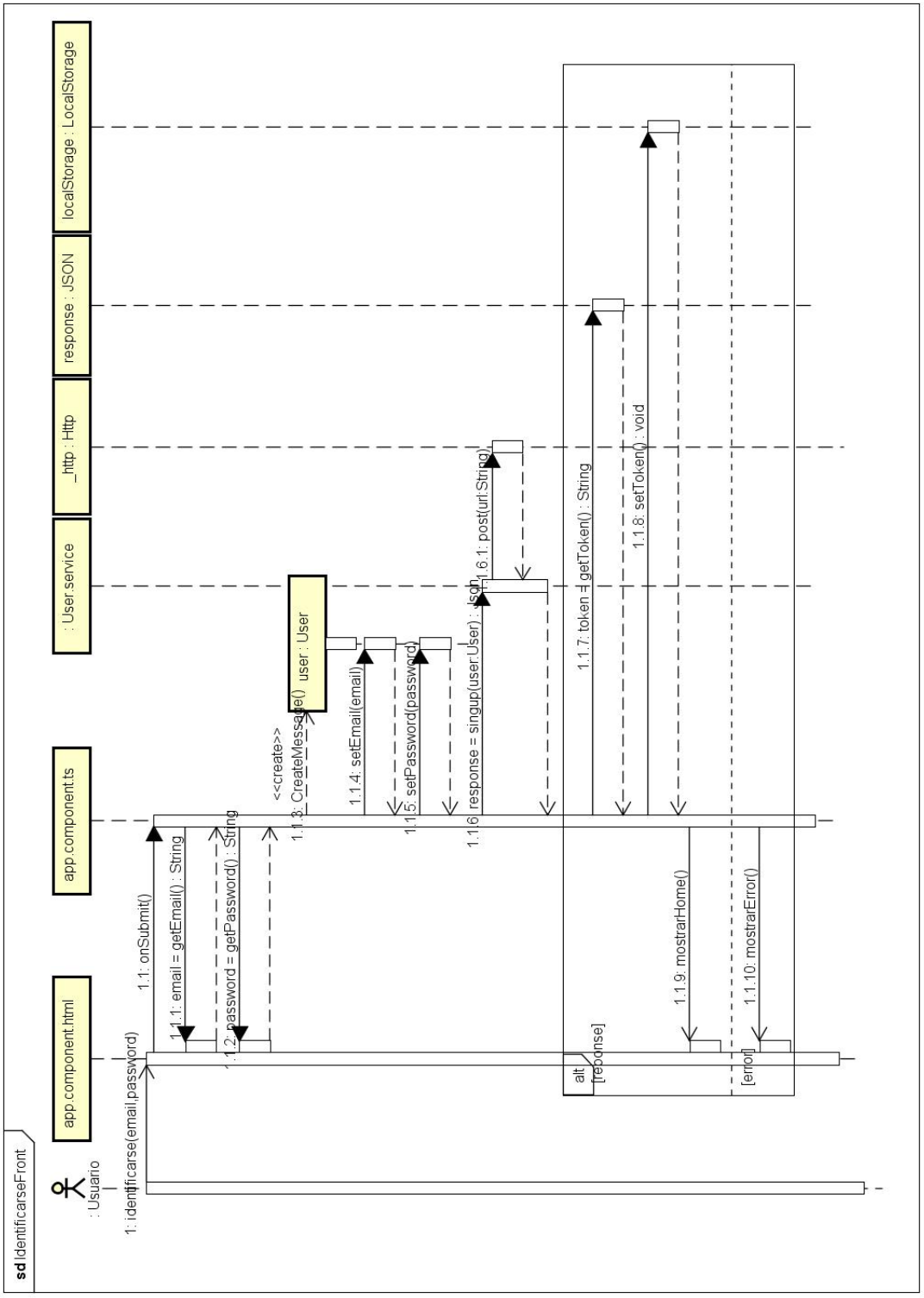


Figura 5.7: Diagrama de secuencia del CU Identificarse Front-end.

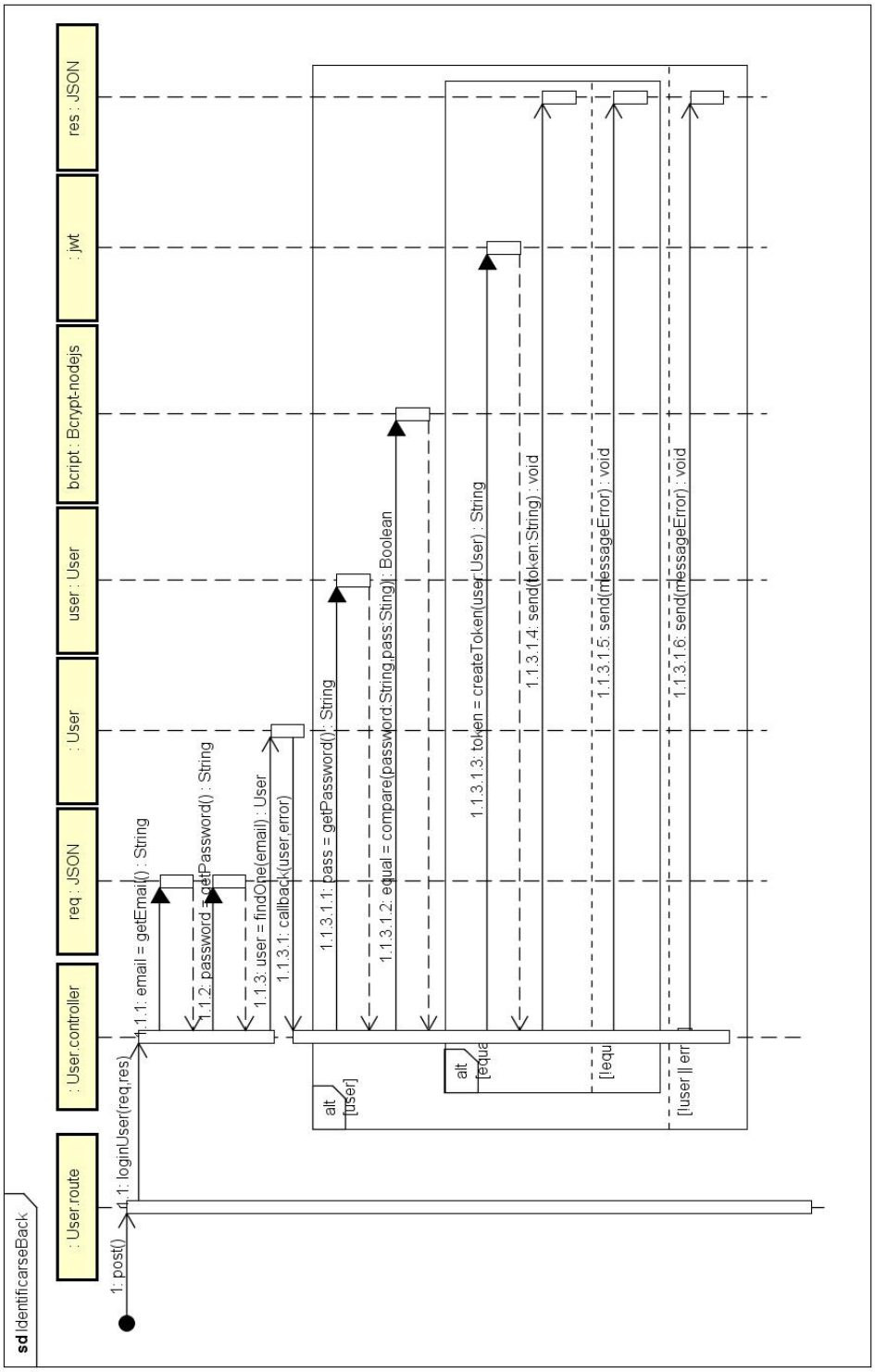


Figura 5.8: Diagrama de secuencia del CU Identificarse Back-end.

El localstorage utilizado en el front-end se utiliza para guardar objetos como en una sesión del usuario aunque, los objetos a diferencia de los objetos de sesión, no tienen fecha de espiración, ya que los datos persisten solo en la ventana en la que se creó y no se eliminan cuando se cierra dicha sesión.

En el back-end se utilizan diferentes frameworks y librerias como bcrypt, que se encarga tanto de cifrar una contraseña como de compararla con un string para comprobar que son la misma. JWT se encarga de crear el token al iniciar sesión. Este token sirve para convertir los datos en este caso del usuario en una cadena de caracteres. La utilizaremos para que cada vez que se haga una acción se necesitará de este token para autentificar el usuario y permitirle o denegarle la acción.

CU Crear deporte

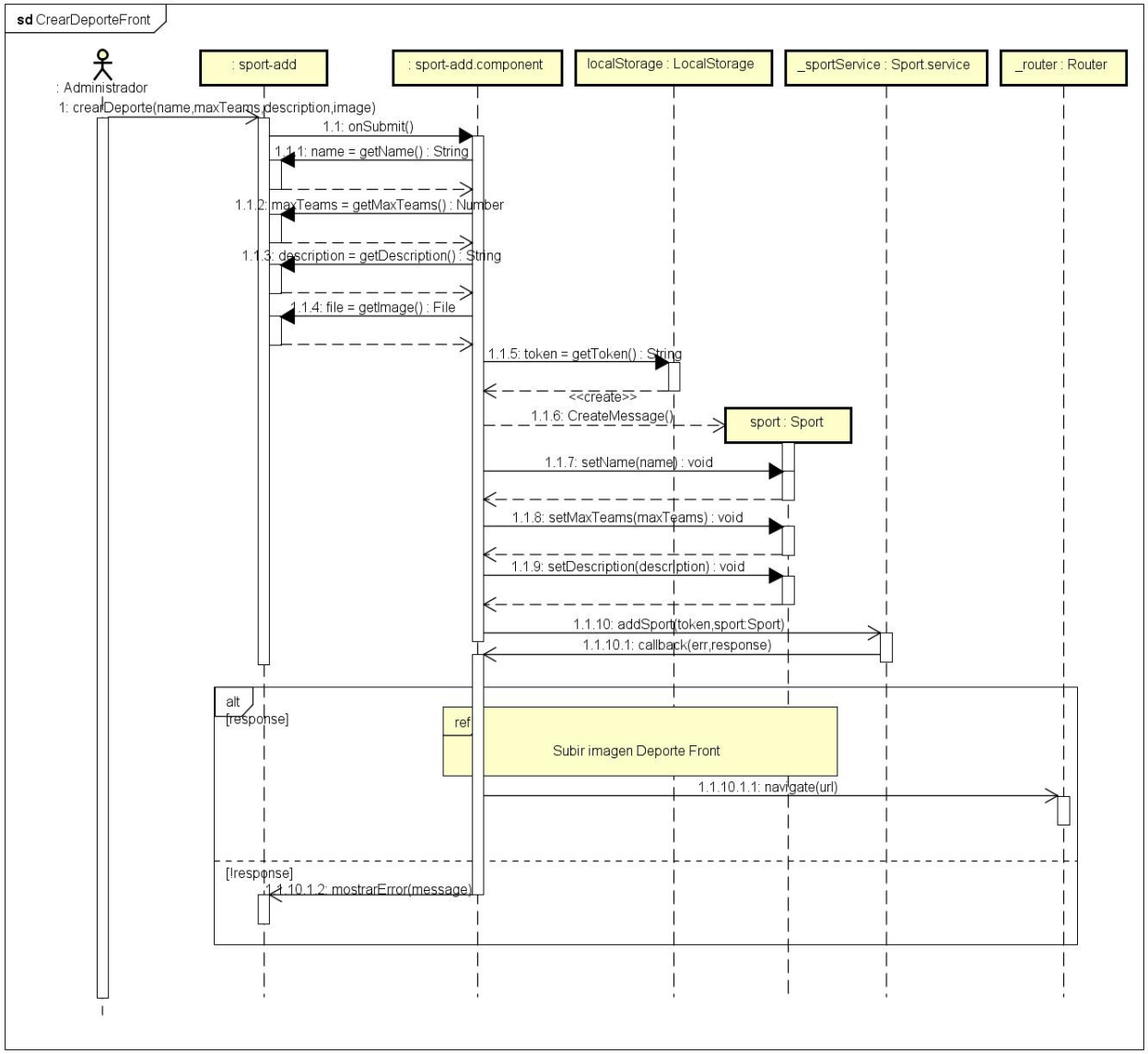


Figura 5.9: Diagrama de secuencia del CU Crear deporte Front-end.

Como vemos en el front-end es practicamente igual que identificarse, solo que no usa las bibliotecas que este usaba. De la vista vamos al componente que se encarga de realizar las operaciones, en este caso necesitamos el token para que la acción pueda ser realizada y la cogemos del localStorage previamente almacenado en el inicio de sesión. Despues contactaremos con el back-end mediante el servicio en este caso *Sport.service*. El router sirve para poder cambiar la vista a partir del url que le demos, en nuestro caso iremos a editar deporte.

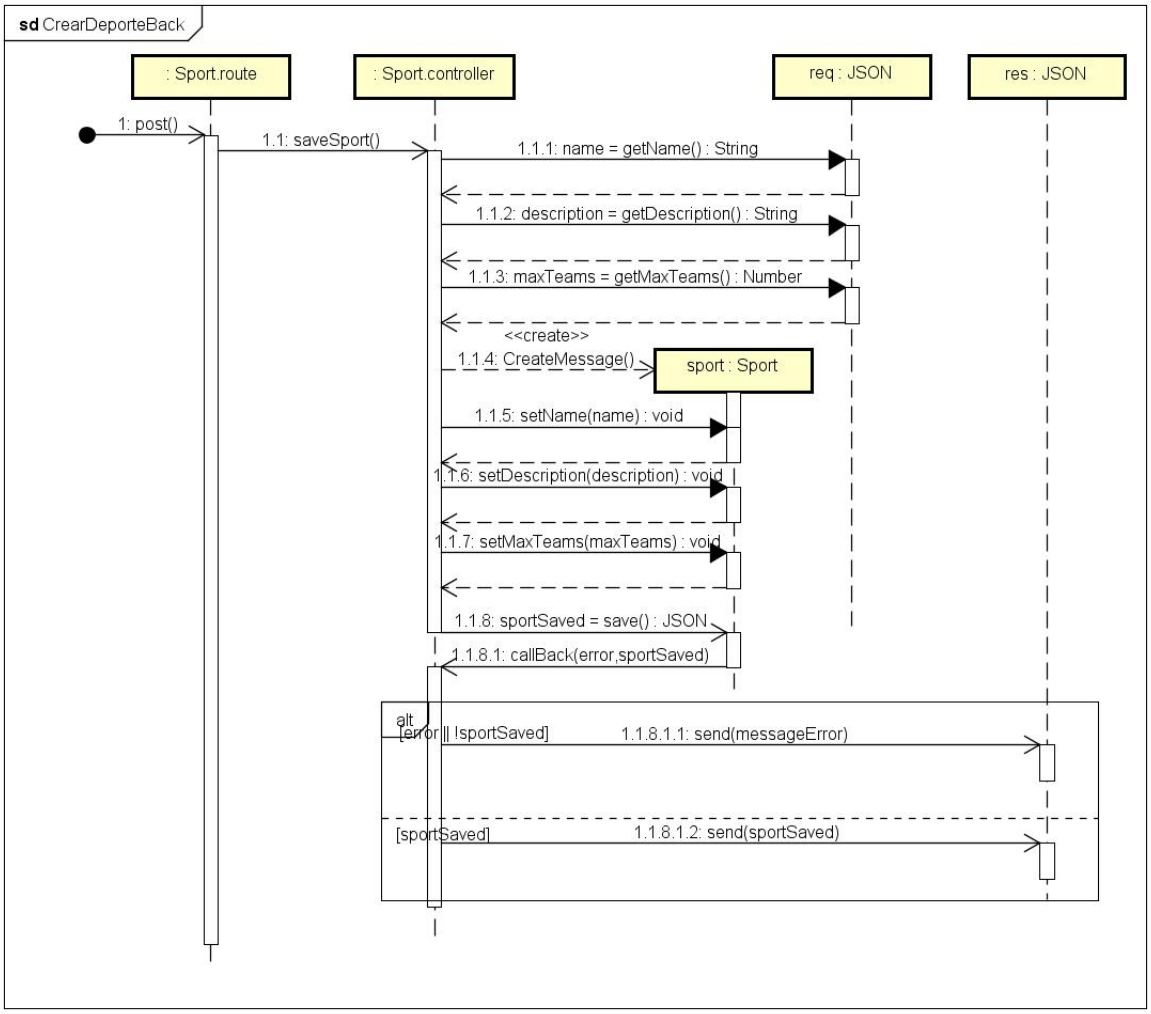


Figura 5.10: Diagrama de secuencia del CU Crear deporte Back-end.

En el back-end tambien es parecido ya que siempre es igual el flujo. De la ruta del reporte que es donde llega nuestra petición del front-end pasaremos al controlador en este caso del deporte. Este a partir de la *request* obtendrá los datos necesarios para crear el deporte. Una vez creado se enviará una respuesta al front en caso de que no haya deporte se devolverá un error y sino se devolverá el deporte que se ha guardado.

En caso de que haya que subir una imagen esta se subirá mediante otro servicio y otro caso de uso que es subir imagen.

Subir imagen de un deporte

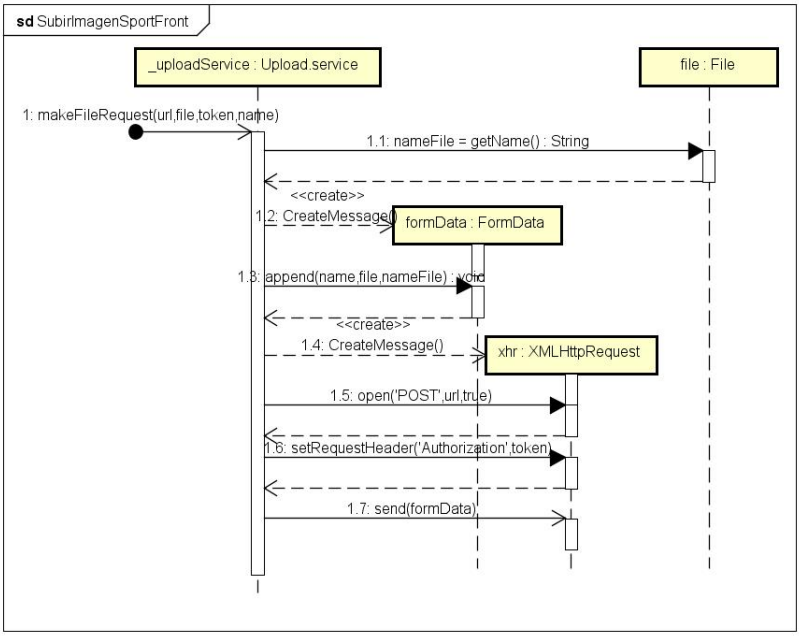


Figura 5.11: Diagrama de secuencia de la subida de una imagen de un deporte Front-end.

En este caso el front-end tiene que enviar el archivo al back-end mediante una petición XMLHttp. Primero tenemos que crear un formulario e introducir el archivo así como su nombre de archivo

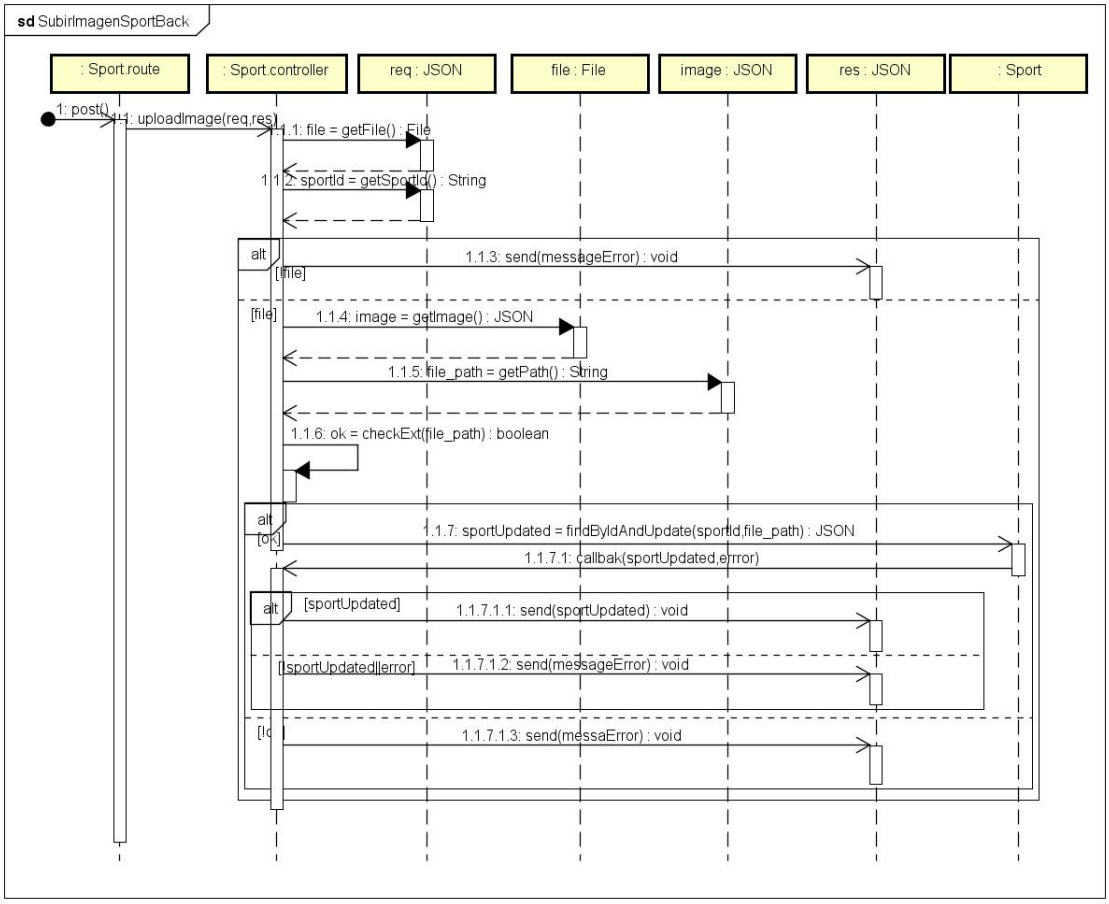


Figura 5.12: Diagrama de secuencia de la subida de una imagen de un deporte Back-end.

En el back-end como es habitual, primero llega a ruta del deporte que es donde recibe dicha petición, este utiliza el método del controlador que sea necesario. El controlador obtiene los parámetros de la petición en este caso solo obtendrá el fichero y el identificador del deporte. Comprueba que es una extensión valida antes de asignar esa imagen al deporte.

Para finalizar devuelve una respuesta dependiendo del resultado.

Consulta de un deporte

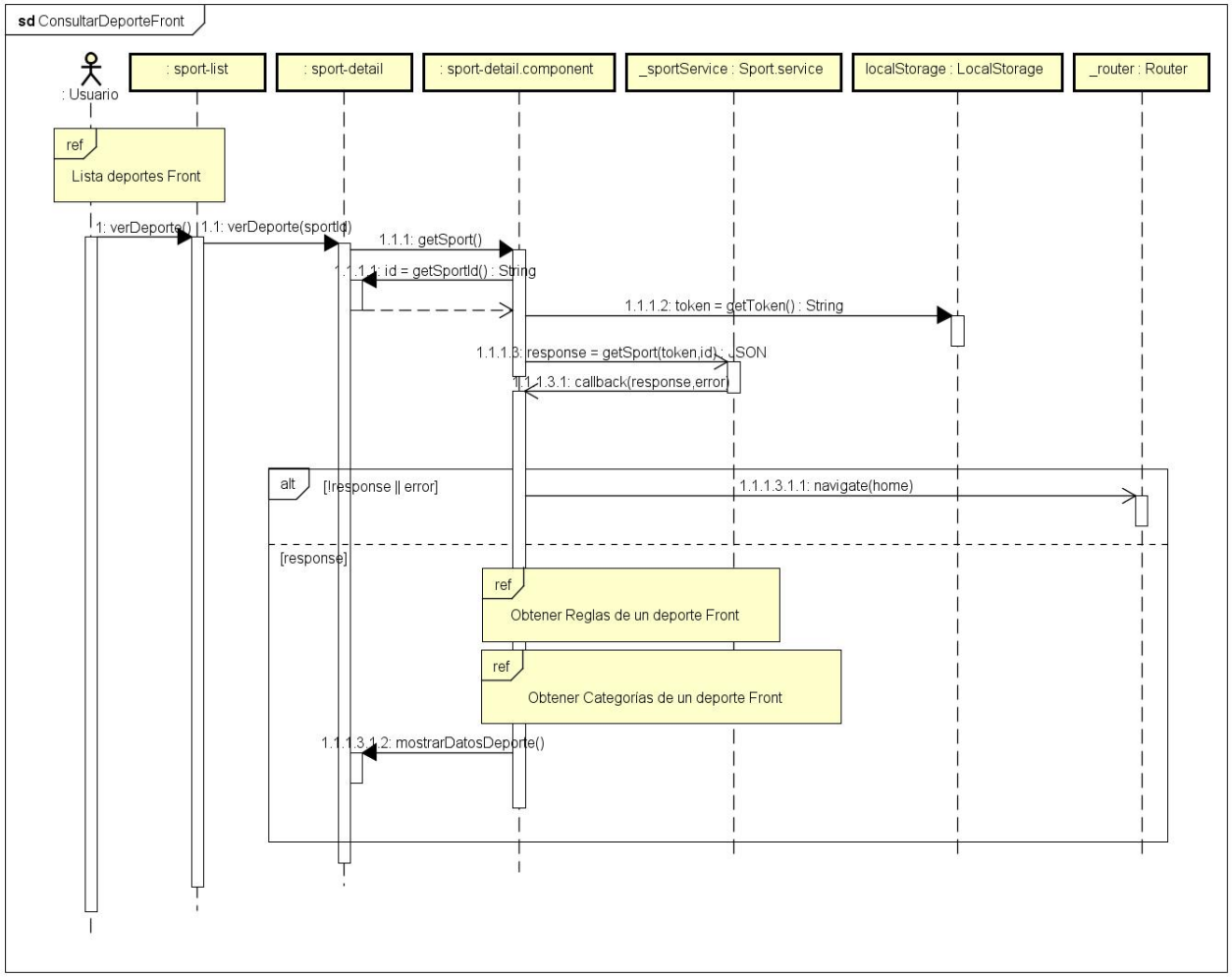


Figura 5.13: Diagrama de secuencia de la consulta de datos de un deporte Front-end.

En este caso primero obtendrá la lista de deportes desde su respectiva vista. Una vez hecho, podremos seleccionar el deporte e ir a su vista de detalle. Iremos del componente al servicio encargado de comunicarse con el back-end. En este caso si hay un error volveremos a la página home, si no lo hay entonces se mostrarán los datos del deporte.

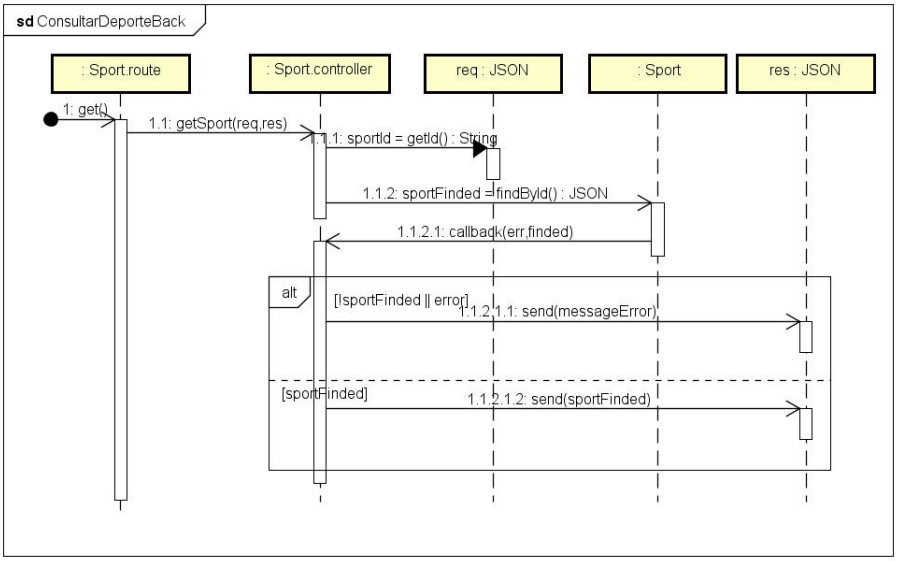


Figura 5.14: Diagrama de secuencia de la consulta de datos de un deporte Back-end.

En el back-end sigue el flujo común, de la ruta de deporte al controlador. El controlador buscará el deporte por el id y responderá dependiendo de si existe el deporte o no.

Capítulo 6

Implementación

Una vez acabado el diseño de la aplicación, se puede realizar la implementación. Como se explicó anteriormente, la implementación se hará siguiendo el patrón MVC, que dividirá la aplicación en tres niveles de absctracción (modelo, vista y controlador). Estos tres niveles se repartirán entre el front-end y el back-end de la aplicación. En el front-end tendremos la capa de la vista y del controlador y en el back-end tendremos la capa del modelo. En este capítulo se detallará como se ha llevado a cabo la implementación en cada una de las capas y como se comunica el front-end con el back-end.

6.1. Modelo

El modelo es el que realiza la comunicación entre el controlador y la base de datos. Como se dijo anteriormente se sitúa en el Back-end. Se compone de varias clases, los rutas controladores y las modelos o entities.
Las clases *routes* o rutas, son clases que sirven para establecer las conexiones y se pueda comunicar el front-end u otras aplicaciones a través de diferentes rutas y de diferentes metodos HTTP. La sintaxis es la siguiente:

```
var express = require('express');
var GameController = require('../controllers/game');

var api = express.Router();
var md_auth = require('../middlewares/authenticated');

api.post('/partido',md_auth.ensureAuth,GameController.
   saveGame);
```

Se utiliza express, y a traver del Router de express se pueden meter rutas, en este caso se utiliza un método post. El primer parámetro el la ruta que queremos que reciba el evento, la segunda es si necesita middleware, en este caso utilizamos el middleware para confirmar que el usuario esta autentificado en el sistema y el tercer parámetro es el

metodo al que tiene que llamar después de pasar por los diferentes middlewares.

Las clases controladoras se encargar de preparar la consulta y de ejecutarla a través de la entities, esta realiza una función de callback con los resultados de la ejecución del a consulta. Puede devolver un error o un objeto JSON con los diferentes datos de la consulta ejecutada. Una vez hecha la consulta enviará un mensaje en este caso al front-end para saber si se ha realizado la petición correctamente o por el contrario ha habido un error.

```
function saveGame(req,res){

    var token = req.headers.authorization;
    var user = jwt.parseJwt(token);
    var game = new Game();
    var params = req.body;
    game.result=params.result;
    game.dateStart= new Date(params.dateStart);
    game.dateEnd= new Date(params.dateEnd);
    game.status = params.status;
    game.broadcaster=user.sub;
    game.location= params.location;
    game.category=params.category;
    //tournament:{type:Schema.ObjectId,ref: 'Tournament'},
    var find =Game.find({broadcaster:game.broadcaster,status
       :{$not:{$in:['Terminado','Anulado']}}});
    find.exec((err,games)=>{
        if(err){
            res.status(500).send(err);
        }else{
            if(!games){
                res.status(404).send({message: 'Error en el
                   servidor'});
            }else{
                if(games.length==0){
                    game.save((err,gameSaved)=>{
                        if(err){
                            res.status(500).send(err);
                        }else{
                            if(!gameSaved){
                                res.status(404).send({message
                                   : 'El partido ha sido
                                   guardado'});
                            }else{
                                res.status(200).send({game:
```

```
                                        gameSaved});
                                }
                            }
                        });
                    }else{
                        res.status(404).send({message: 'No se
                            puede crear partido porque tienes un
                            partido sin acabar'});
                    }
                }
            }
        });
}
```

Se utiliza en este caso la entitie *game* para hacer la consulta. Se hacen dos consultas, la primera para comprobar que la persona que esta intentando hacer la petición no tiene ningún partido activo (es decir busca partidos que no esten en el estado terminado o anulado). Si tiene un partido activo dará error y le enviará al front-end un mensaje que dice que ya tiene un partido activo (En teoria el front-end no permite hacer esta petición si ya tenemos un partido activo). Si el locutor no tiene un partido activo guardará el partido en la base de datos y de igual forma se envía una respuesta.
El modelo o la entitie tiene la siguiente sintaxis.

```
var mongoose = require('mongoose');
var Schema = mongoose.Schema;

var GameSchema = Schema({
    result:{type: String, required: true},
    dateStart:{type: Date, required: true},
    dateEnd:{type: Date},
    status:{type: String, required: true},
    broadcaster:{type:Schema.ObjectId,ref: 'User', required:
       true},
    location: {type: {lat:Number,lng:Number}, required: true
       },
    //tournament:{type:Schema.ObjectId,ref: 'Tournament'},
    category:{type:Schema.ObjectId,ref: 'Category',required:
       true}

});

GameSchema.index({location: '2dsphere'});
```

Es una clase que simplemente tiene un esquema que proporciona mongoose, en el que le introduces las diferentes variables y en algunos casos un *index*. El *index* en este caso es para que al ordenar esa variable location, se ordene teniendo en cuenta los dos parámetros. Estos *index* a parte de introducirlos en el código, se pueden definir en el servido de la base de datos de MongoDB, de esta forma aunque se te olvide ponerlo en los esquemas estos *index* seguirán existiendo.

6.2. Vista

La vista, en este caso es una interfaz web que se utiliza desde un navegador. La tecnología empleada para su implementación es Angular. Se han utiliado los lenguajes HTML para la creación de las páginas, CSS para definir los diferentes estilos de las diferentes páginas y JavaScript. Ademas de estos lenguajes de programación se utilizaron varios frameworks, como Angular para la creación de las páginas y Bootstrap que junto a CSS se utilizarán para definir los estilos de página.

```
<div class="registrar" *ngIf="registro">
    <h1>Registrate</h1>
    <div *ngIf="alertRegister">
        <div class="errorRegister alert alert-info">
          {{alertRegister}}
        </div>
    </div>
    <form \#registerForm="ngForm" (submit)="onSubmitRegister
       ()" class="formIdentificar">
      <p>
        <label>Nombre:</label>
        <input type="text" \#name="ngModel" name="name" [(
           ngModel)]="user_register.name" class="form-control
           " required/>
      </p>
      <span *ngIf="!name.valid && name.touched">
          El nombre es obligatorio
      </span>
      <p>
        <label>Correo electr\'o nico:</label>
        <input type="email" \#email="ngModel" name="email" [(
           ngModel)]="user_register.email" class="form-
           control" required/>
        <span *ngIf="!email.valid && email.touched">
          El email es obligatorio
```

```
        </span>
      </p>
      <p>
        <label>Contrase\~n a:</label>
        <input type="password" \#password="ngModel" name="
           password" [(ngModel)]="user_register.password"
           class="form-control" required/>
        <span *ngIf="!password.valid && password.touched">
          La contrase\~n a es obligatorio
        </span>
      </p>

      <input type="submit" value="Registrarse" class="btn btn
         -primary"/>
    </form>
    <span class="changeView btn btn-warning" (click)="
       changeViewToLogin()">
        Identificarse
    </span>
  </div>
```

Este es un ejemplo del div de registrarse en la aplicación, como podemos observar hay código que no se corresponde con código típico HTML, como `"*ngIf"` y `"ngModel"`, este código lo proporciona angular y se utiliza en el primer caso para poner una condición en la que se muestre esa parte de la página, en el segundo caso es para convertir un formulario en un modelo de nuestro componente, para definir que propiedades del formulario se asocian a que atributos de nuestro modelo.

6.3. Controlador

El controlador esta compuesto por un conjunto de clases que se encargan de comunicar el modelo que está en el back-end, para poder obtener, insertar o modificar datos y ademas es responsable de enviar estos datos a la vista y mantenerla actualizada.

En la capa controlador hay varias clases, los componentes o controladores que se encargan de obtener los datos que le envía el usuario a través de la vista y ademas se encargará de contactar con los servicios para poder comunicarse con el back-end y poder recibir los datos necesarios para llevar a cabo la acción deseada.

El componente se declara de la siguiente manera:

```
@Component({
```

```
    selector: 'user-edit',
    templateUrl: '../views/user-edit.html',
    providers: [UserService]
})
```

El atributo selector se utiliza para definir el nombre del componente, de esta forma si utilizamos el siguiente código :

```
<user-edit> </user-edit>
```

en cualquier página HTML se importará el componente, es decir se verá en esa página HTML en la que se ha importado la página user-edit.html y se cargará su respectiva clase del componente.

Este componente tiene ademas una clase. Esta clase tiene sus atributos y un constructor donde se inicializarán si es necesario dichos atributos. También tienen métodos que se corresponde generalmente con acciones de la vista.

Un ejemplo de clase puede ser este:

```
export class UserEditComponent implements OnInit{
    public titulo: string;
    public user: User;
    public telegramData: TelegramData;
    public identity;
    public tokenGenerated;
    public token;
    public alertMessage;
    public url:string;
    public filesToUpload: Array<File>;

    constructor(
        private _userService:UserService
    ){
        this.titulo='Actualizar mis datos';
        this.tokenGenerated=false;
        this.identity = this._userService.getIdentity();
        this.token = this._userService.getToken();
        this.user = this.identity;
        this.telegramData= new TelegramData('',null,'',null,'
           null');
        this.url = GLOBAL.url;
    }
```

La variable `_userService` es el que se encargará de comunicarse con el back-end a través de sus métodos. Como por ejemplo:

```
this._userService.updateUser(this.user).subscribe(
            response=>{
                //Codigo
                }
            },
            error=>{
               //Codigo
            }
        );
```

Para obtener la respuesta del back-end es necesario que se subscriba, de tal forma que cuando envie una respuesta le mandará una función de *callback* con los datos obtenido o si ha habido un error.

6.4. Servicios

La capa de servicios se encargará de comunicar el back-end entre otras cosas. Un servicio se crea de la siguiente forma:

```
@Injectable()
export class GameService{
    public url:string;

    constructor(private _http: Http){
        this.url = GLOBAL.url;
    }
```

Necesita una url para comunicarse con el back-end, la parte de la url que no siempre es la misma para todas las peticiones la obtiene de una clase llamada `GLOBAL`. Para comunicarse con un servicio tiene que crear una petición HTTP como si de una página web se tratase, de tal forma que tiene que crear las cabeceras y las opciones de la petición. Una vez hecho con el método HTTP necesario (POST,GET,PUT,DELETE) enviará la petición al back-end como podemos ver en el siguiente código.

```
getGames(token,id){
        let headers = new Headers({
            'Content-Type':'application/json',
            'Authorization':token
```

```
        });

        let options = new RequestOptions({headers:headers});
        return this._http.get(this.url+'partido-usuario/'+id,
           options).map(res=>res.json());
    }
```

Capítulo 7

Pruebas

En este capítulo se mostrarán todas las pruebas realizadas como comprobación de la aplicación va correctamente. Estas pruebas son de caja negra, es decir a través de la página depory.com a través de la aplicación web y del Bot. Estas pruebas consisten en elegir ciertos casos para probar la aplicación definiendo el resultado que se espera y compararlo con el resultado que se obtiene.

7.1. Aplicación Web

Registro de la aplicación

Prueba realizada.	**Resultado esperado.**	**Resultado obtenido.**	**Válido.**
P1.1 Registro con todos los datos.	El usuario se habrá creado.	El usuario se ha creado.	✓
P1.2 Registro sin un dato.	El usuario no puede darle al botón de registrarse.	El usuario no puede darle al botón de registrarse.	✓
P1.3 Registro con un formato de email incorrecto.	El sistema mostrará un mensaje de error.	El sistema mostró un mensaje de error.	✓

Cuadro 7.1: Pruebas realizadas en el registro.

Inicio de sesión

Prueba realizada.	**Resultado esperado.**	**Resultado obtenido.**	**Válido.**
P2.1 Iniciar sesión con los datos correctos.	El usuario habrá iniciado sesión en el sistema.	El usuario ha iniciado sesión en el sistema.	✓
P2.2 Iniciar sesión sin un dato.	El usuario no puede iniciar sesión y muestra un error.	El usuario no ha podido iniciar sesión y el sistema ha mostrado un error.	✓

P2.3 Iniciar sesión con un usuario y contraseña incorrecta.	El usuario no puede iniciar sesión y muestra un error.	El usuario no ha podido iniciar sesión y el sistema ha mostrado un error.	✓

Cuadro 7.2: Pruebas realizadas en el inicio de sesión.

Editar datos personales

Prueba realizada.	**Resultado esperado.**	**Resultado obtenido.**	**Válido.**
P3.1 Modificar datos.	El sistema actualizará los datos.	se actualizaron los datos.	✓
P3.2 Generar un token de un id de Telegram que no autoriza mensajes desde la aplicación.	No se enviará el mensaje.	No se envió el mensaje.	✓
P3.3 Generar un token de un id de Telegram que autoriza mensajes desde la aplicación.	Se enviará el token.	Se envió el token.	✓
P3.4 Introducir el token correctamente generado desde la misma cuenta.	Se vincularán la cuenta de la aplicación web al id de Telegram.	Se vinculó la cuenta de la aplicación web al id de Telegram.	✓
P3.5 Introducir el token correctamente generado desde otra cuenta o con el token incorrecto.	No se vincularán la cuenta de la aplicación web al id de Telegram y se mostrará un error.	No se vinculó la cuenta de la aplicación web al id de Telegram y se mostró el error.	✓

Cuadro 7.3: Pruebas realizadas en la edición de datos personales.

Ver lista de deportes

Prueba realizada.	**Resultado esperado.**	**Resultado obtenido.**	**Válido.**
P4.1 Entrar en la lista de deportes.	El sistema mostrará los 8 primeros deportes ordenados alfabéticamente.	El sistema mostró los 8 primeros deportes ordenados alfabéticamente.	✓
P4.2 Pulsar el botón siguiente para ver los 8 siguientes deportes si hay mas.	El sistema mostrara la siguiente página de deportes.	El sistema mostró la siguiente página de deportes.	✓

P4.3 Pulsar el botón siguiente para ver los 8 siguientes deportes si no hay mas deportes.	El sistema se quedará en la página actual.	El sistema se quedó en la página actual.	✓
P4.4 Pulsar el botón anterior para ver los 8 anteriores deportes si no es la primera página.	El sistema se mostrará los 8 deportes de la página anterior.	El sistema mostró los deportes de la página anterior.	✓
P4.5 Pulsar el botón anterior para ver los 8 anteriores deportes estando en la primera página.	El sistema se quedará en la página actual.	El sistema se quedó en la página actual.	✓
P4.6 Mostrar botones editar y borrar.	Si el usuario es administrador aparecerán estos dos botones en cada deporte al entrar en la lista.	Si el usuario es administrador aparecen los dos botones en cada deporte.	✓
P4.7 Borrar deporte	El sistema pedirá confirmación en forma de dos botones, uno para confirmar y otro para cancelar.	El sistema pide confirmación en forma de dos botones.	✓
P4.8 Confirmar eliminación deporte.	Al confirmar la eliminación, se eliminará definitivamente el deporte así como todas sus reglas y categorías.	Al confirmar la eliminación se eliminó definitivamente el deporte y sus reglas y categorías.	✓

Cuadro 7.4: Pruebas realizadas en la lista de deportes.

Ver detalle deporte

Prueba realizada.	**Resultado esperado.**	**Resultado obtenido.**	**Válido.**
P5.1 Entrar en el detalle de un deporte con un usuario con rol administrador.	El sistema mostrará el detalle y permitirá añadir reglas y categorías.	El sistema mostró el detalle y permitió añadir reglas y categorías.	✓
P5.2 Entrar en el detalle de un deporte con un usuario con un rol diferente a administrador.	El sistema mostrará el detalle del deporte.	El sistema mostró el detalle del deporte.	✓
P5.3 Añadir una regla con el campo rellenado.	Se podrá pulsar el botón añadir y al pulsarlo se añadirá la regla.	Se puede pulsar el botón añadir y además al pulsarlo añade la regla al deporte.	✓
P5.4 Añadir una regla con el campo vacío.	No se podrá pulsar el botón añadir.	No se puede pulsar el botón añadir.	✓

Añadir una categoría con los campos rellenados.	Se podrá pulsar el botón añadir y al pulsarlo se añadirá la categoría.	Se puede pulsar el botón añadir y además al pulsarlo añade la categoría al deporte.	✓
P5.5 Añadir una categoría con el campo nombre vacío.	No se podrá pulsar el botón añadir.	No se puede pulsar el botón añadir.	✓
P5.6 Borrar una regla/categoría.	Al borrar una regla/categoría se mostrarán dos botones de confirmación, uno para cancelar y otro para eliminar definitivamente.	Al borrar una regla/categoría se mostraron dos botones de confirmación, uno para cancelar y otro para eliminar definitivamente.	✓
P5.7 Confirmar eliminación regla/categoría.	Al confirmar la eliminación se eliminará definitivamente la regla/categoría.	Al confirmar la eliminación se eliminó definitivamente la regla/categoría.	✓
P5.8 Editar regla/categoría.	Al pulsar editar en la regla/categoría se podrá editar los valores de dicha regla/categoría.	Al pulsar editar se pueden modificar los valores de dicha regla/categoría.	✓

Cuadro 7.5: Pruebas en el detalle del deporte.

Crear/Editar deporte

Prueba realizada.	**Resultado esperado.**	**Resultado obtenido.**	**Válido.**
P6.1 Crear/Editar un deporte con todos los datos.	Dejará crear/editar los datos.	Deja actualizar los datos y al pulsar crear/editar deporte se crea/actualiza.	✓
P6.2 Crear/Editar un deporte con un campo sin rellenar.	No dejará crear/editar los datos.	No deja actualizar los datos.	✓
P6.3 Introducir un archivo sin extension gif,jpg o png como imagen.	No se actualizará la imagen al pulsar editar deporte.	No se actualizó la imagen.	✓
P6.4 Introducir un archivo con extension gif,jpg o png como imagen.	Se actualizará la imagen al pulsar editar deporte.	Se actualizó la imagen.	✓

Cuadro 7.6: Pruebas en la edición/creación del deporte.

Ver lista de equipos

Prueba realizada.	Resultado esperado.	Resultado obtenido.	Válido.
P7.1 Entrar en la lista de equipos.	El sistema mostrará los 8 primeros equipos ordenados alfabéticamente.	El sistema mostró los 8 primeros equipos ordenados alfabéticamente.	✓
P7.2 Pulsar el botón siguiente para ver los 8 siguientes equipos si hay mas.	El sistema mostrara la siguiente página de equipos.	El sistema mostró la siguiente página de equipos.	✓
P7.3 Pulsar el botón siguiente para ver los 8 siguientes equipos si no hay más equipos.	El sistema se quedará en la página actual.	El sistema se quedó en la página actual.	✓
P7.4 Pulsar el botón anterior para ver los 8 anteriores equipos si no es la primera página.	El sistema mostrará los 8 equipos de la página anterior.	El sistema mostró los equipos de la página anterior.	✓
P7.5 Pulsar el botón anterior para ver los 8 anteriores equipos estando en la primera página.	El sistema se quedará en la página actual.	El sistema se quedó en la página actual.	✓
P7.6 Mostrar botones editar y borrar.	Si el usuario es administrador aparecerán estos dos botones en cada equipo al entrar en la lista.	Si el usuario es administrador aparecen los dos botones en cada equipo.	✓
P7.7 Borrar equipo.	El sistema pedirá confirmación en forma de dos botones, uno para confirmar y otro para cancelar.	El sistema pide confirmación en forma de dos botones.	✓
P7.8 Confirmar eliminación equipo.	Al confirmar la eliminación, se eliminará definitivamente el equipo.	Al confirmar la eliminación se eliminó definitivamente el equipo.	✓
P7.9 Pulsar la imagen de un equipo.	Al pulsar la imagen de un equipo se mostrará la página de detalle del equipo.	Al pulsar la imagen de un equipo se mostró la página de detalle del equipo.	✓

Cuadro 7.7: Pruebas realizadas en la lista de equipos.

Crear/Editar equipo

Prueba realizada.	Resultado esperado.	Resultado obtenido.	Válido.

P8.1 Crear/Editar un equipo con todos los datos.	Dejará editar los datos y se creara/actualizará el equipo.	Deja actualizar los datos y al pulsar crear/editar equipo se actualiza.	✓
P8.2 Crear/Editar un equipo con un campo sin rellenar.	No dejará editar/crear el equipo.	No deja actualizar/crear el equipo.	✓
P8.3 Introducir un archivo sin extension gif,jpg o png como imagen.	No se actualizará la imagen al pulsar editar equipo.	No se actualizó la imagen.	✓
P8.4 Introducir un archivo con extension gif,jpg o png como imagen.	Se actualizará la imagen al pulsar editar equipo.	Se actualizó la imagen.	✓

Cuadro 7.8: Pruebas en la edición/creación de un equipo.

Mis partidos

Prueba realizada.	**Resultado esperado.**	**Resultado obtenido.**	**Válido.**
P9.1 Entrar a mis partido sin el id de Telegram asociado a la cuenta (Rol locutor).	Se mostrará un mensaje de error en el que pone que falta el Id de telegram en la cuenta.	Se muestra dicho error.	✓
P9.2 Entrar en la sección mis partidos con el id de telegram asociado y sin partido activo/sin empezar.	Se mostrará un botón de crear partido.	Se muestra un botón de crear partido.	✓
P9.3 Entrar en la sección mis partidos con el id de telegram asociado y con un partido activo/sin empezar.	Se mostrarán algunos datos del partido y de los equipos incluidos los botones de eliminar y borrar.	Se muestran dichos datos.	✓
P9.4 Borrar partido.	El sistema pedirá confirmación en forma de dos botones, uno para confirmar y otro para cancelar.	El sistema pide confirmación en forma de dos botones.	✓
P9.5 Confirmar eliminación partido.	Al confirmar la eliminación, se eliminará definitivamente el partido.	Al confirmar la eliminación se eliminó definitivamente el partido.	✓
P9.6 Hacer click en alguno de los datos de un partido activo.	Al hacer click en alguno de los datos de un partido activo se mostrará el detalle del partido.	Al hacer click en alguno de los datos de un partido activo se mostró el detalle del partido.	✓

Cuadro 7.9: Pruebas en mis partidos.

Crear/Editar un partido

Prueba realizada.	Resultado esperado.	Resultado obtenido.	Válido.
P10.1 Crear/Editar un partido con todos los datos.	Dejará crear/editar.	Deja actualizar los datos y al pulsar crear/editar partido se crea/actualiza.	✓
P10.2 Crear/Editar un partido con un campo sin rellenar.	No dejará crear/editar los datos.	No deja crear/actualizar.	✓

Cuadro 7.10: Pruebas en la edición/creación de un partido.

Ver detalle de un partido

Prueba realizada.	Resultado esperado.	Resultado obtenido.	Válido.
P11.1 Entrar en el detalle de un partido con el usuario creador del partido.	El sistema mostrará el detalle y permitirá añadir equipos al partido que puedan participar en el deporte del partido.	El sistema mostró el detalle y permitió añadir equipos que puedan participar en ese deporte.	✓
P11.2 Añadir un equipo a un partido sin el máximo de equipos alcanzado.	Mostrará la lista de equipos que no están participando y se podrán añadir.	Mostró la lista de equipos que no están participando y se pueden añadir.	✓
P11.3 Añadir un equipo a un partido con el máximo de equipos alcanzado.	Mostrará un mensaje de error al añadir un equipo.	Mostró un mensaje de error.	✓
P11.4 Entrar en el detalle de un partido de otro usuario.	Mostrará un mensaje de error de no poder ver ese partido.	Mostró un mensaje de error.	✓
P11.5 Borrar un equipo participante.	Al borrar un participante se mostrarán dos botones de confirmación, uno para cancelar y otro para eliminar definitivamente.	Al borrar un participante se mostraron dos botones de confirmación, uno para cancelar y otro para eliminar definitivamente.	✓
P11.6 Confirmar eliminación equipo participante.	Al confirmar la eliminación se eliminará definitivamente la participación.	Al confirmar la eliminación se eliminó definitivamente la participación.	✓

Cuadro 7.11: Pruebas en el detalle de un partido.

7.2. Bot de Telegram

Preguntar id de Telegram

Prueba realizada.	Resultado esperado.	Resultado obtenido.	Válido.
P12.1 Introducir /id.	El sistema mostrará el id de Telegram.	El sistema mostró el id de Telegram.	✓
P12.2 Introducir /id +otros caracteres.	El sistema no mostrará el id de Telegram.	El sistema mostró el id de Telegram.	X
P12.2 Introducir /id +otros caracteres.	El sistema no mostrará el id de Telegram.	El sistema no mostró el id de Telegram.	✓

Cuadro 7.12: Pruebas al preguntar el id de Telegram.

Autorizar el envío del token

Prueba realizada.	Resultado esperado.	Resultado obtenido.	Válido.
P13.1 Introducir /autorizaToken sin estar autorizado el id de Telegram.	El sistema mostrará un mensaje de se ha autorizado el envío del token.	El sistema mostró que se autorizó el token.	✓
P13.2 Introducir /autorizaToken con el id de Telegram autorizado.	El sistema mostrará un mensaje de error que dice que el id ya esta autorizado.	El sistema mostró el mensaje de error.	✓
P13.3 Introducir /autorizaToken+otros caracteres.	El sistema no mandará ningún mensaje ya que no es ningún comando.	El sistema no envió ningún mensaje.	✓

Cuadro 7.13: Pruebas en la autorización del envío del token.

Desautorizar token

Prueba realizada.	Resultado esperado.	Resultado obtenido.	Válido.
P14.1 Introducir /desautorizaToken.	El sistema mostrará un mensaje de que se ha desautorizado el token.	El sistema mostró que se desautorizó el token.	✓
P14.2 Introducir /desautorizaToken+otros caracteres.	El sistema no mandará ningún mensaje ya que no es ningún comando.	El sistema no envió ningún mensaje.	✓

Cuadro 7.14: Pruebas en la desautorización del envío del token y borrado del enlace de cuentas.

Preguntar partidos cercanos

Prueba realizada.	Resultado esperado.	Resultado obtenido.	Válido.
P15.1 Introducir /partidosCercanos y hay partidos.	El sistema preguntará al usuario si quiere enviar su localización y si acepta mostrará una lista enumerada con los partidos.	P15.2 El sistema preguntó al usuario si quiere enviar su localización y al aceptar se mostró una lista enumerada con los partidos.	✓
P15.3 Introducir /partidosCercanos y no hay partidos.	El sistema preguntará al usuario si quiere enviar su localización y si acepta mostrará un mensaje de no hay partidos en la zona.	El sistema preguntó al usuario si quiere enviar su localización y al aceptar se mostró un mensaje de no hay partidos en la zona.	✓
P15.4 Introducir /partidosCercanos+otros caracteres.	El no preguntará si se quiere enviar la localización sistema no mostrará datos.	El sistema preguntó si el usuario quiere enviar su localización.	X
P15.4 Introducir /partidosCercanos+otros caracteres.	El no preguntará si se quiere enviar la localización sistema no mostrará datos.	El sistema no preguntó al usuario.	✓

Cuadro 7.15: Pruebas realizadas al preguntar los partidos cercanos.

Presenciar un partido

Prueba realizada.	Resultado esperado.	Resultado obtenido.	Válido.
P16.1 Introducir /presenciarPartido + número válido.	El sistema mostrará un mensaje de se ha presenciado el partido.	El sistema mostró un mensaje de que se ha presenciado el partido.	✓
P16.2 Introducir /presenciarPartido + número inválido.	El sistema mostrará un mensaje de error de que no se pueden escribir un número mas grande que los partidos que se han encontrado en la zona.	El sistema mostró el mensaje de error.	✓
P16.3 Introducir /presenciarPartido+otros caracteres.	El bot no responderá dado que no se trata de ningún comando.	El bot no respondió.	✓

Cuadro 7.16: Pruebas realizadas al presenciar un partido.

Preguntar partidos que sigue un usuario

Prueba realizada.	Resultado esperado.	Resultado obtenido.	Válido.
P17.1 Introducir /partidosQueSigo y el usuario sigue partidos.	El sistema mostrará una lista enumerada con los partidos que sigue.	El sistema mostró una lista enumerada con los partidos que sigue.	✓
P17.2 Introducir /partidosQueSigo y no sigue ningún partido.	El sistema preguntará mostrará un mensaje de no sigue ningún partido.	El sistema mostró un mensaje de no sigue ningún partido.	✓
P17.3 Introducir /partidosQueSigo+otros caracteres.	El no mostrará datos.	El sistema mostró datos.	X
P17.3 Introducir /partidosQueSigo+otros caracteres.	El no mostrará datos.	El sistema no mostró datos.	✓

Cuadro 7.17: Pruebas realizadas al preguntar los partidos que sigue el usuario.

Dejar de seguir un partido

Prueba realizada.	Resultado esperado.	Resultado obtenido.	Válido.
P18.1 Introducir /dejarDeSeguirPartido + número válido.	El sistema mostrará un mensaje de se ha dejado de seguir el partido.	El sistema mostró un mensaje de que se ha dejado de seguir el partido.	✓
P18.2 Introducir /dejarDeSeguirPartido + número inválido.	El sistema mostrará un mensaje de error de que no se pueden escribir un número mas grande que los partidos que se están siguiendo.	El sistema mostró el mensaje de error.	✓
P18.3 Introducir /dejarDeSeguirPartido+otros caracteres.	El bot no responderá dado que no se trata de ningún comando.	El bot no respondió.	✓

Cuadro 7.18: Pruebas realizadas al dejar de presenciar un partido.

Consultar ubicación de un partido

Prueba realizada.	Resultado esperado.	Resultado obtenido.	Válido.
P19.1 Introducir /ubicacionPartido + número válido.	El sistema mostrará la ubicación del partido.	El sistema mostró la ubicación del partido.	✓
P19.2 Introducir /ubicacionPartido + número inválido.	El sistema mostrará un mensaje de error de que no se pueden escribir un número mas grande que los partidos que hay en la zona.	El sistema mostró el mensaje de error.	✓
P19.3 Introducir /ubicacionPartido+otros caracteres.	El bot no responderá dado que no se trata de ningún comando.	El bot no respondió.	✓
P19.4 Introducir /ubicacionPartidoQueSigo + número válido.	El sistema mostrará la ubicación del partido que el usuario está siguiendo.	El sistema mostró la ubicación del partido.	✓
P19.5 Introducir /ubicacionPartidoQueSigo + número inválido.	El sistema mostrará un mensaje de error de que no se pueden escribir un número mas grande que los partidos que se están siguiendo.	El sistema mostró el mensaje de error.	✓
P19.6 Introducir /ubicacionPartidoQueSigo+otros caracteres.	El bot no responderá dado que no se trata de ningún comando.	El bot no respondió.	✓

Cuadro 7.19: Pruebas realizadas al consultar la ubicación de un partido.

Consultar reglas de un partido

Prueba realizada.	Resultado esperado.	Resultado obtenido.	Válido.
P20.1 Introducir /reglasPartido + número válido.	El sistema mostrará las reglas del deporte del partido.	El sistema mostró las reglas del deporte del partido.	✓
P20.2 Introducir /reglasPartido + número inválido.	El sistema mostrará un mensaje de error de que no se pueden escribir un número mas grande que los partidos que hay en la zona.	El sistema mostró el mensaje de error.	✓
P20.3 Introducir /reglasPartido+otros caracteres.	El bot no responderá dado que no se trata de ningún comando.	El bot no respondió.	✓

P20.4 Introducir /reglasPartidoQueSigo + número válido.	El sistema las reglas del deporte del partido que esta siguiendo.	El sistema mostró las reglas del deporte del partido.	✓
P20.5 Introducir /reglasPartidoQueSigo + número inválido.	El sistema mostrará un mensaje de error de que no se pueden escribir un número mas grande que los partidos que se están siguiendo.	El sistema mostró el mensaje de error.	✓
P20.6 Introducir /reglasPartidoQueSigo+otros caracteres.	El bot no responderá dado que no se trata de ningún comando.	El bot no respondió.	✓

Cuadro 7.20: Pruebas realizadas al consultar las reglas de un partido.

Empezar un partido

Prueba realizada.	**Resultado esperado.**	**Resultado obtenido.**	**Válido.**
P21.1 Introducir /empezarPartido con un partido con estado sin empezar.	El sistema mostrará un mensaje de que se ha actualizado el partido al estado en curso y enviará mensaje a los seguidores del partido.	Se mostraron los mensajes esperados.	✓
P21.2 Introducir /empezarPartido con un partido con estado en curso.	El sistema mostrará un mensaje de error de que no tiene ningún partido en estado sin empezar.	El sistema mostró el mensaje de error.	✓
P21.3 Introducir /empezarPartido+otros caracteres.	El bot no responderá dado que no se trata de ningún comando.	El bot respondió.	X
P21.3 Introducir /empezarPartido+otros caracteres.	El bot no responderá dado que no se trata de ningún comando.	El bot no respondió.	✓

Cuadro 7.21: Pruebas realizadas al empezar un partido.

Cambio de resultado de un partido

Prueba realizada.	**Resultado esperado.**	**Resultado obtenido.**	**Válido.**

P22.1 Introducir /cambiarResultado +resultado con un partido en estado en curso.	El sistema mostrará un mensaje de que se ha actualizado el resultado y enviará mensaje a los seguidores del partido.	Se mostraron los mensajes esperados.	✓
P22.2 Introducir /cambiarResultado +resultado con un partido con estado sin empezar.	El sistema mostrará un mensaje de error de que no tiene ningún partido en estado en curso.	El sistema mostró el mensaje de error.	✓
P22.3 Introducir /cambiarResultado+otros caracteres.	El bot no responderá dado que no se trata de ningún comando.	El bot no respondió.	✓

Cuadro 7.22: Pruebas realizadas al cambiar el resultado de un partido.

Consulta de espectadores de un partido

Prueba realizada.	**Resultado esperado.**	**Resultado obtenido.**	**Válido.**
P23.1 Introducir /espectadoresPartido con un partido en estado en curso o sin empezar.	El sistema mostrará el número de espectadores que tiene el partido.	Se mostraron los espectadores del partido.	✓
P23.2 Introducir /espectadoresPartido sin un partido en estado en curso o sin empezar.	El sistema mostrará un mensaje de error de que no tiene ningún partido en curso.	El sistema mostró el mensaje de error.	✓
P23.3 Introducir /espectadoresPartido+otros caracteres.	El bot no responderá dado que no se trata de ningún comando.	El bot respondió.	X
P23.3 Introducir /espectadoresPartido+otros caracteres.	El bot no responderá dado que no se trata de ningún comando.	El bot no respondió.	✓

Cuadro 7.23: Pruebas realizadas al consultar los espectadores de tu partido.

Terminar un partido

Prueba realizada.	**Resultado esperado.**	**Resultado obtenido.**	**Válido.**

P24.1 Introducir /terminarPartido con un partido en estado en curso.	El sistema mostrará un mensaje de se ha terminado el partido y se lo comunicará a todos los seguidores del partido.	Se mostraron los mensajes esperados.	✓
P24.2 Introducir /terminarPartido sin un partido en estado en curso.	El sistema mostrará un mensaje de error de que no tiene ningún partido en curso.	El sistema mostró el mensaje de error.	✓
P24.3 Introducir /terminarPartido+otros caracteres.	El bot no responderá dado que no se trata de ningún comando.	El bot respondió.	X
P24.3 Introducir /terminarPartido+otros caracteres.	El bot no responderá dado que no se trata de ningún comando.	El bot no respondió.	✓

Cuadro 7.24: Pruebas realizadas al terminar un partido.

Anular un partido

Prueba realizada.	**Resultado esperado.**	**Resultado obtenido.**	**Válido.**
P25.1 Introducir /anularPartido con un partido en estado en curso.	El sistema mostrará un mensaje de se ha anulado el partido y se lo comunicará a todos los seguidores del partido.	Se mostraron los mensajes esperados.	✓
P25.2 Introducir /anularPartido sin un partido en estado en curso.	El sistema mostrará un mensaje de error de que no tiene ningún partido en curso.	El sistema mostró el mensaje de error.	✓
P25.3 Introducir /anularPartido+otros caracteres.	El bot no responderá dado que no se trata de ningún comando.	El bot respondió.	X
P25.3 Introducir /anularPartido+otros caracteres.	El bot no responderá dado que no se trata de ningún comando.	El bot no respondió.	✓

Cuadro 7.25: Pruebas realizadas al anular un partido.

Capítulo 8

Manual de Instalación

Este capítulo se explicarán los prerrequisitos técnicos, sistema operativo, etc, necesarios para instalar la aplicación desarrollada en este trabajo de fin de grado.

Los requisitos no son muy altos, un sistema de 1 GB de memoria debería ser suficiente para arrancar el sistema. El sistema operativo es independiente, lo único que tiene que ser capaz de instalar npm para poder gestionar y arrancar el equipo. En este caso se utilizó un servidor de Ubuntu 18.04 x64.//

Una vez tenemos un servidor lo único que hay que hacer es una conexión ssh a nuestro servidor, o entrar en la consola directamente si el proveedor del servidor lo proporciona. En caso de hacer la conexión se debera hacer de la siguiente forma:

```
ssh usuario@ip
```

Pedirá una contraseña, y una vez se haya comprobado se realizará la conexión ssh. Una vez dentro hay que utilizar una serie de comandos para preparar el servidor. Primero habrá que bajar un gestor de paquetes (No es necesario pero es recomendable).

```
apt-get update
apt-get install aptitude
aptitude update
```

Ahora, con este nuevo gestor de paquete se instalará git. Git se utiliza para tener un repositorio y poder trabajar de una forma más cómoda con un control de versiones.

```
aptitude install git
```

Para poder controlar las versiones de una manera mas sencilla se instalara nvm por si alguna version de node no es compatible con nuestro programa o cualquier otro *framework* que necesite ser instalado.

```
aptitude install build-essential libssl-dev
```

```
curl -o- https://raw.githubusercontent.com/creationix/nvm/v0.33.11/install.
    sh | bash
source ~/.profile
```

El primer comando son dependencias de nvm y el último es para corregir un bug que en el que no se listan bien las versiones./

Para listar las versiones disponibles de angular bastará con utilizar el siguiente comando:

```
nvm ls-remote
```

Se recomienda siempre que se instale una versión que tenga escrito al lado (Latest LTS). Significa que es la ultima versión que tiene soporte y que es de larga duración. Para instalar una versión de Node.js tendremos que utilizar el comando

```
nvm install version
```

Ahora instalaremos un servidor web llamado nginx.

```
nvm install nginx
```

Para comprobar que esta instalado solo se tiene que coger la dirección Ip y ponerla en el navegador. Si esta instalado tendrá que salir la página de bienvenida de nginx.

Ahora se creara una carpeta donde se descargará nuestra aplicación. En nuestro caso la crearemos en el directorio home.

```
cd /home
mkdir deporty-produccion
```

Una vez creado, solo tendremos que bajar nuestra aplicación. Para ello utilizaremos Git. Solamente hay que utilizar el comando:

```
git clone url-del-repositorio
```

Ahora hay que crear las carpetas de los archivos que se subirán en nuestra web.

```
cd /home/deporty-produccion/deporty
mkdir -p uploads/users
mkdir -p uploads/sports
mkdir -p uploads/teams
```

En esas carpetas es donde se guardarán nuestras imágenes de los usuarios,deportes y equipos.

Ahora instalaremos todas las dependencias del proyecto, estas vienen en un archivo llamado package.json utilizando el comando:

```
npm update
```

La aplicación ya esta lista para el despliegue utilizando el comando

```
npm start
```

Si vamos a nuestra Ip y el puerto 3977 deberá funcionar correctamente. Aunque esto funciona, es recomendable hacer una serie de pasos para que simplemente entrando en la Ip salga la aplicación es decir que este en el puerto 80. Es decir lo que haremos sera redirigir los datos que lleguen al puerto 80 hacia el puerto 3977 utilizando el comando:

```
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j REDIRECT --to-
    port 3977
```

Para que al cerrar la conexión ssh no se cierre el servidor tendremos que instalar un software que nos permite tener procesos de Node.js en producción.

```
npm install pm2 -g
```

Para arrancar la aplicación se tendrá que usar el comando

```
pm2 start index.js
```

Y un vez completado este comando nuestra aplicación estará funcionando aunque cerremos la consola con la conexión ssh.

Para que esta aplicación funcione es necesario introducir la ip del servidor en el servidor de MongoDB, ya que nuestra aplicación utiliza un servidor de base de datos diferente y tiene que hacer una conexión. Para hacer que la base de datos acepte nuestras peticiones sera necesario entrar en la administración e introducir la dirección en la *WhiteList* para que no rechace dichas peticiones.

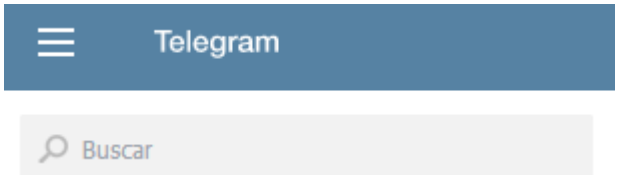
Capítulo 9

Manual de Usuario

Este capítulo final se dedica a desarrollar el manual del usuario final, así como del usuario administrador y del usuario locutor.

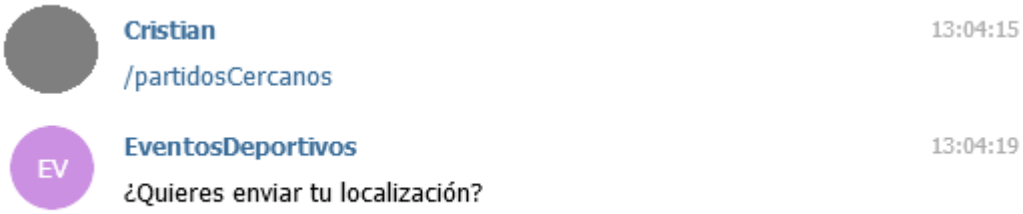
9.1. Manual de Usuario

Para usar el bot de Telegram deberemos utilizar el buscador de telegram.

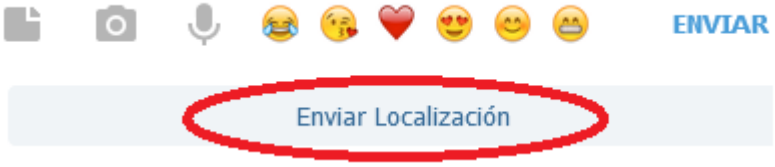


Tendrás que buscar el bot introduciendo '@EventosDeportivosBot'. Saldrá una pestaña de esta forma:

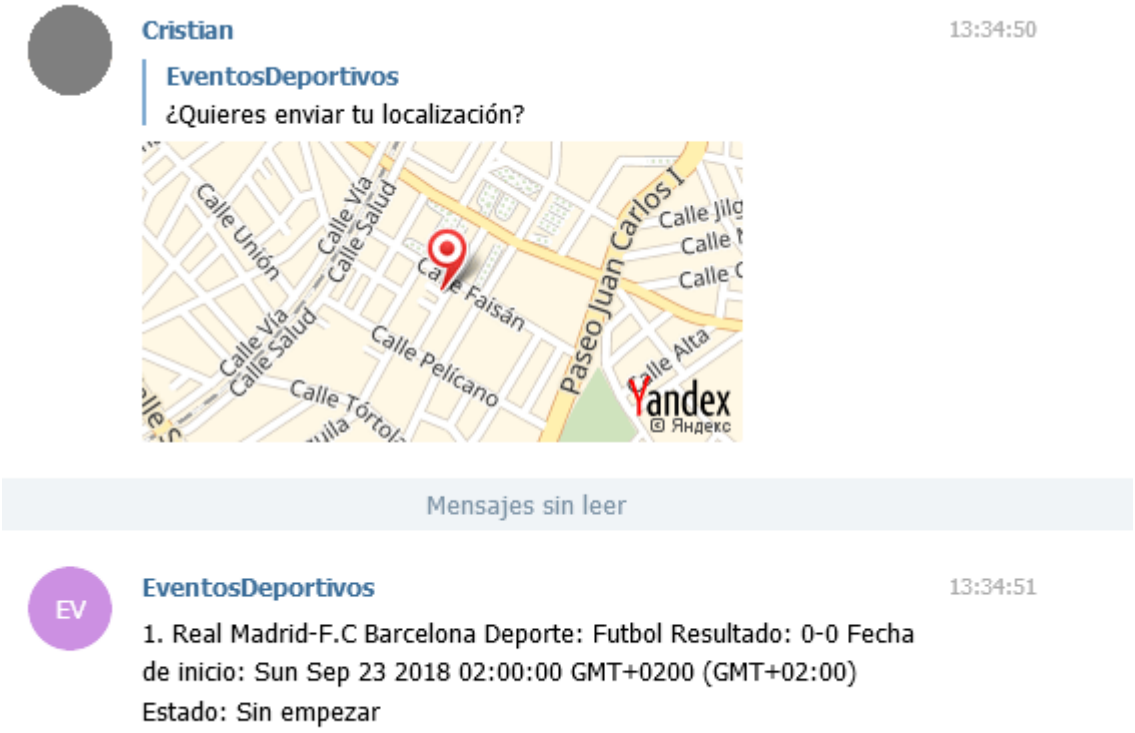
Al hacer click en el nombre se abrirá una conversación. Para buscar un partido cercano solo tendremos que escribir el comando: /partidosCercanos. Saldrá un botón para que envíes tu ubicación.



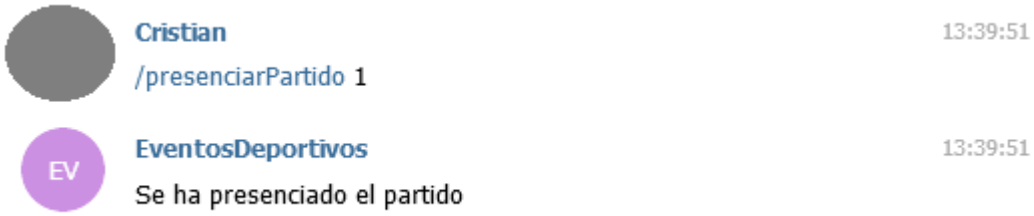
El botón aparecerá justo debajo del chat tal y como se indica en la siguiente foto:



Una vez confirmado, el bot buscará y listará cada partido que haya cerca de tu ubicación y los enumerará. En caso de que no haya ningún partido el bot también te informará.
//

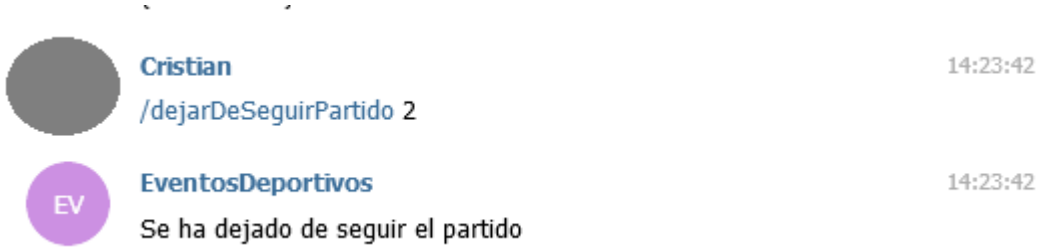


En esta foto de ejemplo solo hay un partido, y esa es la información que da del partido.
//
Para poder presenciar uno de eso partidos tendrás que poner el comando /presenciarPartido y el número del partido que quieras presenciar. Ejemplo: /presenciarPartido 1. El bot responderá si se ha podido presenciar el partido o no, en caso de que no dirá la causa.
//



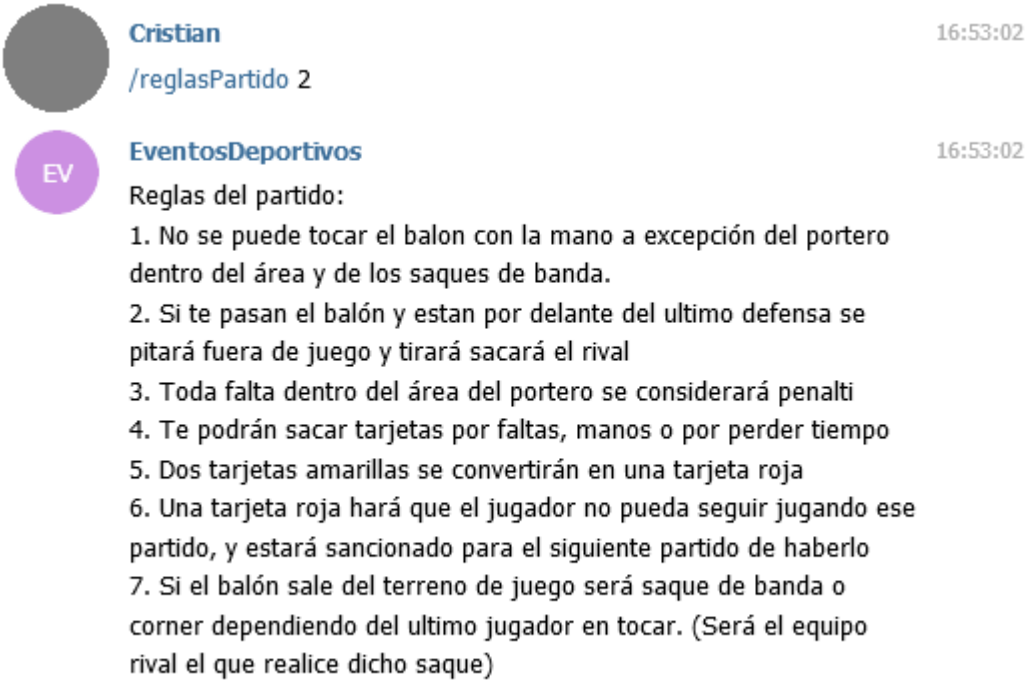
Para poder ver los partidos que estas presenciando solo tienes que escribir el comando /partidosQueSigo, el bot listará los partidos de la misma manera que con el comando /partidosCercanos.

Para dejar de seguir un partido tendrás primero que usar el comando /partidosQueSigo, y una vez obtengas la lista de partidos utilizar el comando /dejarDeSeguirPartido y el número del partido. Ejemplo /dejarDeSeguirPartido 1.



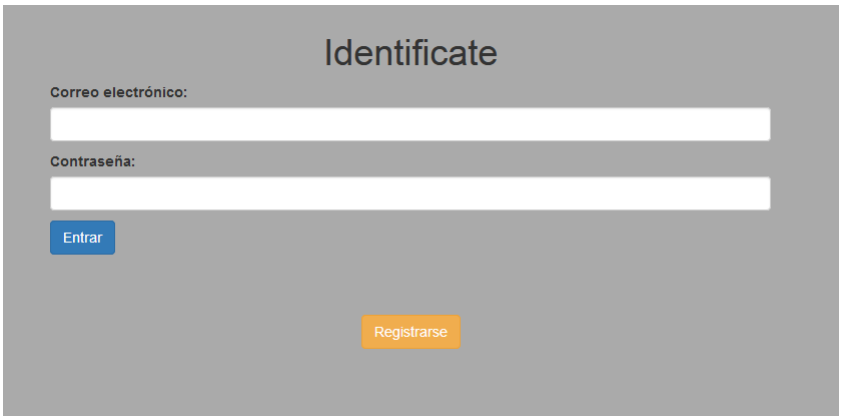
Para conocer la ubicación de un partido a través de un mapa de Google Maps podrás utilizar el comando /ubicacionPartido y el número del partido para ver la ubicación de un partido que no estas presenciando actualmente o /ubicacionPartidoQueSigo y el número para ver la ubicación de un partido que si estas siguiendo.

Para consultar las reglas de un partido, solo tendrás que utilizar el comando /reglasPartido y el número del partido para ver la ubicación de un partido que no estas siguiendo actualmente o /reglasPartidoQueSigo y el número para ver las reglas de un partido que si estas siguiendo.



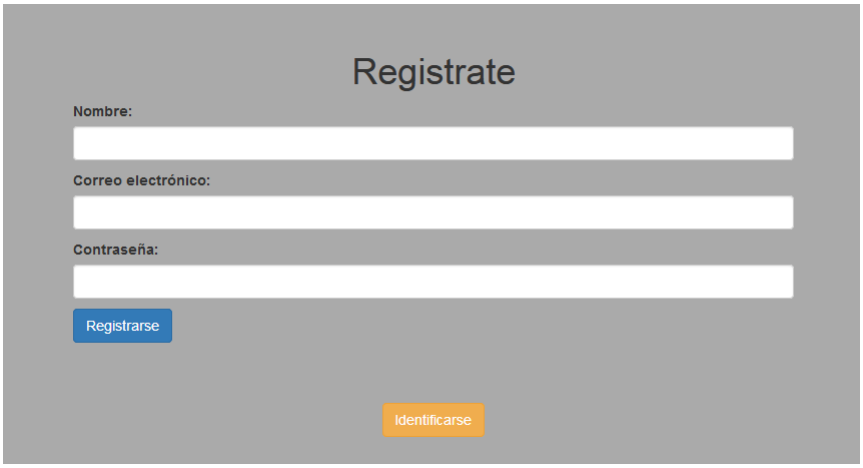
9.2. Manual del Usuario Locutor

Para poder narrar un partido lo primero que tienes que hacer es entrar en la aplicación web de deporty `depory.com`. Aparecerá la pantalla principal como la que podemos observar en la siguiente imagen.



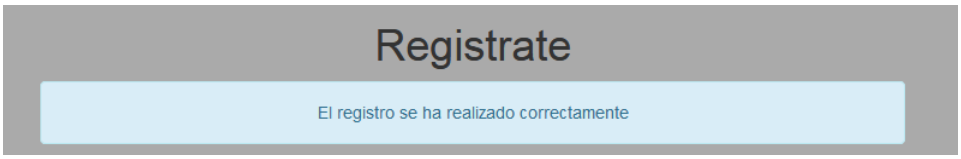
Si ya tienes cuenta lo único que tienes que hacer es introducir tu correo electrónico y contraseña y entrarás en la página principal.

Si no tienes cuenta tendrás que hacer click en el botón 'Registrarse'y aparecerá la siguiente pantalla.



Una vez introducidos los datos, para crear una cuenta deberás pulsar el botón de registrarse que aparece de color azul.

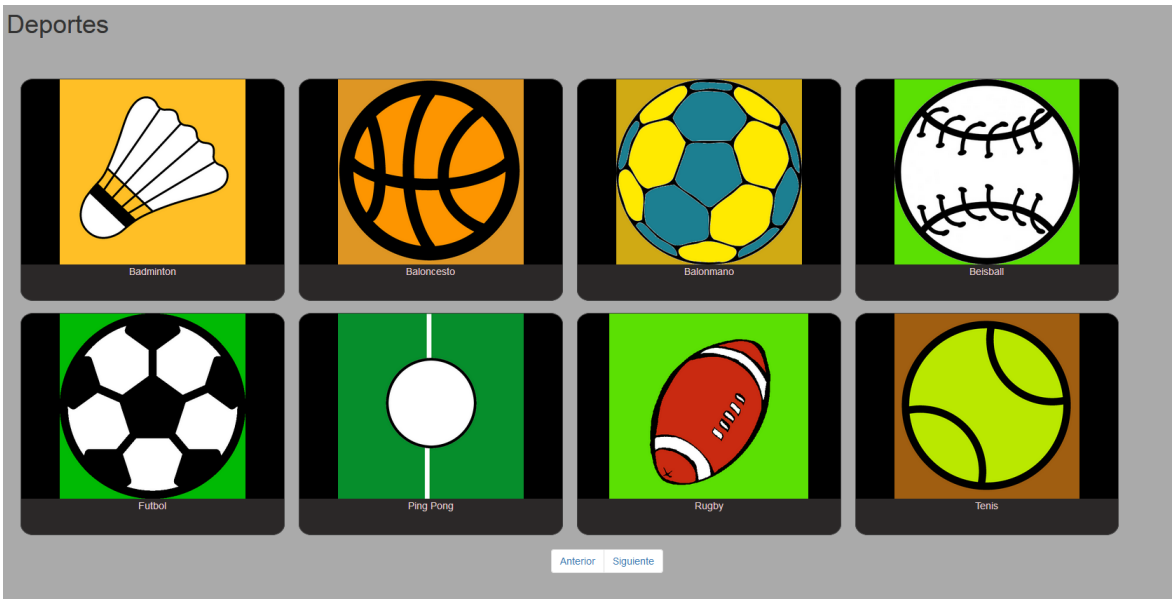
Si todo ha ido bien se creará una cuenta y dará un mensaje de confirmación:



Solo tendras que hacer click en identificarse para volver a la pantalla de inicio de sesión e introducir tus datos para entrar en la aplicación.

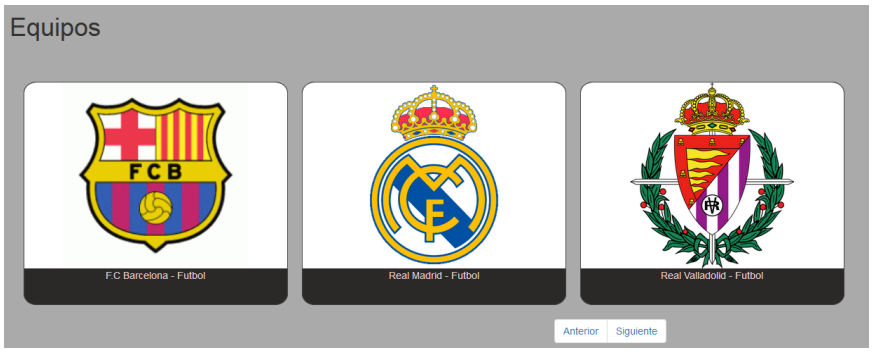
Esta la página principal de la aplicación web. Como se puede ver tiene varios menús.

Para ver todos los deportes y sus características se deberá pulsar el botón que tiene una pelota de fútbol como icono. Aparecerá una página como esta:



Los deportes estarán listados de ocho en ocho y al pulsar siguiente o anterior irás a una página u otra.
Al hacer click sobre la imagen del deporte iremos a la página del detalle del deporte que es la siguiente:

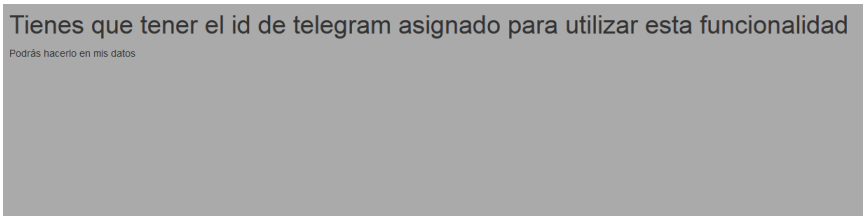
En esta pantalla se pueden ver todas las regla y todas las categorías del deporte. En el botón que tiene como icono mucha gente corresponde a la lista de equipos. Al hacer click iras a la siguiente página:



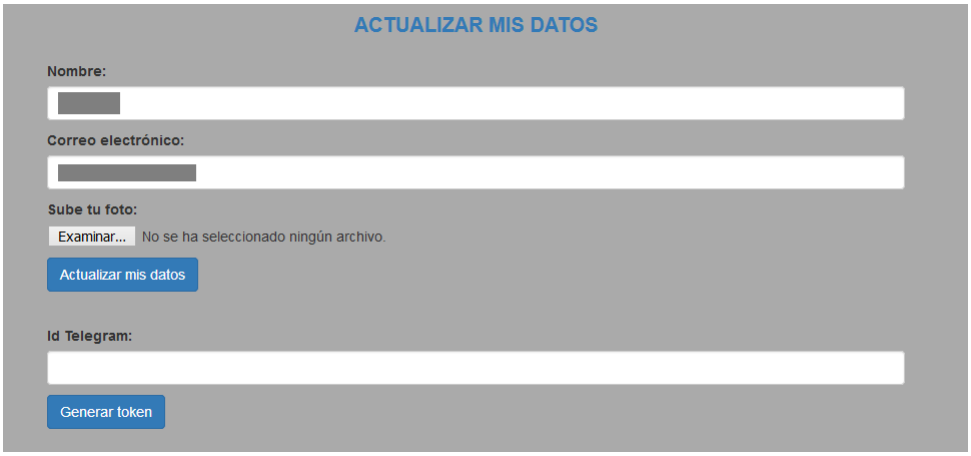
En esta página veras todos los equipos que existen actualmente en la aplicación. Si haces click sobre la imagen del equipo irás al detalle del equipo.

En el detalle podrás ver el nombre, el tag, la descripción del equipo y el deporte en el que puede participar. En la pestaña de mis partidos que tiene un icono de un móvil iremos a la sección de partidos. Allí podrás ver si tienes algún partido activo o si quieres

crearlo. Si aún no tenemos asociada la cuenta de Telegram con la de la aplicación saldrá la siguiente pantalla:



Para asociarlo solo tenemos que ir al menú mis datos, que esta en la parte baja del menú. Saldrá la siguiente página:



Para asociar nuestra cuenta de Telegram, solo tenemos que ir al bot, y preguntarle cual es nuestro id en Telegram con el comando /id y el bot nos lo dará.

Para que el bot te pueda enviar el token, primero tienes que autorizar el envío de mensajes desde la aplicación web escribiendo el comando /autorizaToken. Para eliminar esta autorización o para eliminar el enlace entre las cuentas solo hay que escribir el comando

/desautorizaToken.

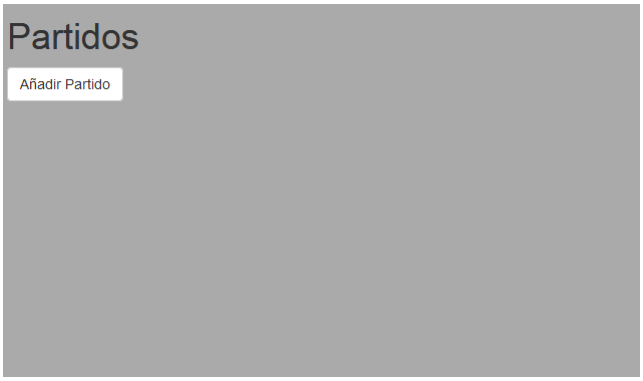
Cuando tengamos nuestro id de Telegram iremos al campo Id Telegram, introduciremos nuestro id y pulsaremos el botón, 'Generar token'.
Ahora se mostrará un campo que se llama 'Token generadoén el que tendrás que introducir los caracteres que te envía el bot. Estos son personales de cada usuario, es decir el *token* que te genera a ti no puede ser introducido por ningún otro usuario de la aplicación y ademas este *token* caduca a los 5 minutos.

Envío del *token*:

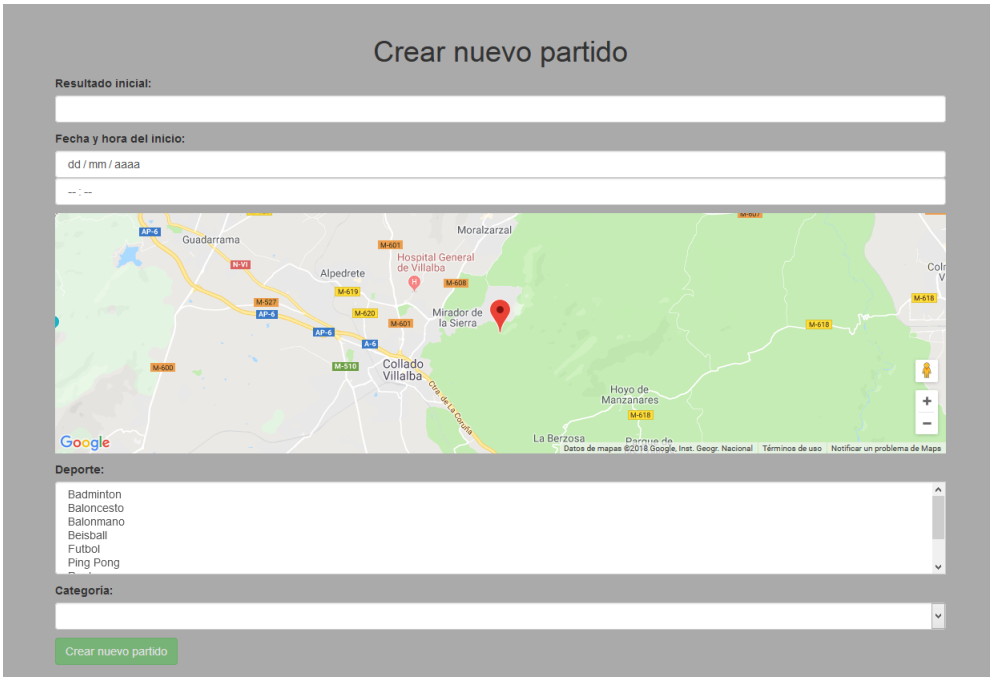
Si introducimos el *token* que nos ha enviado en el campo 'Token generado'y pulsamos el botón 'Confirmar token'si este es correcto para este usuario y no esta caducado se enlazarán las cuentas.

Una vez tengamos el id de Telegram asociado a nuestra cuenta aparecerá encima del menú mis datos nuestro id de Telegram asociado.

Una vez tengamos las cuentas asociadas podemos entrar en el menú 'Mis partidos'. Si no tenemos ningún partido que no haya terminado nos saldrá el botón de Áñadir Partido'.

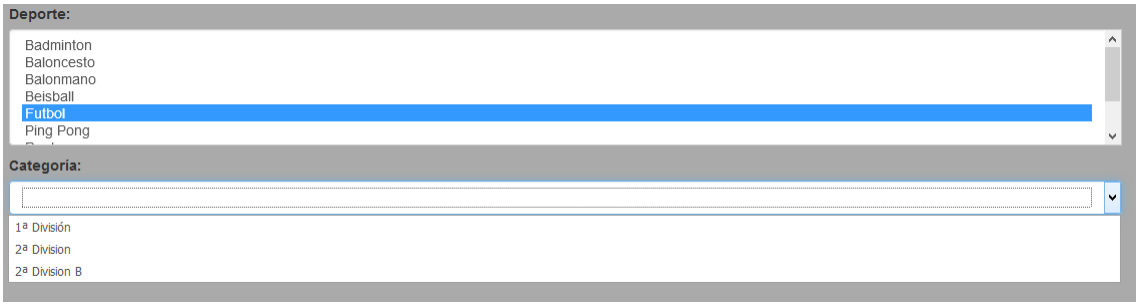


Si pulsamos el botón, nos llevará a una página para introducir todos los datos necesarios para la creación del partido del partido.



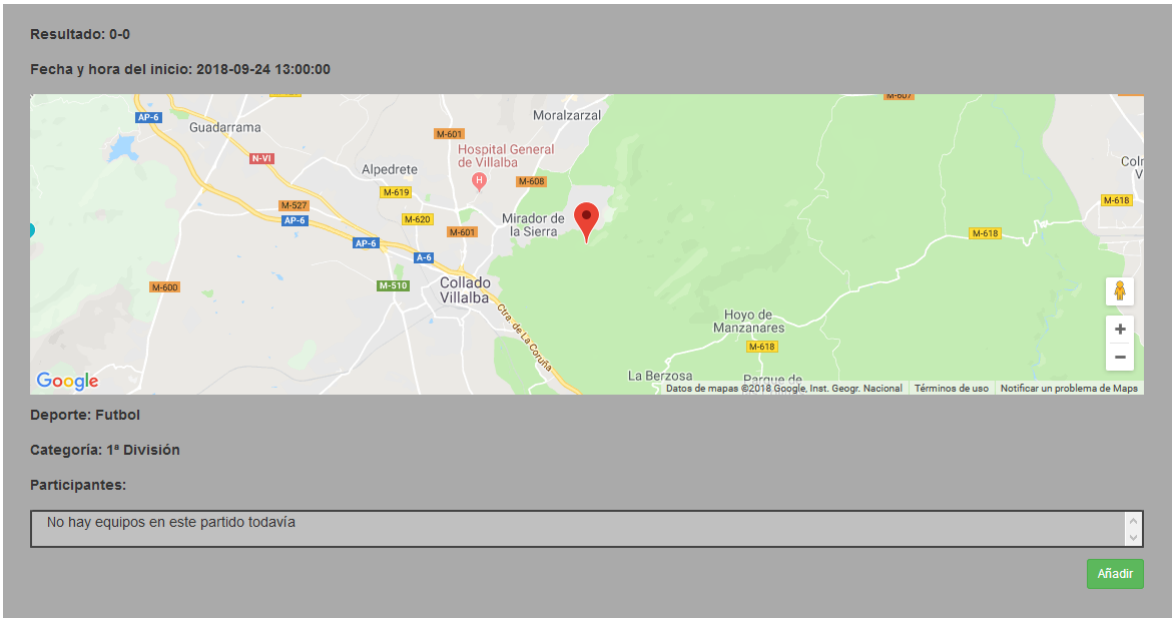
El mapa es de Google Maps, por lo que es interactivo y al pulsar en cualquier punto del mapa pondrá un punto que es donde tendrá lugar el encuentro.

Al pulsar el deporte del partido automáticamente saldrán todas las categorías existentes en la aplicación para ese deporte.

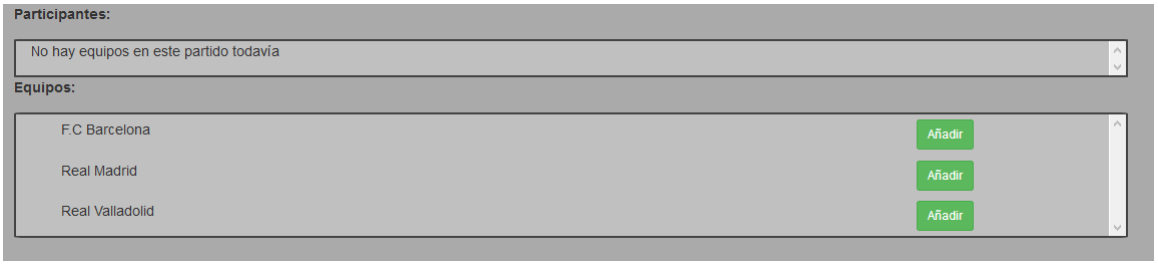


Una vez introducidos todos los datos, podremos pulsar el botón 'Crear nuevo partido'. Si todo ha ido bien iremos a la página de 'Mis partidos'y se podrá ver que hay un partido sin equipos.

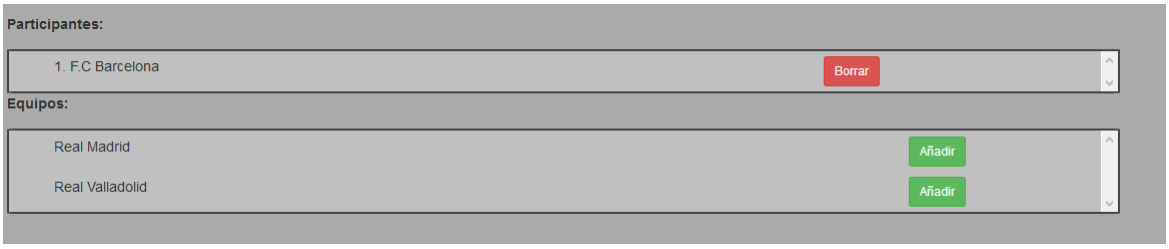
Para introducir equipos al partido lo único que tienes que hacer es hacer click en el partido para entrar en su detalle.



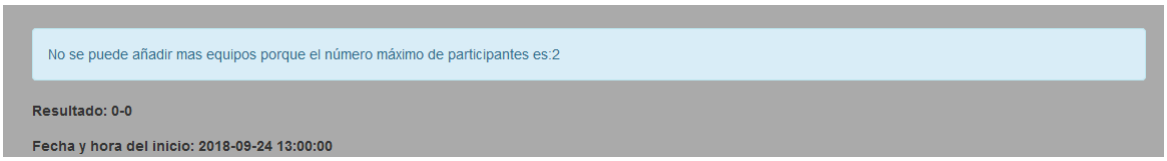
Para introducir un equipo a un partido deberás pulsar el botón añadir en la sección de participantes. Saldrá otra sección con todos los deportes que hay en ese deporte.



Solo tendrás que pulsar añadir en cada fila del equipo que quieres que participe. Cuando se añade un equipo a un partido, este deja de estar en la lista de equipos disponibles y pasa a la lista de participantes.

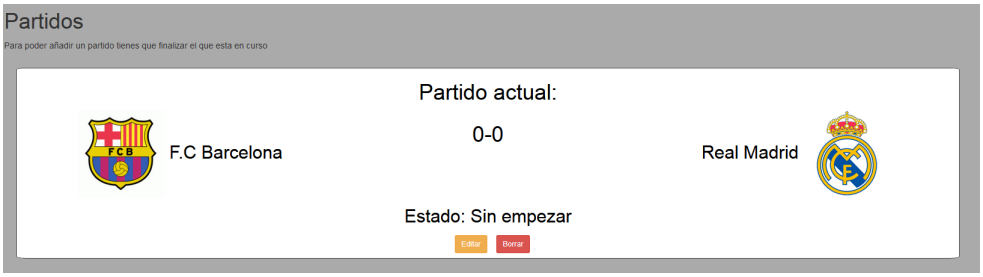


No se pueden meter mas equipos de los que permite el deporte en el partido, si esto llega a ocurrir dará error.



Para eliminar un equipo de un partido se deberá pulsar el botón borrar. Si es pulsado saldrá una confirmación en forma de dos botones, uno para eliminar definitivamente y otro para cancelar.

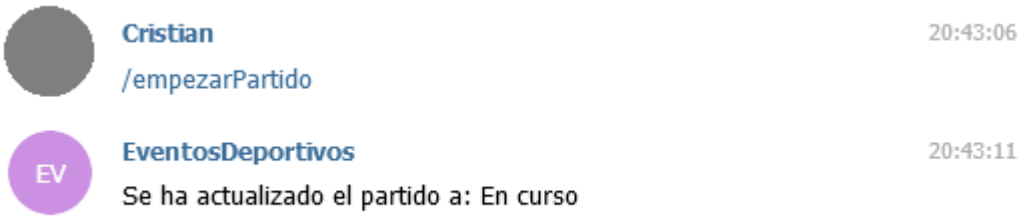
Una vez introducidos los equipos al partido se mostrarán en la página de 'Mis partidos'.



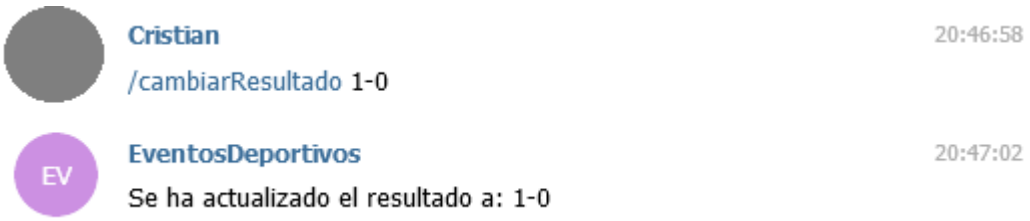
Para eliminar el partido de la misma forma que en el caso anterior pedirá una confirmación de su eliminación. Si queremos editar el partido solo tenemos que pulsar el botón editar y nos llevará a una página exactamente igual que la de la creación.

Una vez creado el partido ya esta listo para que cualquier persona pueda seguirlo y cada acción que haya en el partido serán informados automáticamente.

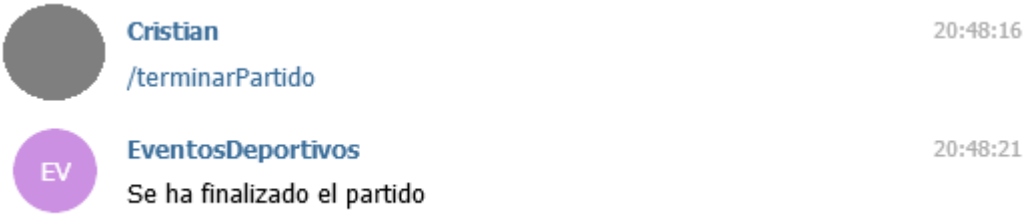
Para hacer que el partido que estas narrando empiece solo tienes que escribir el comando /empezarPartido. Si todo va bien el bot responderá te informará si se ha producido el cambio y si se ha producido informará a los seguidores del partido.



Para cambiar el resultado de un partido solo tienes que utilizar el comando /cambiarResultado y el resultado. Ejemplo /cambiarResultado 1-0.

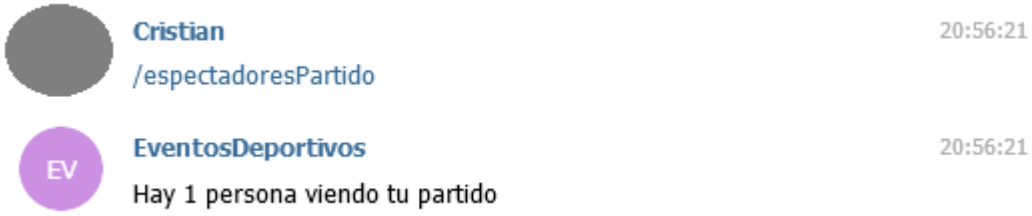


Para finalizar un partido solo tienes que utilizar el comando /terminarPartido. Ya no podrás modificar el resultado una vez se termine el partido. Solo se podrá terminar el partido si estaba en el estado Én curso'.



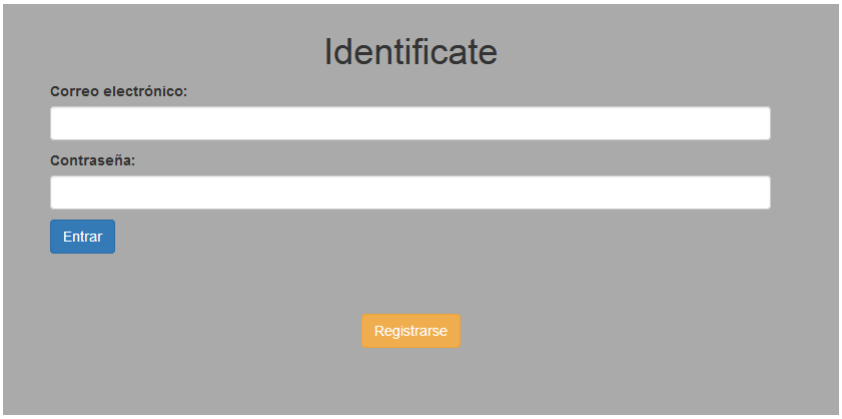
Para anular un partido solo tienes que utilizar el comando /anularPartido. Ya no podrás modificar el resultado una vez se anule el partido. Solo se podrá anular el partido si estaba en el estado Én curso'.

Para ver los seguidores que tiene tu partido podrás utilizar el comando /espectadores-Partido. Este es el único comando que no enviará a los usuarios un mensaje.



9.3. Manual de Administración

Para poder editar los datos de los deportes y equipos deberás entrar en la aplicación web de deporty `depory.com` . Aparecerá la pantalla principal como la que podemos observar en la siguiente imagen.



Solo tendrás que introducir tu correo electrónico y tu contraseña para acceder a tu usuario. Esta es la página principal de la aplicación web. Como se puede ver tiene varios menús.

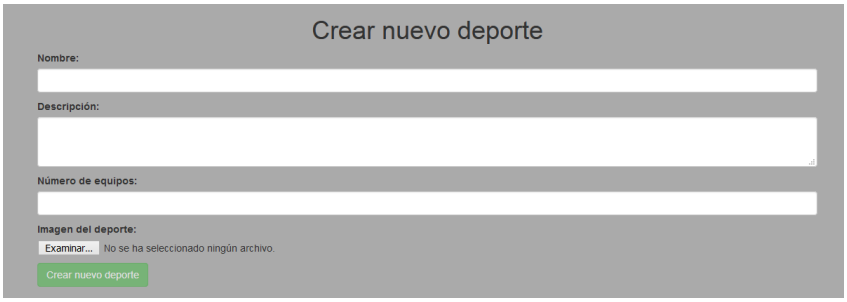
Para modificar tus datos solo tienes que ir al menú 'Mis datos'y se mostrará la página donde podrás modificar tus datos de usuario.

Al pulsar el botón de Áctualizar mis datos', se mostrará un mensaje.

Para ver todos los deportes y sus características se deberá pulsar el botón que tiene una pelota de fútbol como icono. Aparecerá una página como esta:

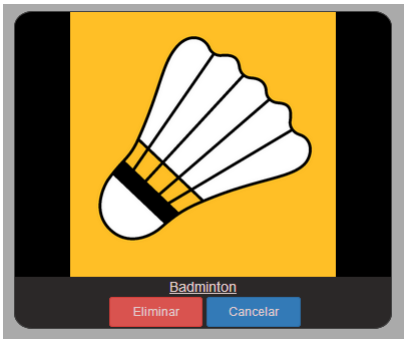
Los deportes estarán listados de ocho en ocho y al pulsar siguiente o anterior irás a una página u otra.

Para introducir un deporte nuevo en el sistema deberás pulsar el botón Áñadir deporte'. Se mostrará la siguiente página:

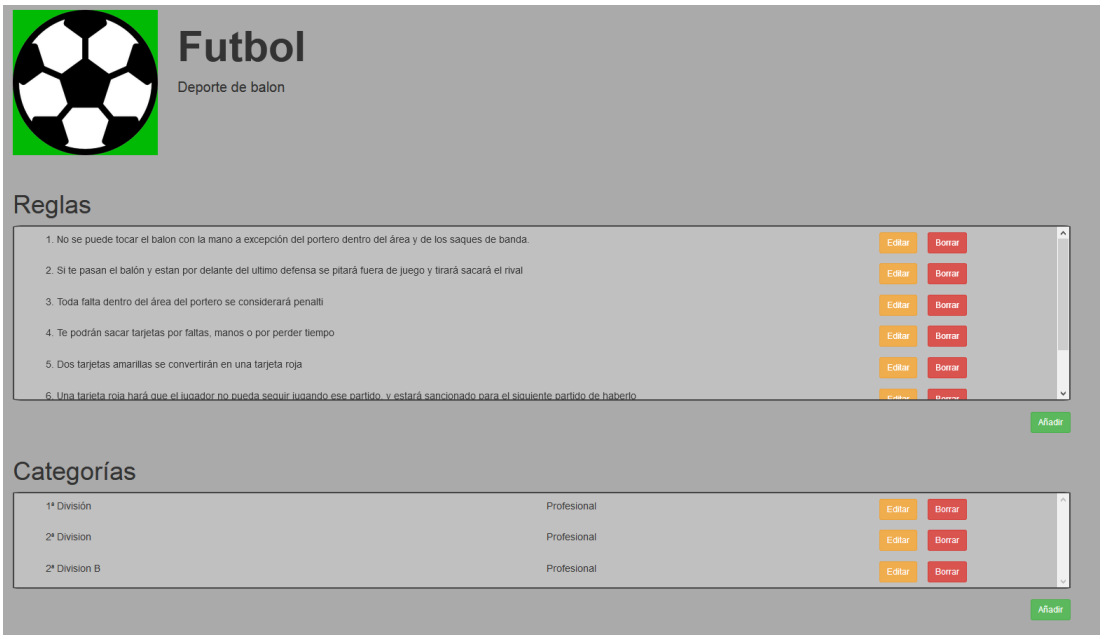


Una vez introducido todos los datos deberás pulsar el botón 'Crear nuevo deporte'. Una vez introducido el deporte se mostrará la pantalla de editar deporte.

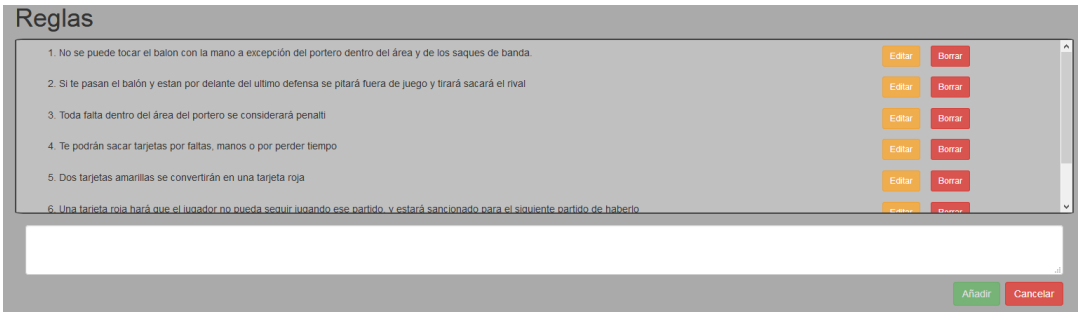
Desde la lista de deportes, hay un botón borrar. Al pulsarlo sobre un deporte pedirá una confirmación a través de dos botones.



Si pulsamos en eliminar se borrará de forma definitiva el deporte. Si pulsamos el botón borrar en otro deporte se cancelará la preparación del borrado del anterior deporte, volviendo a los botones habituales de Éditar deporte'y 'Borrar deporte'. Al hacer click sobre la imagen del deporte iremos a la página del detalle del deporte que es la siguiente:

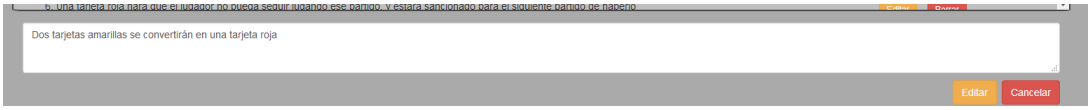


En esta página se podrán añadir reglas y categorías. Para añadir una regla deberemos pulsar el botón añadir que se sitúa debajo del cuadro de reglas y saldrá un campo para poder introducirla.



Una vez introducida la información de la regla deberás pulsar el botón de Añadir. Si pulsas cancelar el campo se ocultará.

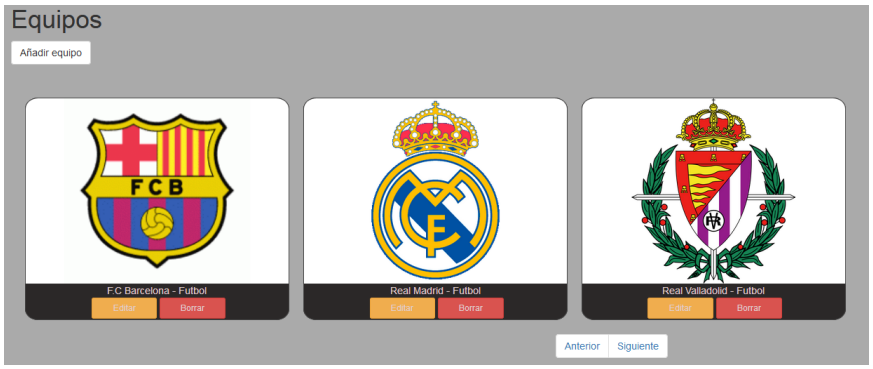
Para editar una regla tienes que pulsar el botón de Éditarén la fila de la regla que quieras modificar, y de la misma forma saldrá un campo para que se pueda modificar.



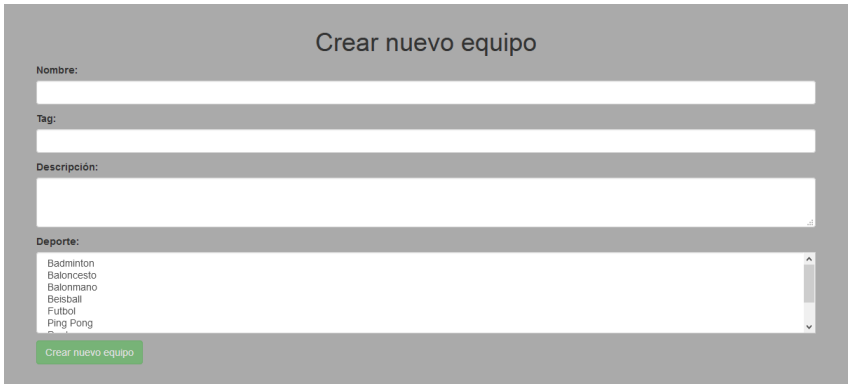
Si quieres eliminar una regla lo único que tienes que hacer es pulsar el botón eliminar. De igual forma que al eliminar el deporte te pedirá una confirmación a través de dos botones.

Para añadir, editar y borrar categorías se hace de la misma forma que las reglas. Lo único es que para añadir una categoría hace falta introducir dos campos.

Para ver los equipos y sus características se deberá pulsar el botón que tiene una varias personas como icono. Aparecerá una página como esta:



Para crear un equipo tienes que pulsar el botón de Áñadir Equipo'. Al pulsarlo se mostrará la siguiente pantalla:



Una vez introducidos los datos, al pulsar el botón 'Crear nuevo equipo'se creará el equipo y se mostrará la pantalla de edición de dicho equipo.

Para borrar un equipo, de igual forma que se hace con los deportes se deberá pulsar el botón 'Borrar', este pedirá una confirmación a través de un botón.

Al hacer click sobre la imagen del equipo se mostrará la página del detalle del equipo.

En el detalle podrás ver el nombre, el tag, la descripción del equipo y el deporte en el que puede participar.

Capítulo 10

Conclusiones y trabajo futuro

Los resultados con respecto a la realización de la aplicación han sido bastante positivos, se han logrado todos los objetivos propuestos, aunque ha habido algún *Bug* se ha corregido gracias a las pruebas empleadas. Aunque las pruebas han sido bastante extensas pueden no cubrir todos los casos que se pueden dar por lo que en un futuro podrá haber algún otro fallo con el uso de la aplicación. Por este motivo se han dedicado 100 horas al mantenimiento de la aplicación por año aunque probablemente el año siguiente sea menor si no se aumentan las funcionalidades.

Fase	**Tiempo estimado (Horas)**	**Tiempo real (Horas)**
1. Análisis	42	49
Requisitos	12	11
Casos de uso	16	18
Diagrama de clases	6	5
Diagramas de secuencia	8	10
2. Diseño	29	30
Diagrama entidad-relación	6	5
Diagrama de clases	6	3
Diagrama de clases	4	7
Diagramas de secuencia	13	15
3. Implementación	102	123
Formación de equipos	26	31
Implementación del back-end	20	27
Implementación del front-end	36	40
Implementación del bot de telegram	20	25
4. Pruebas	10	10
Total	183	212

Cuadro 10.1: Resultado en horas/persona de cada fase.

Los resultados con respecto al tiempo empleado han sido satisfactorios aunque se ha terminado con un retraso de dos semanas debido a que los tiempos estimados han sido en algunos casos bastante bajos, sobretodo debido a la amplitud del proyecto y la inexperiencia del equipo en el desarrollo de aplicaciones MEAN stack y por la poca experiencia del equipo con el Api del bot de Telegram. Por este motivo se retrasó la fecha que se estimó en un principio.

El coste ha supuesto un aumento de 435€en el presupuesto, lo que hace un total de 5592€para desarrollar y mantener este proyecto durante un año.

Las ventajas de que sea una aplicación implementada con Node.js es que la hace escalable por lo que necesitaremos de menos recursos Hardware en el futuro para que esta funcione correctamente. Otra ventaja de nuestra aplicación es que al utilizar un bot de Telegram, no necesitamos hacer la parte del diseño de las vistas del móvil, dado que nos la proporciona Telegram y se comunica a través de eventos. La ventaja de utilizar un bot de Telegram es que los espectadores no necesitan registrarse en la aplicación para seguir partidos. La mayor desventaja es que el bot limita mucho las acciones del usuario, ya que hay comandos que serian muy difíciles de implementar en el bot. Por eso la necesidad de hacer una página web para la comodidad de los usuarios.

Gracias a la realización de este trabajo, he aprendido MEAN Stack, es decir MongoDB, Express, Angular y Node.js. MongoDB es un gestor de base de datos no sql, hasta que empecé a hacer este trabajo no sabia de la existencia de este tipo de gestores de bases de datos. De angular si es verdad que se da en una asignatura pero gracias a la formación que se ha realizado para crear esta aplicación he aprendido muchas cosas nuevas. Lógicamente tampoco conocía Node.js hasta la realización de este trabajo de fin de grado ni del Api de Telegram. Además también he aprendido como crear aplicaciones web donde el front-end y el back-end están separados, y como se comunican entre ellos para atender a las diferentes peticiones.

En cuanto a control de versiones he aprendido Git, ya que es cierto que se da en una asignatura pero con la realización de este proyecto, he aprendido bastantes cosas nuevas y a como utilizarlo. Ademas he aprendido a crear el entorno en un servidor para poder ejecutar la aplicación y que los usuarios puedan utilizarlo.

Como posibles mejoras podrían estar implementar un sistema de torneos en el que puedas seguir el torneo. Al seguir el torneos automáticamente seguirás todos los partidos y te llegarán notificaciones de sus cambios. Otra posible mejora sea que dentro de los resultados se puedan añadir resultados, para que así partidos como el tenis que se necesitan hacer sets, cada set tiene puntos etc... Mejorar el diseño en móviles de la aplicación web. Bien es cierto que es funcional pero no se ve como debería, por lo tanto seria una buena mejora.

Bibliografía

[1] Alberto Basalo, *[Online]. Disponible: https://desarrolloweb.com/articulos/que-es-angularjs-descripcion-framework-javascript-conceptos.html. Ultimo acceso 1 Agosto 2018*
Información sobre que es Angular, que mejoras aporta al código HTML y el uso que hace del patrón MVC.

[2] G. Colouris, J. Dollimore, T. Kindberg and G. Blair., *Sistemas distribuidos: Conceptos y Diseño (3ªEd). Addison-Wesley,2001. Capítulo 2: Modelos de sistema.*
Información sobre los diferentes modelos de sistema, así como sus capas, la arquitectura cliente-servidor y sus variaciones, servidores proxy y caches y requisitos de diseño para la calidad de dichas arquitecturas.

[3] Google, *[Online]. Disponible: https://angular.io/docs. Ultimo acceso 7 Agosto 2018.*
Información del Api de Angular, junto con tutoriales, técnicas y fundamentos.

[4] Google Commerce Ltd, *[Online]. Disponible: https://www.mismarcadores.com/. Ultimo acceso 18 Septiembre 2018.*
Página web de Mis Marcadores donde se puede ver toda la funcionalidad de la aplicación.

[5] MongoDB, Inc., *[Online]. Disponible: https://docs.mongodb.com/manual/contents/. Ultimo acceso 18 Septiembre 2018.*
Información sobre la documentación de mongoDB. Incluye una guía de como instalarlo y varias guías sobre su uso.

[6] Netconsulting, *[Online]. Disponible: https://www.netconsulting.es/blog/nodejs/. Ultimo acceso 1 Agosto 2018.*
Información sobre la definición de lo que es un bot de Telegram, de su historia, de su gestión, el impacto que ha tenido y de las ventajas y desventajas de este entorno.

[7] Node.js Foundation, *[Online]. Disponible: https://nodejs.org/es/docs/. Último acceso 1 Septiembre 2018.*
Información sobre el Api de Node.js en las diferentes versiones existentes, también incluye guías y la información correspondiente a sus dependencias.

[8] Real academia española, *[Online]. Disponible: http://dle.rae.es/srv/fetch?id=CFEFwiY. Ultimo acceso 18 Septiembre 2018.*
Información sobre la definición de lo que es considerado un deporte.

[9] SofaScore, *[Online]. Disponible: https://www.sofascore.com/es/. Ultimo acceso 3 Septiembre 2018.*
Página web de SofaScore donde se puede ver la funcionalidad de la aplicación.

[10] Stratebi, *[Online]. http://www.stratebi.es/todobi/Oct13/MongoDB_UpRunning.pdf. Último acceso 15 Agosto 2018.*
Información acerca de qué es MongoDB y que ventajas presenta frente a las bases de datos SQL.

[11] Telegram Messenger LLP, *[Online]. Disponible: https://core.telegram.org/bots/api. Último acceso 20 Septiembre 2018.*
Información sobre el Api de Telegram en el que se encontrarán todos los métodos y clases existentes.

[12] TJ Holowaychuk, StrongLoop y otros, *[Online]. Disponible: https://expressjs.com/es/4x/api.html. Ultimo acceso 1 Septiembre 2018.*
Información sobre el Api de Express con todos los métodos y como usarlo, también ofrece una guía en español para entender como funciona este framework.

[13] Wikipedia, *[Online]. Disponible: https://es.wikipedia.org/wiki/MEAN. Ultimo acceso 1 Septiembre 2018.*
Información sobre qué es MEAN Stack,acerca de su historia, cuales son sus componentes y las funciones de cada uno de ellos.

[14] Wikipedia, *[Online]. Disponible: https://es.wikipedia.org/wiki/Telegram_Bot_API. Ultimo acceso 18 Septiembre 2018.*
Información sobre la definición de lo que es un bot de Telegram, de su historia, de su gestión y el impacto que ha tenido.

Contenido del CD

Se entrega un CD con el código del desarrollo de este trabajo de fin de grado, cuyo nombre de la carpeta es deporty. En ella tendremos el Back-end y el Front-end listo para el desarrollo. El Back-end es la carpeta client-dev. El resto a excepción de la carpeta Client que se utiliza para el despliegue de la aplicación es el Back-end. También se adjunta el archivo astah con el análisis y el diseño de la aplicación y la memoria de dicho trabajo.