Universidad de Valladolid

ESCUELA DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA
MENCIÓN EN INGENIERÍA DE SOFTWARE

Watch Footbal: Aplicación para la visualización de resultados deportivos.

Alumno: Eduardo González García

Tutor: Carlos Enrique Vivaracho Pascual

A mi familia, que siempre me apoyó

Agradecimientos

A mi familia, por apoyarme durante toda mi vida y ayudarme a llegar a quien soy hoy.

A mis amigos y compañeros de la carrera, que me han animado en tiempos difíciles y me han ayudado a crecer tanto personal como académicamente.

A mi tutor Carlos por ayudarme a presentar el proyecto.

Gracias

Resumen

El objetivo de este proyecto es desarrollar una aplicación web que facilite el acceso a información de actualidad sobre resultados deportivos a cualquier tipo de público interesado. La aplicación se enfocará en difundir información futbolística como pueden ser Ligas, equipos o jugadores. Para ello nos serviremos de la información brindada por el equipo de api-sports en su API de resultados futbolísticos "API-Football".

Las herramientas utilizadas para el desarrollo de este proyecto han sido el framework para el desarrollo web vue.js, JavaScript, HTML5 y CSS utilizando una metodología ágil (Scrum) para la monitorización y seguimiento del proyecto.

Abstract

The objective of this project is to develop a web application that facilitates access to current sports information results to any type of interested public. The application will focus on broadcast football information such as leagues, teams or players. For this purpose we will use the information provided by the api-sports team in their API about football results “API-Football”.

The tools used for the development of this project have been the framework for web development vue.js, JavaScript, HTML5 and CSS using an agile methodology (Scrum) for the monitoring and follow-up of the project.

Índice general

Agradecimientos **III**

Resumen **V**

Abstract **VII**

Lista de Figuras **XV**

Lista de Tablas **XIX**

1. Introducción **1**

1.1. Introducción 1

1.2. Motivación 1

1.3. Público Objetivo 1

1.4. Aplicaciones Similares 2

1.5. Objetivos 3

1.5.1. Objetivos específicos 3

1.5.2. Objetivos personales 3

1.6. Estructura de la memoria 3

2. Conceptos Previos **5**

2.1. Frameworks front-end 5

2.1.1. Angular 5

2.1.2. React.js 6

2.1.3. Vue.js 6

2.1.4. Framework front-end escogido 7

2.2. Proveedor de información 7
2.2.1. Besoccer 7
2.2.2. football-data.org 8
2.2.3. TodoPorElFutbol.com 8
2.2.4. API-Football 8
2.2.5. Proveedor de información escogido 9
2.3. Plataforma back-end 9
2.3.1. Firebase 9
2.3.2. Back4App 10
2.3.3. Parse 10
2.3.4. AWS Amplify 11
2.3.5. Plataforma back-end escogida 11

3. Planificación y presupuesto del proyecto **13**

3.1. Scrum 13
3.1.1. ¿Qué es Scrum? 13
3.1.2. El Equipo Scrum (*Scrum Team*) 13
3.1.3. Eventos Scrum 14
3.1.4. Artefactos Scrum 15
3.1.5. Historias de Usuario 15
3.2. Adaptación de Scrum al proyecto 15
3.2.1. Planificación inicial de los sprints 16
3.2.2. Descripción de historias de usuario 16
3.2.3. Modificaciones posteriores de la planificación inicial 17
3.3. Planificación del presupuesto 19

4. Plan de riesgos **21**

4.1. Plan de riesgos 21
4.1.1. ¿Qué es un riesgo? 21
4.1.2. Como actuar ante los riesgos 22
4.1.3. Como crear un plan de riesgos 22

4.1.4. Riesgos encontrados en el proyecto . . . 22

5. Tecnologías utilizadas **27**

5.1. Herramientas utilizadas . . . 27

5.1.1. Git . . . 27

5.1.2. API-Football . . . 28

5.1.3. Vue.js . . . 29

5.1.4. Firebase . . . 32

5.1.5. NPM . . . 32

5.1.6. Astah . . . 34

5.1.7. Trello . . . 34

5.1.8. Zotero . . . 34

5.1.9. Overleaf . . . 34

5.1.10. InkScape . . . 34

5.1.11. MockFlow . . . 34

5.1.12. Coolors . . . 34

5.2. Herramientas descartadas . . . 35

5.2.1. Storybook . . . 35

5.2.2. Adobe InDesign . . . 35

5.2.3. Beautiful Soup . . . 35

5.2.4. @vue/cli-plugin-unit-jest . . . 35

5.2.5. GitLab CI/CD . . . 35

6. Seguimiento del proyecto **37**

6.1. Seguimiento de los sprints realizados . . . 37

6.1.1. Sprint 0 . . . 37

6.1.2. Sprint 1 . . . 38

6.1.3. Sprint 2 . . . 39

6.1.4. Sprint 3 . . . 40

6.1.5. Sprint 4 . . . 41

6.1.6. Sprint 5 . . . 41

6.1.7. Sprint 6 . . . 42
6.1.8. Sprint 7 . . . 42
6.1.9. Sprint 8 . . . 44
6.2. Planificación extendida . . . 44
6.2.1. Sprint 9 . . . 45
6.2.2. Sprint 10 . . . 46
6.2.3. Sprint 11 . . . 47
6.2.4. Sprint 12 . . . 48
6.2.5. Sprint 13 . . . 48

7. Diseño **51**

7.1. Model-View-ViewModel . . . 51
7.1.1. Vue.js y MVVM . . . 51
7.2. Atomic Design . . . 53
7.2.1. Aplicación al proyecto . . . 53
7.3. Diseño de la interfaz de usuario . . . 61
7.3.1. Bocetos . . . 61
7.3.2. Paleta de Colores . . . 67
7.3.3. Fuentes Utilizadas . . . 68
7.4. Diseño de la base de datos . . . 69
7.4.1. Diseño inicial . . . 69
7.4.2. Diseño Firebase . . . 71
7.5. Modelo de despliegue de la web . . . 71

8. Implementación **73**

8.1. Estructura del directorio del proyecto . . . 73
8.2. Medidas de seguridad implementadas . . . 76
8.3. Decisiones tomadas a lo largo del proyecto . . . 76
8.4. Cambios realizados a lo largo del proyecto . . . 77
8.4.1. Cambios en la interfaz de usuario . . . 77
8.4.2. Cambios en implementación del modelo conceptual . . . 77

8.5. Licencia 77

9. Pruebas **79**

9.1. Introducción 79

9.2. Pruebas en este proyecto 80

9.2.1. Pruebas de Componentes 80

9.2.2. Pruebas End-to-end 81

10. Conclusiones **83**

10.1. Conclusiones 83

10.2. Lineas Futuras 84

A. Manual de usuario **85**

A.1. Vista Home 85

A.2. Vista Countries 86

A.3. Vista Country 86

A.4. Vista Leagues 86

A.5. Vista League 87

A.6. Vista Fixture 88

A.7. Vista Team 93

A.8. Vista Player 95

A.9. Vista Coach 96

A.10. Cambio de idioma 97

A.11. SignIn 98

A.12. SignUp 101

A.13. Herramientas disponibles tras el Login 104

B. Manual de ejecución y despliegue **115**

B.1. Herramientas necesarias 115

B.2. Instalación del proyecto 115

B.3. Ejecución 116

B.4. Despliegue 116

C. Resumen de enlaces adicionales **119**

Bibliografía **121**

Lista de Figuras

5.1. Arquitectura de la API [40]. 29

5.2. Ciclo de vida de la instancia Vue[45]. 31

7.1. Comparativa entre MVP y MVVM [71]. 52

7.2. Implementación de MVVM en Vue [71]. 52

7.3. Estructuras de Atomic Design [72]. 54

7.4. Diagrama de componentes. 60

7.5. Boceto navegación. 61

7.6. Boceto vista login. 62

7.7. Boceto vista SignUp 63

7.8. Boceto vista League. 63

7.9. Boceto vista Fixture. 64

7.10. Boceto vista Team. 65

7.11. Boceto vista Player. 65

7.12. Boceto vista Coach. 66

7.13. Boceto vista User. 67

7.14. Boceto vista EditUser. 67

7.15. Paleta de Colores. 68

7.16. Diagrama conceptual DB de fútbol. 70

7.17. Diagrama conceptual DB de usuarios. 71

7.18. Diagrama de despliegue. 72

A.1. Apariencia final de la página Home. 85

A.2. Apariencia final de la página Countries. 86

A.3. Apariencia final de la pestaña Leagues en la vista Country. 87

A.4. Apariencia final de la pestaña Teams en la vista Country. 87

A.5. Apariencia final de la página Leagues. 88

A.6. Apariencia final de la página League. 89

A.7. Visualización de las herramientas de cambio de temporada. 89

A.8. Visualización de las herramientas de filtrado. 90

A.9. Visualización de las herramientas de organizar. 90

A.10.Apariencia final de la pestaña Standings en la vista League. 91

A.11.Apariencia final de la pestaña Teams en la vista League. 91

A.12.Apariencia final de la pestaña Top Players en la vista League. 92

A.13.Apariencia final de la página Fixture. 92

A.14.Apariencia final de la pestaña Events en la vista Fixture. 93

A.15.Apariencia final de la pestaña Head To Head en la vista Fixture. 93

A.16.Apariencia final de la pestaña Lineups en la vista Fixture. 94

A.17.Apariencia final de la pestaña Statistics en la vista Fixture. 94

A.18.Apariencia final de la página Team. 95

A.19.Visualización de las herramientas de selección de temporadas. 96

A.20.Apariencia final de la pestaña Statistics en la vista Team. 96

A.21.Apariencia final de la pestaña Leagues en la vista Team. 97

A.22.Apariencia final de la pestaña Fixtures en la vista Team.. 97

A.23.Apariencia final de la pestaña Players en la vista Team. 98

A.24.Apariencia final de la pestaña Transfers en la vista Team. 98

A.25.Apariencia final de la pestaña Coachs en la vista Team. 99

A.26.Apariencia final de la pestaña Statistics en la vista Player. 99

A.27.Apariencia final de la pestaña Transfers en la vista Player. 100

A.28.Apariencia final de la pestaña Trophies en la vista Player. 100

A.29.Apariencia final de la pestaña Sidelined en la vista Player. 101

A.30.Apariencia final de la pestaña Career en la vista Coach. 101

A.31.Apariencia final de la pestaña Trophies en la vista Coach. 102

A.32.Apariencia final de la pestaña Sidelined en la vista Coach. . . . 102
A.33.Visualización de las herramientas de cambio de idioma. . . . 103
A.34.Apariencia de la aplicación tras el cambio de idioma. . . . 103
A.35.Apariencia final de la página SignIn. . . . 104
A.36.Error en la página SignIn al no rellenar los campos. . . . 104
A.37.Apariencia de la página SignIn con los campos rellenados. . . . 105
A.38.Mensaje de inicio de sesión correcto. . . . 105
A.39.Formulario para recuperar contraseña. . . . 106
A.40.Apariencia final de la página SignUp. . . . 106
A.41.Error en la página signUp al no rellenar los campos. . . . 107
A.42.Apariencia final de la ventana de términos y condiciones. . . . 107
A.43.Menú de usuario. . . . 108
A.44.Menú de usuario en la barra de navegación. . . . 108
A.45.Apariencia final de la página de Usuario. . . . 109
A.46.Botón añadir liga de la página Leagues. . . . 109
A.47.Botón eliminar liga de la página Leagues. . . . 110
A.48.Botón añadir partido de la página Fixtures. . . . 110
A.49.Botón eliminar partido de la página Fixtures. . . . 111
A.50.Botón añadir equipo de la página Teams. . . . 111
A.51.Botón eliminar equipo de la página Teams. . . . 112
A.52.Apariencia final de la pestaña Leagues en la vista User. . . . 112
A.53.Apariencia final de la pestaña Fixtures en la vista User. . . . 113
A.54.Apariencia final de la pestaña Teams en la vista User. . . . 113
A.55.Apariencia final de la página de EditUser. . . . 114
A.56.Botón de editar usuario desbloqueado. . . . 114

Lista de Tablas

3.1. Planificación inicial 16
3.2. Resumen de las historias de usuario 18
3.3. Resumen del presupuesto del proyecto 19

4.1. Calificación de riesgos 23
4.2. RG-01 23
4.3. RG-02 23
4.4. RG-03 23
4.5. RG-04 24
4.6. RG-05 24
4.7. RG-06 24
4.8. RG-07 24
4.9. RG-08 25
4.10. RG-09 25

6.1. Tareas establecidas para el sprint 0 38
6.2. Tareas establecidas para el sprint 1 39
6.3. Tareas establecidas para el sprint 2 40
6.4. Tareas establecidas para el sprint 3 40
6.5. Tareas establecidas para el sprint 5 41
6.6. Tareas establecidas para el sprint 6 42
6.7. Tareas establecidas para el sprint 7 43
6.8. Tareas establecidas para el sprint 8 44
6.9. Planificación extendida 44

6.10. Tareas establecidas para el sprint 9 45

6.11. Tareas establecidas para el sprint 10 46

6.12. Tareas establecidas para el sprint 11 47

6.13. Tareas establecidas para el sprint 12 48

6.14. Tareas establecidas para el sprint 13 49

Capítulo 1

Introducción

1.1. Introducción

El fútbol es uno de los deportes más importantes no solo de España sino de gran parte del mundo y no solo funciona como deporte, sino que también se trata de uno de los medios de entretenimiento más importantes de nuestro país [1] siendo capaz de apoderarse de más de el 50 % de cuota de pantalla cada vez que se produce un encuentro importante [2].

En los últimos años se ha podido observar un desplazamiento de este público principalmente televisivo a nuevos medios alojados en internet. Esto plantea la oportunidad de construir una aplicación albergada en el mismo lugar hacia el que se está desplazando el público objetivo de este medio de entretenimiento, facilitando, de esta manera, el acceso a información actualizada sobre el mundo del deporte sin la necesidad de cambiar de dispositivo de visualización.

1.2. Motivación

Aunque he de reconocer que personalmente no soy un gran fan del fútbol soy consciente de la gran importancia que este tiene en nuestra sociedad y soy testigo de el gran interés que este deporte genera en mis círculos cercanos, por lo que considero que realizar una aplicación de este tipo puede facilitar el acceso a información de interés a un gran número de personas. Además, es un medio que facilita mucha información visual como logos, escudos, banderas, etc., lo que permite un enfoque más artístico de la aplicación que suscita mi interés.

Soy consciente de que a día de hoy existe una gran cantidad de alternativas sobre visualización de resultados deportivos, por lo que mi idea para este proyecto es aportar un granito de arena a un mundo que últimamente parece estar más interesado en publicitar las apuestas deportivas que proporcionar la información consultada.

1.3. Público Objetivo

Tras lo descrito anteriormente se puede deducir que el grupo de usuarios objetivo son todas aquellas personas interesadas en el mundo del deporte que quieran consultar información sobre sus equipos, ligas o jugadores favoritos.

Además, esto no se encuentra solo restringido al publico del fútbol como entretenimiento, sino que cualquier profesional del medio podrá valerse de la información y estadísticas presentadas por la aplicación.

Por último, se pretende crear una web con una interfaz sencilla de utilizar para que cualquier persona (sin depender de sus rangos de edad y sus conocimientos previos de este tipo de tecnologías) pueda acceder de forma sencilla a la información presentada.

1.4. Aplicaciones Similares

A día de hoy existen unas cuantas alternativas que se pueden usar para consultar información deportiva por lo que pasaré a describir solo las más relevantes:

- **resultados-futbol.com** [3]: Es la principal fuente de inspiración de este proyecto. Cuenta con gran cantidad de información futbolística muy completa y uno de los nombres más fáciles de recordar para la comunidad hispanohablante.
- **flashscore.es** [4]: Más conocida en España por su antiguo nombre "MisMarcadores.com" es una de las webs de fútbol más populares de nuestro país. Muestra tanto resultados deportivos (no solo de fútbol sino que también de otros deportes como tenis o baloncesto) como noticias.
- **marca.com** [5]: La prensa escrita al dar el paso a medios digitales también ha entrado a la competición por este nicho de mercado, centrándose más en noticias relacionadas con el mundo del deporte pero también proporcionando datos e información relevante sobre los encuentros y equipos que participan en estas competiciones.
- **whoscored.com** [6]: Se trata de una de las webs más conocidas de la comunidad angloparlante. Cuenta tanto con información deportiva (dedicada al fútbol exclusivamente) como noticias.
- **transfermarkt.com** [7]: Esta página, también de la comunidad anglófona, está especializada en la información referente a los mercados de fichajes y traspasos publicando desde rumores hasta los precios de mercado estimados para jugadores y clubes de fútbol.
- **Google Football** [8]: Por último, me gustaría destacar que el gigante de internet también cuenta con una pequeña participación en este nicho de mercado, ya que aunque no cuenta con una sección dedicada a los resultados deportivos si que muestra sus propias páginas especializadas al realizar consultas relacionadas con partidos o equipos en su buscador.

Todas las webs citadas anteriormente cuentan, en mi opinión, con un problema bastante importante que son los anuncios.

El problema no son los anuncios en si mismos, cualquier aplicación que pretenda durar un mínimo de tiempo necesita una fuente de ingresos para mantener activos sus servicios, y la forma más habitual dentro del mundo de internet es a través de anunciantes y patrocinadores. El problema con estas webs es que la mayoría de ellas se nutren tanto de la publicidad de apuestas deportivas como de versiones "premium" con información referente a estas, lo que personalmente no me gusta ya que cualquier persona (independientemente de su rango de edad) puede acceder a este tipo de páginas y visualizar estos contenidos.

1.5. Objetivos

El objetivo general del proyecto es construir una aplicación web capaz de proveer información relativa a resultados futbolísticos.

En las siguientes secciones se procederá a describir los objetivos, tanto específicos del proyecto como personales, más en detalle.

1.5.1. Objetivos específicos

Estos objetivos son un desglose del objetivo general en metas más concretas que, de alcanzarse, permitirían finalizar el proyecto de manera satisfactoria.

- Construir una aplicación web de tipo SPA (*Single-page application*).
- La aplicación deberá mostrar información de ligas, equipos, partidos, jugadores y entrenadores de manera sencilla y de fácil acceso.
- La aplicación deberá permitir a los usuarios guardar sus ligas, equipos, y partidos favoritos.
- La aplicación deberá mantener su información actualizada.

1.5.2. Objetivos personales

Respecto a los objetivos personales, dado que este proyecto se ha desarrollado en un contexto principalmente educativo, lo he utilizado para intentar desarrollar mis competencias como Ingeniero Informático.

- El principal objetivo personal del proyecto se centra en continuar mi aprendizaje sobre los frameworks de desarrollo web basados en JavaScript.
- Aprender a trabajar con APIs externas y entender el reto que supone adaptarse al trabajo realizado por otras personas ajenas a tu proyecto y tus necesidades concretas.
- Mejorar mis habilidades artísticas y todo lo referente a diseño y desarrollo de interfaces y aprender a valorar todo el trabajo que esto esconde.
- Mejorar mis capacidades respecto al desarrollo web utilizando HTML, CSS y JavaScript y conocer un poco mejor el mundo del desarrollo frontend del que no he podido disfrutar tanto como me habría gustado durante mis estudios.
- Conocer los retos que supone el desarrollo en solitario de un proyecto de este tipo y entender el tiempo y el trabajo necesario que se esconde tras ellos.
- Aprender y poner en práctica algunos de los procesos de desarrollo que se utilizan en la industria.

1.6. Estructura de la memoria

Este documento se estructura de la siguiente manera:

- **Capítulo 3 Conceptos Previos**: En el que se realizará un análisis sobre las alternativas disponibles para la creación de aplicaciones web.
- **Capítulo 3 Requisitos y planificación**: En el que se expone la planificación diseñada inicialmente para el proyecto.
- **Capítulo 5 Tecnologías utilizadas**: En el que se describen las diferentes tecnologías utilizadas a lo largo del proyecto.
- **Capítulo 4 Plan de riesgos y estimación de costes**: En el que se realizará una estimación de costes y se creará un plan de posibles riesgos durante el transcurso del proyecto.
- **Capítulo 6 Seguimiento del proyecto**: En el que se detallará el desarrollo del proyecto reflejando los eventos relevantes sucedidos a lo largo de su trayecto.
- **Capítulo 7 Diseño**: En el que se detallará el diseño del proyecto.
- **Capítulo 8 Implementación**:En el que se tratarán los detalles de la implementación del proyecto.
- **Capítulo 9 Pruebas**: En el que se explicarán las distintas pruebas que se han llevado a cabo sobre las partes software del proyecto.
- **Capítulo 10 Conclusiones y líneas futuras**: En el que se expondrán las conclusiones obtenidas de la participación en el proyecto así como posibles mejoras a lleva a cabo e un futuro.
- **Apéndices**: Que contendrá manuales de usuario, despliegue y el detalle de los archivos y repositorios utilizados durante el proyecto.

Capítulo 2

Conceptos Previos

Durante este capítulo se realizará un análisis sobre las distintas alternativas existentes para la creación de una aplicación web. Este estudio se centrará en los tres pilares fundamentales que constituyen nuestro proyecto: el *framework front-end*, el proveedor de información y el proveedor de plataforma *back-end*.

En cada sección se comenzará exponiendo los principales candidatos planteados para cada categoría y se terminará enunciando el candidato escogido junto con las razones que han llevado a dicha decisión.

Para información más detallada de las herramientas escogidas y otras herramientas utilizadas durante el proyecto consultar el capítulo 5.

2.1. Frameworks front-end

A la hora de escoger un *famework* con el que desarrollar el front-end de nuestro proyecto nos encontramos con tres principales competidores [9] [10].

2.1.1. Angular

Angular es un *framework* Javascript que cuenta con una gran cantidad de características integradas y que sigue la filosofía de tener todas las herramientas que un desarrollador necesita. Este *framework* apareció en 2010 y es considerado uno de los principales precursores del concepto moderno de *framework* Javascript. A día de hoy, aunque muy evolucionado, sigue siendo una de las soluciones más populares en lo que a desarrollo *front-end* se refiere.

Angular se basa en una estructura formada por un *template* HTML con su sintaxis de instrucciones específicas y lógica Javascript + Typescript. Esta separación se produce de manera estricta ya que, a diferencia de otras opciones, cada una de las partes debe colocarse en un fichero diferente. Posteriormente, Angular se encargará de enlazar ambos archivos para así manipular la información incluida en el *template* cuando sea necesario

Algunas de las características con las que cuenta Angular son las siguientes:

- Cuenta con enlace de datos bidireccional lo que permite que cualquier propiedad de nuestro modelo agregada a alguna entrada de nuestro documento HTML pueda ser modificada tanto por el usuario como por la aplicación.

- Incluye el patrón de inyección de dependencias que permite añadir servicios externos a los componentes generados.
- Incorpora herramientas de *routing* que permiten gestionar las rutas de la aplicación en en *front-end* en lugar de en el *back-end* como se hacía habitualmente.
- En sus versiones más modernas agrega TypeScript para incorporar funcionalidades extra.
- Su arquitectura se basa en la jerarquía de componentes donde estos son considerados una unidad formada por una directiva y un *template* que a su vez pueden llamar a otros componentes distintos.
- Introduce el concepto de CLI, una consola de comandos utilizada para generar la estructura de nuevos proyectos con facilidad.

Finalmente, comentar que Aungular cuenta con una dificultad de aprendizaje alta ya que se necesita aprender la sintaxis general de Angular junto al lenguaje Typescript para trabajar con él. También es necesario entender la forma en que el fichero Javascript + Typescript se comunica con el *template* HTML. Por último, también es necesario utilizar la herramienta CLI (Command-Line Interface) para generar un nuevo proyecto, por lo que este no se puede incorporar a cualquier proyecto ya existente.

2.1.2. React.js

A diferencia de Angular, React.js no es un *framework* en sí, sino una librería de Javascript que gestiona el renderizado. De esta manera, algunas características como el *routing*, que si están disponibles en otros frameworks, no vienen incluidas en el paquete principal de React, siendo necesario añadirlas en el caso de que el proyecto las requiera. Otra propiedad característica de React es que no define un flujo de trabajo, es decir, proporciona métodos y recomendaciones para realizar el renderizado sin obligar a sus usuarios a trabajar con ellas.

Al contar con su propio motor de renderizado, React también puede utilizar una sintaxis propia conocida como JSX, que cuenta con una estructura similar a HTML pero con extensiones basadas en Javascript. Esto implica que los ficheros fuente de React son ficheros Javascript que incorporan fragmentos de JSX, mientras que en Angular los documentos se estructuran realizando una separación entre HTML y Javascript.

Otra importante característica de React es que permite la detección de cambios implementable por componente, lo que permite a sus usuarios gestionar bajo que condiciones se debe producir un nuevo renderizado de cada componente.

Por último, aunque puede parecer sencillo en un inicio, React puede ser complejo de aprender debido a JSX y los nuevos conceptos y herramientas que proporciona. También cuenta con la dificultad de que, aunque es posible incorporarlo a un proyecto existente, algunas de sus funciones como JSX necesitan un nuevo proyecto para funcionar. Por suerte, al igual que Angular, React incorpora su propio CLI para facilitar la creación de nuevos proyectos.

2.1.3. Vue.js

Se podría considerar que Vue.js es un *framework* que se sitúa en el medio de Angular y React, ya que si incorpora un flujo de trabajo definido y algunas funciones integradas, pero no tantas como Angular, manteniendo así un peso más ligero. Al igual que React, algunas características importantes como el *routing* y la gestión de estados no se encuentran incluidas en el paquete principal pero pueden instalarse fácilmente a partir de su herramienta CLI.

Vue utiliza una estructura formada por Javascript puro y un template HTML casi puro que se conectan para que la lógica Javascript pueda modificar dicho template dinámicamente. A diferencia de Angular, estas dos estructuras si se pueden encontrar juntas en el mismo fichero en lo que se conoce como SFC (Single File Component).

A diferencia de otros frameworks Vue es incremental, es decir, no necesita crear un proyecto exclusivo para poder usarlo y puede ser incluido en proyectos ya existentes lo que le convierte en la opción ideal si lo que se busca es modernizar un proyecto antiguo.

Otra característica importante es que se trata de un *framework* progresivo. Vue se concibió con el objetivo de crear una solución ligera que soportara las funciones de gestión del renderizado y componentes. Sin embargo, también es posible incorporar nuevas funcionalidades de manera sencilla a partir de un sistema de *plugins*.

Finalmente, Vue es considerado como uno de los *frameworks* más fáciles de aprender de la actualidad. Al utilizar Javascript puro como lenguaje base (también es posible utilizar Typescript lo que facilita el acceso a personas con conocimientos previos sobre Angular) junto con HTML, solo es necesario aprender las características especiales del template de Vue para empezar a usarlo. Aunque no es necesario crear un nuevo proyecto para utilizarlo, Vue también proporciona una herramienta CLI que facilita la creación e incorporación de los *plugins* necesarios para el desarrollo.

2.1.4. Framework front-end escogido

De entre los tres candidatos se ha escogido Vue.js como la opción que mejor se adapta a las necesidades de nuestro proyecto. Esta decisión se debe a que Vue es la mezcla perfecta de las otras dos alternativas disponibles, ya que cuenta con gran parte de las características de Angular mientras que mantiene el peso ligero y extensibilidad de React. Además, Vue cuenta con un tiempo de renderizado menor que las otras alternativas y su sintaxis basada en Javascript y HTML facilitará el aprendizaje de esta tecnología.

Con el objetivo de hacer la comparativa entre los distintos *frameworks* más sencilla se ha decidido trasladar las explicaciones más detalladas de algunas de las características técnicas de este a la sección 5.1.3.

2.2. Proveedor de información

Aunque existe una gran cantidad de alternativas entre las que escoger, las necesidades concretas de nuestro proyecto limitan en gran medida cuales de ellas pueden adaptarse a nuestra aplicación.

2.2.1. Besoccer

Besoccer [11] es una de las APIs más populares del mercado y la proveedora de información de algunas de las páginas más importantes del sector. Cuenta con una gran base de datos que agrupa estadísticas e histórico de partidos desde el año 1990.

Algunas de sus características más importantes son las siguientes:

- Resultados en tiempo real con un tiempo de respuesta inferior a otras alternativas.
- Comentarios de partidos en tiempo real.
- Información detallada de partidos, equipos y jugadores.

- Gran cantidad de ligas y competiciones disponibles.
- No incluye versión gratuita y el plan de pago más barato está muy limitado.

2.2.2. football-data.org

Football-data.org [12] es una API que destaca por tener un plan gratuito vitalicio con la tasa de peticiones más alta de todas las alternativas.

Algunas de sus características más importantes son las siguientes:

- Plan gratuito de por vida.
- El plan gratuito permite acceso a 12 competiciones distintas.
- Los resultados del plan gratuito se actualizan con retraso.
- El plan gratuito permite acceso a partidos, calendarios y clasificaciones.
- El plan gratuito tiene un límite de 10 peticiones por minuto.

2.2.3. TodoPorElFutbol.com

TodoPorElFutbol.com [13] es una API con gran cantidad de información y soporte para distintos idiomas. Ofrece un plan gratuito pero con restricciones significativas.

Algunas de sus características más importantes son las siguientes:

- Fácil integración con soporte para JSON y XML.
- Cobertura sobre más de 800 competiciones de 120 países.
- Gran cantidad de información sobre encuentros, resultados, estadísticas, etc.
- Resultados en tiempo real.
- Disponibles 6 idiomas distintos.
- Plan gratuito con una única competición y limitado a 10 peticiones por hora.

2.2.4. API-Football

API-Footbal [14] es una de las mejores opciones disponibles del mercado gracias a su plan gratuito que permite acceso a todo el contenido de la API con un límite de 100 peticiones diarias.

Algunas de sus características más importantes son las siguientes:

- Cobertura de casi 900 competiciones.
- Fácil integración.
- Soporte las 24h.

- Resultados en tiempo real para todos los planes.
- Varios años de datos disponibles.
- Plan gratuito con acceso a toda la API y un límite de 100 peticiones diarias.

2.2.5. Proveedor de información escogido

Según lo expuesto anteriormente, la API que mejor se adapta a las necesidades de nuestro proyecto es API-Football. Esto se debe a su plan gratuito que permite acceso a toda la colección de datos completa y con un límite de peticiones diario (en lugar de los limites por hora que establecen otras alternativas) que se adecua mejor a las restricciones de nuestro proyecto.

En la sección 5.1.2 se comentarán más en detalle las características técnicas de la API.

2.3. Plataforma back-end

Por último, también se han investigado distintas plataformas BaaS (Backend as a Service) necesarias para construir y gestionar el *back-end* de nuestro proyecto [15] [16].

2.3.1. Firebase

Firebase [17] es una plataforma de tipo BaaS utilizada para facilitar el desarrollo de aplicaciones web y móvil. La principal intención de esta plataforma es agrupar todas las herramientas de gestión necesarias para un proyecto en un mismo sitio, evitando así a los desarrolladores dedicarle demasiado tiempo al *back-end*.

Firebase proporciona gran cantidad de herramientas que se pueden agrupar en cuatro grupos distintos: desarrollo, crecimiento, monetización y análisis. Nosotros solo tendremos en cuenta las herramientas de desarrollo puesto que son las más importantes para nuestros requisitos.

Entre las herramientas de desarrollo que proporciona Firebase nos encontramos con las siguientes:

- Servicio de base de datos en tiempo real.
- Autenticación de usuarios.
- Almacenamiento en la nube.
- Notificaciones de fallos.
- Configuración remota para modificar ciertos aspectos de la aplicación sin necesidad de actualizarla.
- Servicio de *hosting*.

Proporciona un plan de precios gratuito con características más que suficientes para las necesidades de este proyecto, ofreciendo, además, escalabilidad en función de las necesidades del proyecto.

2.3.2. Back4App

Back4App [18], al igual que Firebase, es otra solución BaaS diseñada para la creación de aplicaciones de manera rápida y sencilla. Esta plataforma proporciona la gran mayoría de herramientas que proporciona Firebase pero con algunos añadidos significativos como el soporte para APIs GraphQL. Otra característica importante de Back4App es que, a diferencia de Firebase, esta es una solución de código abierto, lo que permite a los desarrolladores conocer el funcionamiento interno del servicio.

Algunas de las principales herramientas que nos proporciona son las siguientes:

- Modelo de datos + APIs GraphQL.
- Base de datos colaborativa y en tiempo real.
- Nube privada.
- Autenticación de dos factores.
- Verificación a través de correo electrónico.
- Panel de administración de datos.

Al igual que Firebase, ofrece un plan de precios gratuito bastante generoso y fácilmente escalable que es más que suficiente para las necesidades de nuestro proyecto.

2.3.3. Parse

Parse [19] es una de las plataformas BaaS líderes del mercado. Cuenta con el respaldo de Facebook y, aunque no ofrece alojamiento, como sí hacen otros competidores, brinda una gran cantidad de características y herramientas que pueden ser útiles a la hora de desarrollar y administrar los servicios *back-end* de cualquier aplicación. También destaca por ser una de las mejores opciones para el desarrollo de *middleware* gracias a su facilidad de uso.

Algunas de sus principales características son las siguientes:

- Base de datos en tiempo real.
- Integración con redes sociales.
- Servicio de autenticación.
- Funciones de seguridad.
- Sistema de gestión de usuarios.

De todas las opciones citadas y aun siendo una plataforma de código abierto, esta es la única que no ofrece un plan gratuito, ya que al no proporcionar opciones de almacenamiento, es necesario alojarla en un servidor de pago.

2.3.4. AWS Amplify

AWS Amplify [20] es la propuesta de Amazon para soluciones BaaS construida sobre AWS (Amazon Web Services). AWS Amplify está diseñado para proveer un conjunto de herramientas y características para que los desarrolladores *front-end* puedan crear sus servicios de *back-end* de manera rápida y sencilla.

Algunas de sus principales características son las siguientes:

- Base de datos en tiempo real.
- Servicios para la creación de APIs.
- Geolocalización.
- Notificaciones *push*.

Al igual que Firebase y Back4App, esta propuesta ofrece alojamiento web gratuito de un año de duración y un plan de pago por uso.

2.3.5. Plataforma back-end escogida

Finalmente, se ha decidido utilizar Firebase como plataforma *back-end* para nuestro proyecto por ser una de las soluciones BaaS más populares lo que la hace estar muy bien documentada. Además, cuenta con herramientas muy útiles, como el servicio de autenticación, que facilitaran el desarrollo de algunas de las funcionalidades planteadas para nuestro proyecto. Por último, ofrece un plan gratuito lo suficientemente abierto como para no necesitar preocuparnos por el límite de peticiones ni de almacenamiento.

En la sección 5.1.4 se comentarán las herramientas proporcionadas por esta plataforma que han sido utilizadas en el proyecto.

Capítulo 3

Planificación y presupuesto del proyecto

3.1. Scrum

3.1.1. ¿Qué es Scrum?

Scrum [21] [22] es un marco de trabajo desarrollado a principios de los años 90 utilizado para el desarrollo y mantenimiento de productos complejos siguiendo un proceso de desarrollo ágil. Este se basa en la teoría de que el desarrollo de proyectos sigue un planteamiento empírico, es decir, acepta que el equipo de trabajo no sabe todo al inicio del proyecto y este evolucionará a medida que avance el mismo.

Scrum se basa en tres pilares esenciales: transparencia, inspección y adaptación.

- **Transparencia**: Todos los aspectos significativos del proceso tienen que ser visibles y entendibles para todos los responsables del resultado. Esto implica que dichos aspectos deben ser definidos a través de un estándar común que facilite el entendimiento entre todas las partes involucradas.
- **Inspección**: Los usuarios de Scrum deben llevar a cabo revisiones frecuentes del progreso con el objetivo de detectar posibles variaciones no deseadas. Es importante que estas inspecciones no sean tan frecuentes como para que puedan afectar a las pautas de trabajo.
- **Adaptación**: En el caso de detectar desviaciones importantes en el producto es necesario realizar acciones con el objetivo de ajustar dichas desviaciones. Estas acciones deben efectuarse cuanto antes para intentar evitar desviaciones mayores.

3.1.2. El Equipo Scrum (*Scrum Team*)

Los equipos scrum son auto-organizados y multifuncionales, es decir, no son dirigidos por nadie externo lo que les permite elegir la forma de llevar a cabo su trabajo y están formados por especialistas de distintas disciplinas por lo que constan de todas las competencias necesarias para efectuar el trabajo sin depender de personas externas al proyecto.

El Equipo Scrum está compuesto por tres roles: el Dueño de Producto (*Product Owner*), el Equipo de Desarrollo (*Development Team*) y un Experto en Scrum *(Scrum Master)*.

- **Dueño de Producto (*Product Owner*)**: Es el encargado de maximizar el valor del producto y el trabajo del Equipo de Desarrollo. Es también el encargado de gestionar la Lista del Producto (*Product Backlog*). Para que este pueda hacer bien su trabajo, toda la organización debe respetar sus decisiones.
- **Equipo de Desarrollo (*Development Team*)**: El Equipo de Desarrollo está compuesto por el conjunto de profesionales que se va a encargar de desarrollar y entregar los incrementos. La organización es la encargada de estructurar y empoderar a los Equipos de Desarrollo para que estos puedan ejecutar su trabajo de manera independiente. El tamaño de este debe ser lo suficientemente pequeño como para permanecer ágil (los equipos de desarrollo grandes requieren de mayor coordinación lo que genera mayor complejidad) y lo suficientemente grande como para que pueda completar cantidades de trabajo suficientemente significativas (los equipos de desarrollo demasiado pequeños reducen la productividad y pueden encontrar limitaciones en cuanto a las habilidades necesarias para entregar incrementos).
- **Experto en Scrum *(Scrum Master)***: Es el líder del Equipo Scrum y el encargado de que Scrum se entienda y se ponga en practica. Ayuda a personas externas al equipo a comprender que interacciones pueden ser útiles de manera que se modifiquen dichas interacciones con el fin de maximizar el valor creado por el equipo.

3.1.3. Eventos Scrum

Scrum consta de eventos predefinidos con el fin de crear regularidad y minimizar la necesidad de realizar reuniones externas a la planificación. Estos eventos son bloques de tiempo (*time-boxes*) de tamaño fijo que se desempeñan de forma periódica a lo largo de las distintas etapas del proyecto. Estos eventos han sido diseñados de manera que habiliten los pilares esenciales de Scrum y constituyen una oportunidad para transformar sus artefactos.

- **Sprint**: Se trata del evento principal de Scrum. Es un bloque de tiempo no superior a un mes en el que se crea un incremento de producto utilizable y potencialmente desplegable. Es recomendable que la duración de estos sea fija y esté definida por el Equipo de Desarrollo o el Scrum Master. Cada Sprint debe comenzar inmediatamente después de la finalización del Sprint anterior. Durante el desarrollo de un Sprint se deben evitar los cambios que puedan modificar el Objetivo del Sprint o sus objetivos de calidad aunque si es posible modificar su alcance.
- **Planificación de Sprint (*Sprint Planning*)**: Durante este evento se concreta la planificación del Sprint. En el se establecen los objetivos y los desarrollos a llevar a cabo durante el incremento. Durante este evento también se estima la carga de trabajo de cada tarea. Su duración máxima es de 8 horas para un Sprint de un mes.
- **Objetivo del Sprint *(Sprint Goal)***: El Objetivo del Sprint es la meta establecida durante la planificación para que el incremento pueda terminarse de manera satisfactoria. Sirve como guía al Equipo de Desarrollo para formar una idea de sobre que se está trabajando en el incremento. A medida que el Equipo de desarrollo avanza debe mantener el objetivo en mente, implementando la funcionalidad necesaria para que se pueda satisfacer dicho objetivo.
- **Scrum Diario *(Daily Scrum)***: Es una reunión de aproximadamente 15 minutos durante la cual se resumen las actividades realizadas durante las 24 horas anteriores y se crea un plan de trabajo para las próximas 24 horas. El Scrum diario se utiliza para evaluar el progreso de los equipos de desarrollo y sincronizar su trabajo.
- **Revisión de Sprint *(Sprint Review)***: Al final de cada Sprint se lleva a cabo una revisión del incremento ejecutado para adaptar la Lista de Producto si fuera necesario. El objetivo de esta revisión es facilitar la retroalimentación de información y fomentar la colaboración. Esta evento está constituido por una reunión de una duración no superior a las 4 horas.

- **Retrospectiva de Sprint *(Sprint Retrospective)***: La retrospectiva del Sprint es una oportunidad para el Equipo Scrum de evaluar el trabajo desempeñado así como los problemas surgidos para generar un plan de mejoras que pueda optimizar el trabajo durante próximos Sprints.

3.1.4. Artefactos Scrum

Además de los eventos, Scrum también cuenta con una serie de artefactos diseñados específicamente para asegurar que todos los interesados tengan la misma idea del artefacto. Estos sirven para representar trabajo o valor y proporcionar transparencia y oportunidades de inspección y adaptación dentro del proyecto.

- **Lista de Producto (*Product Backlog*)**: La Lista de Producto es una lista que comprende todos los elementos necesarios para materializar el producto, es decir, es la fuente de requisitos del proyecto. Esta lista nunca va a estar completa ya que puede evolucionar al mismo tiempo que el producto y el equipo lo hacen. Esto es debido a que a medida que se van entregando incrementos del producto, el mercado proporciona retroalimentación que puede modificar los requisitos ideados inicialmente.
- **Lista de Pendientes del Sprint (*Sprint Backlog*)**: Es el conjunto de elementos seleccionados de la Lista de Producto que se van a realizar durante el Sprint. Esta lista representa el trabajo necesario que hay que desempeñar para alcanzar el objetivo del Sprint de manera satisfactoria. Los elementos de la lista deben tener un nivel de detalle lo suficientemente alto como para que los avances puedan ser entendidos durante el Scrum Diario. Al igual que la Lista de Producto esta puede evolucionar a lo largo del Sprint añadiendo nuevos elementos según se requiera nuevo trabajo.
- **Incremento**: El incremento es la suma de todos los elementos completados de la Lista de Producto a lo largo de la duración de un Sprint. Al final de un Sprint el incremento debe estar terminado y en condiciones de utilizarse.

3.1.5. Historias de Usuario

Una historia de usuario es una explicación general e informal de una funcionalidad software escrita desde el punto de vista del usuario final. Su propósito es mostrar como la inclusión de dicha funcionalidad puede aportar valor al cliente [23].

Las historias de usuario se adaptan perfectamente a Scrum porque aportan una forma rápida de administrar los requisitos proporcionados por los usuarios sin tener la necesidad que generar una gran cantidad de documentos facilitando a su vez la respuesta ante requisitos cambiantes [24].

3.2. Adaptación de Scrum al proyecto

Para adaptar este proyecto al marco de trabajo Scrum primero hay que tener en cuenta que se trata de un Trabajo de Fin de Grado, lo que significa que, teniendo en cuenta lo expuesto en el documento sobre el Proyecto docente del TFG [25], este proyecto debería tener una duración de unas 300 horas de trabajo. Dado que la duración estimada de los Sprints va a ser de 2 semanas y que las jornadas de trabajo van a tener una duración de 4 horas diarias da lugar a una estimación total de 8 Sprints.

Debido a la situación actual y que la mayor parte del desarrollo se ha llevado en solitario el propio alumno se ha encargado de interpretar los tres distintos roles del Equipo Scrum (*Product Owner*, *Development Team* y *Scrum*

Master). Esto no es del todo recomendable y como veremos en el apartado 3.2.3 ha llevado a algunos problemas durante el desarrollo. Aun teniendo en cuenta que la misma persona ha sido la encargada de llevar a cabo los 3 distintos roles si que se ha intentado mantener la esencia de Scrum a modo de simulación estableciendo metas para cada Sprint, realizando diariamente pequeñas planificaciones de que partes del proyecto se debería tener finalizadas al final del día a modo de *Daily Scrum* y efectuando revisiones de lo que se tiene al final de cada Sprint a modo de *Sprint Review* y *Sprint Retrospective*.

Para la lista de producto (*Product Backlog*) se han utilizado tanto Trello como el planificador de *issues* de GitLab.

También cabe destacar que, al tratarse de una tecnología totalmente nueva se ha realizado un Sprint inicial (Sprint 0) para llevar a cabo tareas de documentación básica sobre la tecnología que se va a utilizar y generar la estructura inicial del proyecto. De esta manera se pretende conseguir separar los Sprints dedicados al desarrollo del resto de tareas realizadas.

3.2.1. Planificación inicial de los sprints

Como se ha comentado anteriormente, se ha estimado un total de 8 Sprints (9 contando el Sprint inicial) para jornadas de trabajo de 4 horas y una duración del Sprint de 2 semanas, lo que nos deja con la planificación inicial planteada mostrada en la tabla 3.1. En ella se encuentra detallada la fecha de inicio y final planteada para cada Sprint así como una enumeración de todos los Sprints planeados inicialmente.

Sprint	**Comienzo**	**Finalización**	**Observaciones**
Sprint 0	01/03/2021	15/03/2021	Sprint Inicial
Sprint 1	15/03/2021	29/03/2021	
Sprint 2	29/03/2021	12/04/2021	
Sprint 3	12/04/2021	26/04/2021	
Sprint 4	26/04/2021	10/05/2021	
Sprint 5	10/05/2021	24/05/2021	
Sprint 6	24/05/2021	07/06/2021	
Sprint 7	07/06/2021	21/06/2021	
Sprint 8	21/06/2021	05/07/2021	

Tabla 3.1: Planificación inicial

Como veremos en la sección 3.2.3, esta planificación no es fija y puede variar a medida que el proyecto avance. Es posible que se aumente la carga de trabajo en determinados Sprints o que aumente el numero de Sprints en caso de que el tiempo de trabajo estimado aumente o si se considera necesario añadir nueva funcionalidad.

3.2.2. Descripción de historias de usuario

En esta sección listaremos un pequeño resumen (disponible en la tabla 3.2) de las historias de usuario planteadas para el proyecto. Se puede obtener información más detallada de estas consultando el Capítulo 6 en el que se realizará un seguimiento del proyecto.

3.2.3. Modificaciones posteriores de la planificación inicial

Como he comentado anteriormente, debido a una serie de problemas y retrasos durante la etapa de desarrollo, la planificación final terminó desviándose de la planificación planteada en un principio. Gran parte de la funcionalidad original fue desechada y el numero de Sprints necesarios aumentó. Esto se encuentra más detallado en el seguimiento del proyecto en el Capítulo 6.

Número	**Título**	**Descripción**
1	Consultar una liga	Como usuario quiero consultar la información disponible de una liga.
2	Consultar un equipo	Como usuario quiero consultar la información disponible de un equipo.
3	Consultar un partido	Como usuario quiero consultar la información disponible de un partido.
4	Consultar un jugador	Como usuario quiero consultar la información disponible de un jugador.
5	Consultar un entrenador	Como usuario quiero consultar la información disponible de un entrenador.
6	Consultar un país	Como usuario quiero consultar la información disponible de un país.
7	Consultar información según la temporada	Como usuario quiero consultar la información disponible de una liga, equipo o jugador para una temporada determinada.
8	*SignUp*	Como usuario quiero registrarme en la web para acceder a todas las funcionalidades disponibles.
9	*LogIn*	Como usuario registrado quiero iniciar sesión para acceder a todas las funcionalidades disponibles.
10	*LogOut*	Como usuario logueado quiero cerrar sesión para borrar mi información del SessionStorage.
11	Consultar perfil	Como usuario logueado quiero consultar mi perfil.
12	Editar perfil	Como usuario logueado quiero editar mi perfil.
13	Guardar liga	Como usuario logueado quiero guardar una liga.
14	Guardar equipo	Como usuario logueado quiero guardar un equipo.
15	Guardar partido	Como usuario logueado quiero guardar un partido.
16	Borrar liga guardada	Como usuario logueado quiero borrar una liga guardada.
17	Borrar equipo guardado	Como usuario logueado quiero borrar un equipo guardado.
18	Borrar partido guardado	Como usuario logueado quiero borrar un partido guardado.
19	Cambiar idioma	Como usuario quiero cambiar el idioma de los textos para favorecer el entendimiento de los datos de la web.

Tabla 3.2: Resumen de las historias de usuario

3.3. Planificación del presupuesto

Para realizar una estimación inicial de los costes de nuestro proyecto, lo primero que se ha hecho es consultar el sueldo medio de un programador decretado por el Boletín Oficial del Estado (BOE) [26]. En este se establece que para el área de Consultoría, Desarrollo y Sistemas un Analista programador, Diseñador de páginas web (Grupo III) debe cobrar un salario base de 21.555,66 € anuales al que sumándole un "plus convenio" de 1.438,08 € nos dejaría con un salario total de 22.993,74 €. Una vez calculado el salario también hay que tener en cuenta la contribución de la empresa a la Seguridad Social que viene a ser aproximadamente de un 30 % del sueldo bruto por cada trabajador [27], lo que en nuestro caso supondría un gasto total entre sueldo y Seguridad Social de 29.891,862 € anuales por trabajador.

Una vez calculado el coste anual total por trabajador podemos calcular el coste total de los trabajadores para la duración total del proyecto. A partir de los cálculos realizados anteriormente y ajustándonos a la estimación de que un trabajador puede ejercer un máximo de 1800 horas de trabajo anuales obtenemos que el costo por hora de cada trabajador es de 16,60 €. Por último, teniendo en cuenta que vamos a tener un único desarrollador trabajando en el proyecto durante aproximadamente 300 horas tenemos que el coste total de los trabajadores para la completa duración del proyecto será de aproximadamente 4.981,97 €.

Debido a que este proyecto se ha realizado durante la actual pandemia del COVID-19 también debemos tener en cuenta costes por teletrabajo. Referenciando de nuevo al BOE, todo empleado cuenta con derechos relativos a la dotación y mantenimiento de medios y al abono y compensación de gastos [28], por lo que, teniendo esto en cuenta, debemos añadir también el precio de equipos y gastos fijos. Aunque no se ha realizado un calculo del total de los gastos podemos utilizar como referencia los complementos salariales debidos al teletrabajo establecidos por otras empresas (como el caso de BP que aporta 6.407,52 € brutos anuales[29]) para estimar un total de 1.067,92 € adicionales por cada trabajador. El precio aproximado del material informático utilizado sería de unos 1500 € adicionales.

Por último, aunque la gran mayoría de herramientas utilizadas son de uso gratuito, tenemos el caso de Firebase y la API de fútbol que podrían suponer un costo adicional en el caso de superar la tasa de peticiones necesarias.

Tipo de coste	**Precio (€)**
Herramientas	0 € (Susceptible a cambios)
Costes teletrabajo	1.067,92 €
Equipo de desarrollo	4.981,97 €
Material Informático	1500 €
Total	7.549,89 €

Tabla 3.3: Resumen del presupuesto del proyecto

Capítulo 4

Plan de riesgos

4.1. Plan de riesgos

Cualquier proyecto que necesita un plan de acciones frente a los posibles riesgos que surjan durante el desarrollo. En este capítulo se detallará que se entiende por riesgo y posteriormente se procederá a realizar un listado de todos los posibles riesgos que deberemos afrontar durante el desarrollo del proyecto.

4.1.1. ¿Qué es un riesgo?

Antes de comenzar a describir los riesgos encontrados en nuestro proyecto primero debemos definir lo que significa para nosotros riesgo y los distintos tipos de riesgo que podemos encontrar.

Entendemos por riesgo un "evento o condición incierto que de ocurrir, tendrá efectos tanto positivos como negativos en los objetivos de nuestro proyecto"[30]. Teniendo en cuenta esta definición podremos agrupar los riesgos dependiendo de varios factores [31].

- Agrupados según su impacto:
  - **Riesgos Positivos**: También conocidos como oportunidad, son los riesgos que en si mismos no suponen algo negativo y que de ser abordados de la manera correcta pueden aportar un mayor valor al proyecto (Un ejemplo de esto podría ser una caída de servidores debida a una sobre demanda del servicio que, de ser abordada correctamente podría suponer una oportunidad para aumentar la popularidad de nuestro proyecto).
  - **Riesgos Negativos**: Son aquellos riesgos que de producirse podrían tener un efecto negativo en el desarrollo del proyecto y que de no ser abordados correctamente podrían implicar retrasos en el proyecto o que este no alcance adecuadamente los objetivos esperados.
- Agrupados según su tipo:
  - **Riesgos de proyecto**: Son aquellos riesgos relacionados con el propio proyecto (Por ejemplo no alcanzar los objetivos planteados de manera satisfactoria).
  - **Riesgos de negocio**: Son los riesgos relacionados con la parte económica del proyecto (Por ejemplo no alcanzar la popularidad esperada para el proyecto).

4.1.2. Como actuar ante los riesgos

A la hora de crear un plan de acciones frente a un riesgo determinado se puede actuar de distinta forma dependiendo de el tipo de impacto que este suponga.

- **Riesgos Negativos**:
  - Evitar el riesgo y tratar de eliminar la amenaza protegiendo al proyecto de su impacto.
  - Transferir el impacto del riesgo a un tercero e intentar buscar una solución en conjunto.
  - Mitigar el impacto o la probabilidad de que suceda el riesgo.
  - Aceptar el riesgo y no tomar acciones a menos que este se produzca.
- **Riesgos Positivos**:
  - Aprovechar la oportunidad asegurándose de que se cumplen sus efectos positivos.
  - Mejorar el riesgo aumentando la probabilidad de su impacto.
  - Compartir la oportunidad con un tercero para aumentar la probabilidad de capturar la oportunidad.
  - Aceptar la oportunidad si se presenta pero sin enfocar el proyecto buscando su ocurrencia.

4.1.3. Como crear un plan de riesgos

Las principales etapas necesarias para llevar a cabo un plan de riesgos efectivo son las siguientes[32]:

1. Identificar los riesgos.
2. Analizar dichos riesgos y organizarlos según su prioridad.
3. Planificar las acciones frente a los riesgos.
4. Monitorizar y controlar los riesgos encontrados.

4.1.4. Riesgos encontrados en el proyecto

Durante la iteración inicial del proyecto (Sprint 0) se llevó a cabo un estudio con el fin de encontrar todos los posibles riesgos que podríamos afrontar durante el desarrollo del proyecto. La totalidad de los riesgos abarcados por este estudio se encuentra enmarcada en la categoría Riesgos de Proyecto, ya que este no se ha concebido como un proyecto comercial.

Con el fin de facilitar la comprensión de los riesgos se ha creado la tabla 4.1 donde se proporciona un valor numérico tanto a la probabilidad de los riesgos como a su impacto.

Una vez establecido el valor numérico de la probabilidad e impacto de los riesgos se mostrarán los resultados del estudio comprendido entre las tablas 4.2 y 4.10.

Valor	Probabilidad	Impacto
1	Muy poco probable	Muy leve
2	Poco probable	Leve
3	Probabilidad media	Medio
4	Probable	Alto
5	Muy probable	Critico

Tabla 4.1: Calificación de riesgos

Título	Falta de formación
Descripción	Algún miembro del equipo de desarrollo no cuenta con la suficiente formación en el uso de alguna tecnología
Probabilidad	3
Impacto	3
Plan de mitigación	Elegir tecnologías mejor documentadas para reducir los tiempos de aprendizaje
Plan de contingencia	Consultar a alguien con más conocimientos sobre la tecnología

Tabla 4.2: RG-01

Título	Alguna de las tecnologías utilizadas evoluciona
Descripción	Alguna de las tecnologías utilizadas evoluciona mejorando algunos aspectos de sus versiones anteriores
Probabilidad	1
Impacto	3
Plan de mitigación	Escoger las versiones más estables de las tecnologías utilizadas
Plan de contingencia	Cambiar a la nueva versión de la tecnología si esta ofrece muchas ventajas

Tabla 4.3: RG-02

Título	Enfermedad de algún miembro del equipo de desarrollo
Descripción	Algún miembro del equipo de desarrollo cae enfermo y no puede asistir a alguna jornada de trabajo
Probabilidad	2
Impacto	4
Plan de mitigación	Planificar los Sprints de manera que exista cierto tipo de margen para llevar a cabo tareas retrasadas
Plan de contingencia	Contar con tiempo extra al finalizar el proyecto por si es necesario añadir nuevos Sprints

Tabla 4.4: RG-03

Título	Ausencia de algún desarrollador por causas relativas al teletrabajo
Descripción	Algún miembro del equipo de desarrollo no puede asistir a alguna jornada de trabajo
Probabilidad	2
Impacto	4
Plan de mitigación	Planificar los Sprints de manera que exista cierto tipo de margen para llevar a cabo tareas retrasadas
Plan de contingencia	Contar con tiempo extra al finalizar el proyecto por si es necesario añadir nuevos Sprints

Tabla 4.5: RG-04

Título	Retraso en una tarea
Descripción	El tiempo estimado para una tarea es demasiado optimista y lleva más tiempo del esperado
Probabilidad	3
Impacto	3
Plan de mitigación	Planificar los Sprints de manera que exista cierto tipo de margen para llevar a cabo tareas retrasadas
Plan de contingencia	Contar con tiempo extra al finalizar el proyecto por si es necesario añadir nuevos Sprints

Tabla 4.6: RG-05

Título	El proyecto se alarga más de lo esperado
Descripción	El tiempo de desarrollo y el numero de Sprints se alarga más de lo esperado
Probabilidad	2
Impacto	4
Plan de mitigación	Priorizar las tareas esenciales para el desarrollo del proyecto
Plan de contingencia	Recortar funcionalidad extra de menor valor

Tabla 4.7: RG-06

Título	Cambios en los requisitos del proyecto
Descripción	Los requisitos del proyecto cambian siendo necesario implementar nueva funcionalidad
Probabilidad	3
Impacto	4
Plan de mitigación	Utilizar un marco de desarrollo ágil para facilitar la incorporación de nuevas tareas
Plan de contingencia	Re-planificar los Sprints para añadir las nuevas tareas

Tabla 4.8: RG-07

Título	Desconexión del servicio de Back-end
Descripción	El servicio de Back-end deja de funcionar debido a una limitación del número de accesos o una caída del servicio
Probabilidad	1
Impacto	4
Plan de mitigación	Utilizar un servicio de back-end ampliable que permita gran cantidad de peticiones
Plan de contingencia	Comprar una ampliación del servicio que permita más peticiones de las contratadas actualmente

Tabla 4.9: RG-08

Título	Falta de peticiones en la API
Descripción	Los recursos de la API dejan de funcionar durante una jornada
Probabilidad	3
Impacto	4
Plan de mitigación	Configurar un servidor mock complementario que simule las peticiones a la API para utilizarlo de respaldo
Plan de contingencia	Cambiar la tarea actual por una que no requiera del uso de la API

Tabla 4.10: RG-09

Capítulo 5

Tecnologías utilizadas

En este capítulo hablaremos sobre las distintas herramientas utilizadas durante el desarrollo del proyecto. Explicaremos de forma breve el funcionamiento de estas herramientas intentando resumir todos los conocimientos adquiridos sobre ellas a lo largo de la elaboración del proyecto. También hablaremos de algunas de las herramientas que fueron planteadas para su uso pero que por diversos motivos fueron descartadas en la versión final de la aplicación.

5.1. Herramientas utilizadas

A continuación pasaremos a hablar de todo el instrumental utilizado que aparece en la versión final de la aplicación así como algunas herramientas que aunque no se perciben en el incremento final consideramos indispensables para el desarrollo de manera satisfactoria toda la funcionalidad de la aplicación.

5.1.1. Git

Git [33] es una herramienta de control de versiones de tipo distribuido y de código abierto diseñada en 2005 por Linus Torvalds. Es una de las herramientas de control de versiones más utilizada en la actualidad por su facilidad de uso y su eficiencia.

Entendemos por sistema de control de versiones (VCS por sus siglas en inglés) una herramienta que nos permite guardar los cambios realizados sobre un proyecto software a lo largo del tiempo de manera que se puedan recuperar distintas versiones específicas del mismo más adelante. Para ello, Git organiza los proyectos en repositorios de código que contienen todos los archivos y cambios que se han realizado sobre el software. Estos repositorios de código se dividen en ramas que permiten separar conjuntos de cambios (por ejemplo agrupándolos según la tarea) de la funcionalidad central posibilitando volver a integrarlos una vez se haya comprobado su correcto funcionamiento [34] [35].

Git es un sistema de control de versiones distribuido lo que implica que aparte del servidor central existen copias del repositorio en todos los dispositivos que trabajan en el. Esto no solo permite que en caso de que alguno de los servidores deje de funcionar se siga teniendo acceso al código, sino que facilita mucho los flujos de trabajo al permitirte probar los cambios en local antes de enviarlos al servidor central.

GitLab

Aunque existen diversos tipos de gestores de repositorios *online* nos hemos decantado por GitLab [36] por su gran cantidad de herramientas adicionales y porque sus políticas de *hosting* gratuitas son algo mejores para el trabajo en equipo que las de la competencia.

Una de las principales herramientas que ha hecho que nos decantemos por GitLab es su gestor de *issues* [37], el cual se encuentra integrado en el propio sistema de control de versiones y permite controlar las tareas de desarrollo de manera sencilla. Este gestor de *issues* nos permite, entre muchas otras cosas, crear tareas, asignarlas a los distintos miembros del equipo de desarrollo y estimar y contabilizar los tiempos de desarrollo para dichas tareas.

Bitbucket

Además de GitLab, se ha creado un otro repositorio en Atlassian Bitbucket [38] a modo de repositorio de respaldo. De esta manera, en caso de perder el acceso temporalmente a nuestro repositorio original de GitLab podremos seguir trabajando con la copia almacenada en este otro. Para ello, se ha configurado GitLab para crear un *mirror* que cada vez que se incorporen nuevos cambios los envíe de manera automática a nuestro repositorio de respaldo de Bitbucket.

5.1.2. API-Football

Los datos utilizados por la aplicación se han obtenido a través de la versión de fútbol [14] de la colección de APIs "API-Sports"[39]. Durante el desarrollo del proyecto se ha trabajado con dos versiones distintas de esta API: la versión v2.0 y la versión v3.0. La versión v2.0 se utilizó durante la primera mitad del proyecto ya que, aunque la nueva versión era pública y contenía considerables mejoras respecto a su predecesora, se encontraba en fase beta de desarrollo pudiendo producir errores inesperados. Finalmente, con la *release* oficial de la versión v3.0 se decidió adaptar el proyecto a este nuevo servicio, puesto que facilitaba numerosas acciones que anteriormente no eran posibles (p. ej. recuperar todas las temporadas disponibles para una liga concreta).

A continuación se realizará un listado con los *endpoints* de la API que se han utilizado en el proyecto:

- **Countries**: Permite el acceso a la información de todos los países disponibles en la API.
- **Leagues**: Facilita el acceso a información relativa a todas las ligas de la API.
- **Teams**: Permite el acceso a la información de los equipos disponibles así como sus estadísticas.
- **Standings**: Permite el acceso a las clasificaciones de una liga concreta.
- **Fixtures**: Permite acceder a partidos (según la liga, cara a caras, partidos concretos, etc) e información relativa a estos (estadísticas, eventos, o alineaciones).
- **Injuries**: Proporciona información sobre lesiones aunque de manera muy limitada.
- **Coachs**: Permite el acceso a información de entrenadores.
- **Players**: Proporciona toda la información relativa a los distintos jugadores.
- **Transfers**: Facilita información sobre transferencias y el mercado de fichajes.
- **Trophies**: Permite acceder a los trofeos obtenidos por un jugador o entrenador.

- **Sidelined**: Proporciona información relativa a expulsiones o jugadores lesionados.

Por brevedad se han omitido gran parte de los datos que se pueden obtener de estos *endpoints*. Para más detalle consultar la figura 5.1 o la documentación de la API [40].

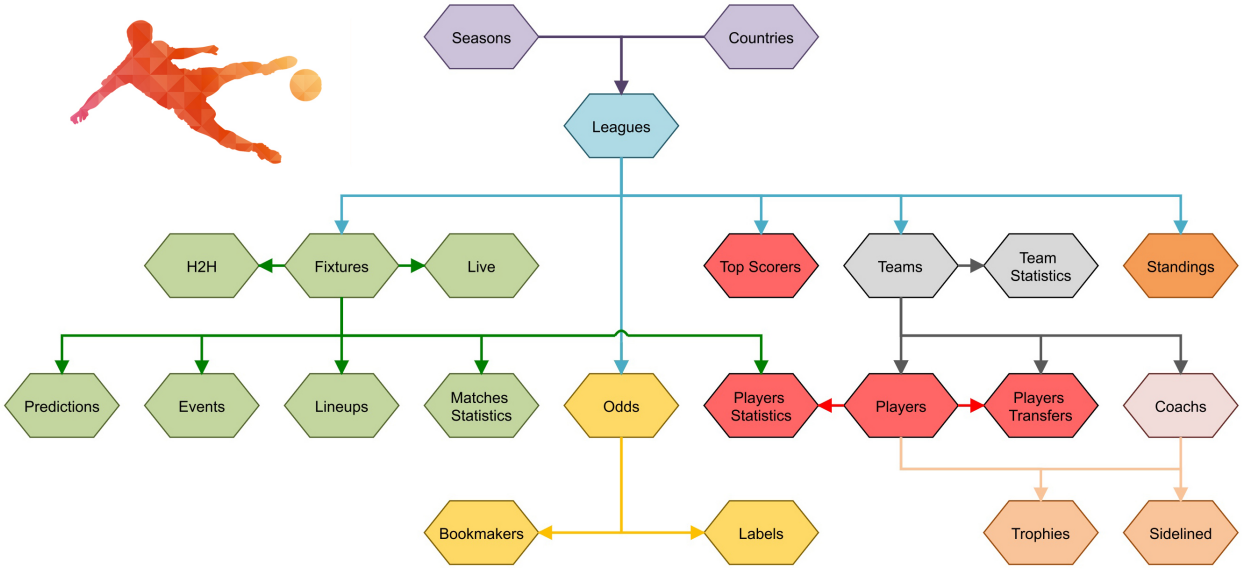


Figura 5.1: Arquitectura de la API [40].

5.1.3. Vue.js

Vue.js es un *framework* progresivo utilizado para construir interfaces para páginas web [41]. Este *framework* es ampliamente utilizado en el desarrollo *front-end* como herramienta para construir *Single-page application* (SPA). Este tipo de aplicaciones se caracterizan por estar compuestas por una única página que se carga una única vez y se va actualizando de manera dinámica en función de las acciones del usuario [42].

El funcionamiento de Vue se basa en componentes de un único archivo, de este modo en vez de dividir el código en tres capas independientes lo dividimos en componentes más livianos completamente funcionales. Estos componentes se estructuran de la siguiente manera:

- **Template**: Contiene la estructura del componente construida a partir de código HTML.
- **Script**: Contiene toda la funcionalidad y los datos del componente escrita utilizando código JavaScript O TypeScript. Este apartado cuenta a su vez con sus propias secciones predefinidas que permiten distintas funcionalidades en nuestro componente [43]:
  - *name*: Nombre del componente.
  - *components*: Lista de todos los componentes que se utilizan en el componente actual.
  - *props*: Propiedades del componente. Permiten compartir información entre componentes y habilitar características de nuestro componente de forma sencilla.
  - *data*: Una función que devuelve todos los datos del componente. Se utiliza una función en vez de un objeto para que cada instancia del componente tenga sus propios datos independientes de las demás instancias.

- *computed*: Permite declarar variables computadas en nuestro componente, es decir, son como variables del *data* pero calculadas en tiempo de ejecución de manera que si alguno de los datos cambia el valor de la variable computada cambiará.
- *watch*: Permite observar el comportamiento de determinadas variables posibilitando la realización acciones en el momento que se realice un cambio sobre la variable observada.
- Eventos del ciclo de vida: Se encuentran detallados más adelante en la sección 5.1.3.
- *methods*: Permite declarar las funciones que utilizará nuestro componente.

- **Style**: Contiene el código CSS del componente. También permite configurar el *scoope* de las clases CSS del componente y establecer un preprocesador a utilizar permitiendo de esta manera expandir las características base de CSS.

Ciclo de vida

Cada vez que se crea una nueva instancia de Vue esta atraviesa una secuencia de pasos de inicialización. Vue cuenta con una serie de funciones predefinidas que permiten añadir nueva funcionalidad durante estas etapas [44]. Las etapas mencionadas se pueden consultar en la lista mostrada a continuación o en la figura 5.2.

- *beforeCreate*: Es invocado inmediatamente después de que se haya inicializado la instancia y antes de la observación de datos y la configuración de eventos.
- *created*: Se ejecuta después de crear la instancia, una vez se ha terminado de procesar las opciones y se han configurado los eventos.
- *beforeMount*: Se invoca justo antes de que comience el montaje.
- *mounted*: Es ejecutado después de que se haya montado la instancia (no garantiza que los componentes secundarios se hayan montado).
- *beforeUpdate*: En invocado cuando los datos cambian, antes de actualizar el DOM.
- *updated*: Se ejecuta justo después del re-renderizado del DOM de la instancia.
- *beforeDestroy*: Se ejecuta justo antes de la destrucción de la instancia.
- *destroyed*: Se ejecuta justo después de la destrucción de la instancia.

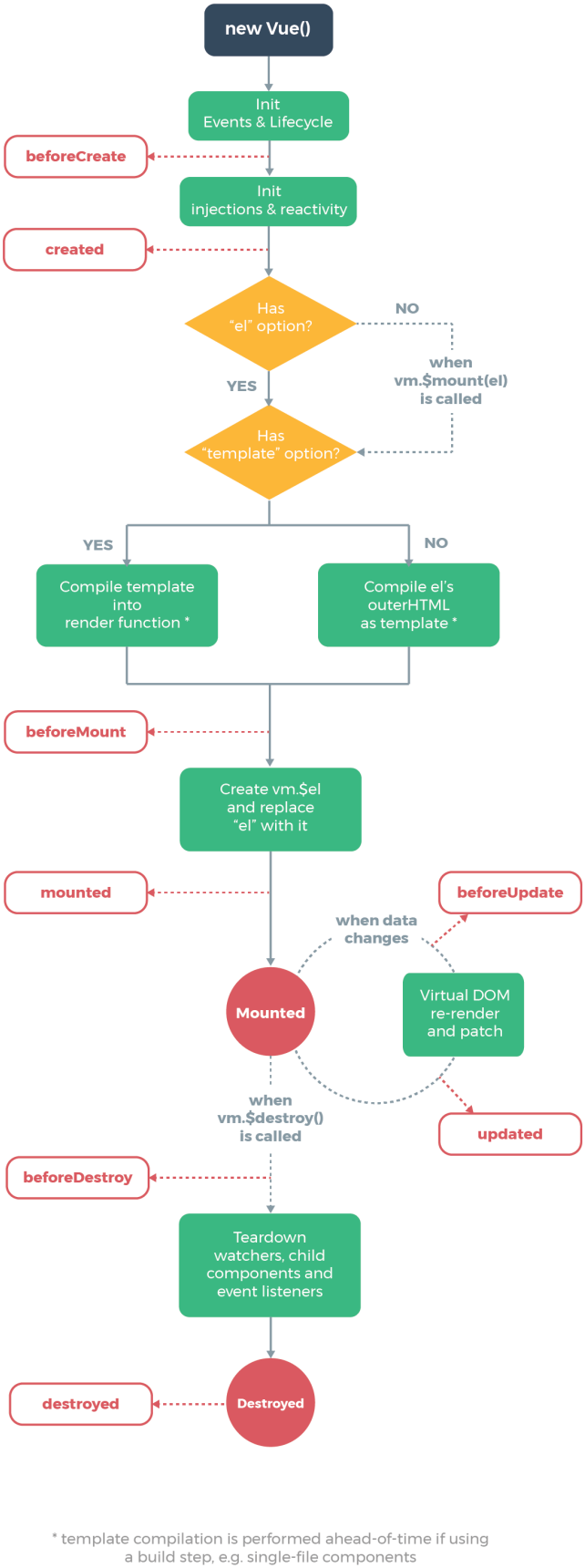


Figura 5.2: Ciclo de vida de la instancia Vue[45].

5.1.4. Firebase

Es una plataforma en la nube diseñada para el desarrollo de aplicaciones web, que proporciona un conjunto de herramientas para la creación y sincronización de proyectos, facilitando tareas como la extensibilidad [46]. Firebase puede ser utilizado para sustituir los Back-end tradicionales, ya que permite la implementación de una base de datos propia y una forma de implementar funciones personalizadas. Proporciona también una gran cantidad de herramientas [17] de las cuales en este proyecto se han usado las siguientes:

- **Authentication**: Facilita las tareas de autentificación del proyecto. Permite crear usuarios de forma sencilla a partir de distintos métodos de autentificación y todas sus tareas asociadas (p.ej. recuperación de contraseñas).
- **Firestore**: Proporciona una base de datos NoSQL muy fácil de utilizar. Se encuentra formada por colecciones y documentos. Las colecciones pueden una serie de documentos que representan cada objeto de la base de datos. Cada documento es un fichero JSON con un identificador único para toda la base de datos que puede contener cualquier tipo de datos, incluyendo sub-colecciones.
- **Hosting**: Proporciona un sistema de *hosting* que permite el despliegue de aplicaciones web.

5.1.5. NPM

Node Package Manager [47], más conocido como NPM, es el gestor de paquetes de Node.js, uno de los entornos de ejecución de JavaScript más utilizados. Este nos proporciona una forma sencilla de gestionar las dependencias de nuestro proyecto.

@vue/cli

Vue CLI [48] es una herramienta que nos permite crear proyectos Vue de manera sencilla y ordenada. Esta herramienta ayuda a crear la estructura del proyecto siguiendo los estándares de la industria y agregar dependencias compatibles y plugins evitando posibles conflictos.

@vue-cli-plugin-vuetify

Vuetify [49] es una librería de componentes Vue para la creación de interfaces siguiendo los estándares de diseño establecidos por Material Design. Esta herramienta facilita las labores de diseño de interfaces proporcionando componentes predefinidos fácilmente configurables que pueden ser utilizados para agregar ciertas funcionalidades a nuestro proyecto.

@vue/cli-plugin-router

Vue Router [50] es un *plugin* que facilita la configuración de la navegación en Vue. Este permite establecer un sistema a partir de rutas como se haría en las aplicaciones web tradicionales, pero dentro de una SPA. Permite añadir rutas, configurar opciones de restricción para estas, parsear los enlaces para obtener parámetros o querys, etc.

@vue/cli-plugin-vuex

Vuex [51] es un *plugin* de Vue que proporciona una forma sencilla para añadir estado a nuestra aplicación Vue. Sirve como almacén centralizado para los datos que se van a compartir entre los distintos componentes de la aplicación. Para ello utiliza reglas que garantizan que el estado solo se puede modificar de una manera predecible. En nuestra aplicación se ha utilizado para gestionar el login de usuarios.

@vue-cli-plugin-i18n

Vue i18n [52] es un *plugin* de internacionalización para aplicaciones Vue. Proporciona una forma sencilla de configurar los textos de la aplicación de manera que se puedan añadir distintos lenguajes y traducciones.

@vue/cli-plugin-babel

Babel [53] es una cadena de herramientas que permite transformar código ECMAScript a versiones compatibles para los distintos navegadores. Esta es una herramienta muy útil, ya que a partir de un único código podemos crear aplicaciones funcionales para todos los navegadores.

@vue/cli-plugin-eslint

EsLint [54] es una herramienta utilizada para comprobar la estructura de nuestro código. De esta manera nos aseguramos de que este se encuentra formateado de manera correcta favoreciendo su legibilidad para futuros trabajadores y evitando que existan secciones de código que no se utilizan correctamente.

@vue/cli-plugin-e2e-cypress

Cypress [55] es una herramienta utilizada para la creación de pruebas automatizadas del tipo *end-to-end* para aplicaciones Vue. Este tipo de testeo trata de probar una aplicación imitando los posibles comportamientos de un usuario visitando la web.

dyson

Dyson [56] es una herramienta que permite la creación de un *fake server* para simular las respuestas de una API. En este proyecto se ha utilizado para *mokear* las respuestas de los distintos *endpoints* de la API de fútbol utilizada para así facilitar las tareas de desarrollo.

@mdi/font

Esta herramienta [57] permite utilizar la colección de iconos de Material Design [58]. Es muy útil porque permite una fácil incorporación de iconos muy conocidos y que representan un estándar en la industria.

5.1.6. Astah

Astah [59] es la herramienta por excelencia para la realización de diagramas UML en la Universidad de Valladolid. Aunque existen diversas alternativas se ha escogido esta herramienta por ser la que más hemos utilizado durante la etapa de formación.

5.1.7. Trello

Trello [60] es una herramienta de administración y gestión de proyectos. Se ha utilizado junto al gestor de issues de GitLab para realizar un seguimiento de las tareas de desarrollo del proyecto.

5.1.8. Zotero

Zotero [61] es una herramienta muy útil para la gestión de documentación de un proyecto. Permite guardar y catalogar toda la información consultada durante el desarrollo del proyecto, creando incluso *snapshots* de las páginas consultadas por si el contenido original es eliminado. También facilita la creación de bibliografías escaneando los datos de las páginas guardadas y permitiendo exportarlos en distintos formatos para su uso.

5.1.9. Overleaf

Overleaf [62] es un editor *online* de documentos LaTeX. Se ha utilizado para redactar la documentación de este proyecto. Ha sido escogido por su facilidad de uso y porque cuenta con herramientas como un sistema de control de versiones integrado.

5.1.10. InkScape

InkScape [63] es una herramienta de diseño de gráficos vectoriales de código abierto. En el proyecto se ha utilizado para realizar el diseño del logo de la aplicación, iconos y algunos gráficos vectoriales extra.

5.1.11. MockFlow

MockFlow [64] es una herramienta web que permite diseñar *wireframes* (bocetos) para interfaces web. Se ha utilizado su versión gratuita para el diseño inicial de las interfaces a desarrollar.

5.1.12. Coolors

Coolors [65] es una herramienta web que permite realizar tareas relacionadas con el color. En ella se pueden visualizar paletas de colores, generarlas de manera aleatoria, generar paletas siguiendo patrones preestablecidos o explorar paletas de colores populares en el momento.

5.2. Herramientas descartadas

A continuación se describirán una serie de herramientas que, aunque no fueron utilizadas en la versión final del proyecto, si formaron parte de la etapa de aprendizaje durante el Sprint 0.

5.2.1. Storybook

Storybook [66] es una herramienta de código abierto para el desarrollo y testeo de componentes UI de manera independiente. Aunque es una herramienta tremendamente útil para desarrollar y probar los componentes de manera independiente a las páginas fue descartada tras una serie de problemas al intentar compatibilizarla con Vuetify.

5.2.2. Adobe InDesign

Adobe InDesign [67] es un software de diseño y composición de páginas gratuito perteneciente a la suite de Adobe. Aunque es una herramienta muy potente se terminó descartando por contar con una curva de aprendizaje superior a otras aplicaciones.

5.2.3. Beautiful Soup

Beautiful Soup [68] es una biblioteca de python para analizar documentos HTML utilizada en *web scraping*. Inicialmente se planeó obtener la información relativa al fútbol a través de métodos de *web scraping* pero finalmente por razones legales y facilidad de uso fue descartada y reemplazada por la API "API-futbol".

5.2.4. @vue/cli-plugin-unit-jest

Jest [69] es una herramienta diseñada para la creación exclusiva de tests unitarios para aplicaciones Vue. Aunque la idea inicial era realizar test unitarios y *end-to-end* de toda la aplicación, los primeros fueron reemplazados por pruebas de componentes más adecuadas para las características de este proyecto.

5.2.5. GitLab CI/CD

Es la herramienta que proporciona la plataforma GitLab para tareas de integración y despliegue continuos [70]. Aunque la idea inicial era implementar este sistema de manera que se ejecutaran los tests de cypress y jest se terminó descartando porque las ventajas que proporciona este sistema no son tan grandes para un grupo de desarrollo de una única persona y conllevan un tiempo de aprendizaje amplio. Además al haber aplazado las tareas de testeo al final del proyecto las ventajas se vieron reducidas drásticamente.

Capítulo 6

Seguimiento del proyecto

6.1. Seguimiento de los sprints realizados

Para complementar el capítulo 3 sobre la planificación del proyecto y para asegurarnos de que este se desarrolla de manera adecuada se ha realizado un seguimiento del proyecto. Las tareas asociadas a cada sprint se han establecido al inicio del mismo. Para facilitar la comprensión del seguimiento del proyecto se ha establecido un código para catalogar las distintas tareas según su tipo:

DOC: Tareas relacionadas con la documentación del proyecto.

DEV: Tareas relacionadas con el desarrollo de alguna propiedad del proyecto.

HU-n: Tareas relacionadas con alguna historia de usuario concreta identificada por su número.

FIX: Tareas relacionadas con algún tipo de error encontrado.

Para cada tarea realizada durante un sprint se establecerá su número de tarea, su tipo, una breve descripción de la tarea a realizar, el número de horas que se han empleado para llevar a cabo la tarea y el estado de la misma al finalizar el Sprint. Los distintos estados por los que pasará una tarea son los siguientes:

No iniciada: La tarea especificada no ha sido iniciada por lo que se aplazará a futuros sprints.

En progreso: La tarea especificada se encuentra en progreso pero no ha sido finalizada por lo que se finalizará en el próximo sprint.

Completada: La tarea especificada ha sido finalizada durante el desarrollo del sprint.

Descartada: Tarea en la que se ha trabajado pero que por alguna razón ha terminado siendo descartada.

6.1.1. Sprint 0

Como se ha mencionado en la Sección 3.2, al inicio del proyecto se ha realizado un primer sprint inicial para llevar a cabo las tareas de documentación relevantes y preparar la estructura inicial del proyecto añadiendo las dependencias necesarias.

Durante este Sprint también se han llevado a cabo las tareas de configuración de los *plugins* i18n, Vue Router y Vuetify (Sección 5.1.5). Además, se ha configurado el linter (herramienta de control sobre el formateo del código) de manera que se consiga un código limpio pero sin que la herramienta termine siendo pesada resaltando más errores de los necesarios.

Finalmente se ha arreglado un error en el archivo base de configuración del *plugin* i18n por el cual los ficheros JSON pertenecientes a cada idioma no se paseaban de manera correcta mostrando errores en las cadenas de texto de prueba.

Tarea	Tipo	Descripción	Horas	Estado
T - 001	DOC	Definición de las tareas del backlog	2h	Completada
T - 002	DOC	Estructura de la documentación	1h	Completada
T - 003	DOC	Documentación de la introducción y los objetivos	6h	Completada
T - 004	DOC	Documentación de la planificación y presupuesto	1h	En progreso
T - 005	DOC	Documentación de los riesgos	5h	En progreso
T - 006	DOC	Documentación del sprint 0	2h	Completada
T - 007	DOC	Formación sobre la tecnología utilizada Vue.js	10h	Completada
T - 008	DOC	Configuración del proyecto Git	1h	Completada
T - 009	DEV	Creación de la estructura inicial del proyecto	2h	Completada
T - 010	DEV	Instalación de las dependencias del proyecto	1h	Completada
T - 011	DEV	Configuración de i18n	3h	Completada
T - 012	DEV	Configuración de ESLint	2h	Completada
T - 013	DEV	Configuración de Vue Router	2h	Completada
T - 014	DEV	Configuración de Vuetify	0.5h	Completada
T - 015	DEV	Creación de los ficheros de localización del proyecto	1h	Completada
T - 016	FIX	Solución de un error en la configuración de la localización	2h	Completada
Total			**35.5h**	

Tabla 6.1: Tareas establecidas para el sprint 0

6.1.2. Sprint 1

Durante este sprint se ha finalizado la tarea de documentación sobre la planificación del proyecto y presupuestos aplazada en el Sprint anterior. La tarea relacionada con la documentación del plan de riesgos ha sido aplazada al siguiente sprint.

También se ha trabajado en la estructura inicial de las barras de navegación. Para ello se han implementado los componentes NavigationDrawer, AppBar y Footer (Sección 7.2.1). Esta estructura inicial está más planteada a modo de maquetación que de sistema funcional ya que se está buscando una manera óptima de compartir los enlaces de la barra de aplicación y la barra de navegación.

Además, se han creado las vistas de *LogIn* y *SignUp* (Sección 7.2.1) junto con dos componentes separados para

cada uno de sus formularios. Se han añadido reglas de validación de los formularios pero aun no hay funcionalidad.

Se ha añadido también un fichero de variables de estilo compartidas para utilizar entre los distintos componentes.

Por último, se ha creado un componente para cambiar el idioma de la página web de manera dinámica. Esto ha necesitado varias horas de documentación debido a los bajos conocimientos sobre esta herramienta.

Tarea	Tipo	Descripción	Horas	Estado
T - 004	DOC	Documentación de la planificación y presupuesto	10h	Completada
T - 005	DOC	Documentación de los riesgos	0h	En progreso
T - 017	DOC	Documentación del sprint 1	1h	Completada
T - 018	HU-8	Realización de bocetos	1h	Completada
T - 019	HU-9	Realización de bocetos	1h	Completada
T - 020	DEV	Sistema de navegación	2h	En progreso
T - 021	HU-19	Construcción del componente LocaleSwitcher	5h	Completada
T - 022	HU-8	Construcción del componente SignUpCard	5h	Completada
T - 023	HU-9	Construcción del componente LogInCard	4h	Completada
T - 024	HU-8	Creada vista SignUp	0.5h	Completada
T - 025	HU-9	Creada vista LogIn	0.5h	Completada
T - 026	DEV	Añadido y configurado un fichero de variables CSS	2h	Completada
Total			**32h**	

Tabla 6.2: Tareas establecidas para el sprint 1

6.1.3. Sprint 2

A partir de este sprint el tiempo dedicado a las tareas se va a reducir por falta de tiempo del personal de desarrollo.

Durante este sprint se finaliza la documentación del plan de riesgos y se comienza con la documentación de las tecnologías utilizadas. Además se continua con el sistema de navegación. Para esto se ha creado un tercer componente que sirva de *wrapper* para los componentes de navegación y sea el el que se encargue de compartir las rutas, activar o desactivar el *navigation drawer*, etc.

Debido a lo anterior se ha terminado de implementar la funcionalidad de los componentes NavigationDrawer, AppBar y Footer (Sección 7.2.1).

También se ha empezado a trabajar con el *plugin* StoryBook (Sección 5.2.1) para trabajar en los componentes de manera aislada y poder probarlos de forma independiente al resto de la aplicación pero debido a problemas de compatibilidad con la configuración de Vuetify (Sección 5.1.5) ha terminado siendo descartada.

Por último, la tarea de añadir la API al proyecto se retrasa para futuras iteraciones.

Tarea	Tipo	Descripción	Horas	Estado
T - 005	DOC	Documentación de los riesgos	4h	Completada
T - 027	DOC	Documentación del sprint 2	1h	Completada
T - 028	DOC	Documentación de las tecnologías utilizadas	4h	En progreso
T - 029	DEV	Construcción del componente NavigationHandler	6h	En progreso
T - 030	DEV	Construcción del componente NavigationDrawer	1h	Completada
T - 031	DEV	Construcción del componente AppBar	2h	Completada
T - 020	DEV	Sistema de navegación	1h	Completada
T - 032	DEV	Añadir StoryBook al proyecto	7h	Descartada
T - 033	DEV	Añadir API-Football	0h	No iniciada
Total			**26h**	

Tabla 6.3: Tareas establecidas para el sprint 2

6.1.4. Sprint 3

Durante este sprint se ha finalizado la tarea de documentación sobre las tecnologías utilizadas.

Se ha trabajado en agregar la API de resultados deportivos. Se ha conseguido integrar correctamente y se han implementado algunas de las peticiones que se utilizarán más adelante.

Se ha trabajado en la corrección de algunos *bugs* encontrados en los componentes de la navegación y la localización así como en la incorporación de todas las cadenas de texto añadidas durante este Sprint a los ficheros de localización.

Se ha comenzado a trabajar en el contenido de la web creando los componentes de tipo *card* para las ligas y partidos (Sección 7.2.1) y un contenedor horizontal que muestre las *cards* de las ligas.

Tarea	Tipo	Descripción	Horas	Estado
T - 028	DOC	Documentación de las tecnologías utilizadas	6h	Completada
T - 029	DEV	Construcción del componente NavigationHandler	2.5h	Completada
T - 033	DEV	Añadir API-Football	5h	En progreso
T - 034	DOC	Documentación del sprint 3	1h	Completada
T - 035	HU-19	Internacionalización de las cadenas de texto añadidas en le Sprint 3	0.5h	Completada
T - 036	HU-8	Refactorizar formulario de SignUp 3	1h	Completada
T - 037	FIX	Corrección de errores en la navegación	2h	Completada
T - 038	FIX	Corrección de errores en la localización	1.5h	Completada
T - 039	HU-1	Crear LeagueCard	2h	Completada
T - 040	HU-3	Crear FixtureCard	1h	Completada
T - 041	HU-1	Crear LeaguesSliedGroup	2h	Completada
Total			**24.5h**	

Tabla 6.4: Tareas establecidas para el sprint 3

6.1.5. Sprint 4

El Sprint 4 no se ha podido realizar por la ausencia del personal del equipo de desarrollo durante las dos semanas de duración del mismo. Para mantenernos ajustados a la planificación original se mantiene el nombre del sprint asignado a las fechas planificadas y, siguiendo el plan de contingencia disponible en la sección 4.1.4, se añadirá un nuevo sprint al final de la planificación inicial para finalizar las tareas que hayan sido retrasadas.

6.1.6. Sprint 5

Durante este sprint se ha realizado la documentación del diseño de la aplicación (el diseño de la interfaz de usuario al no estar finalizado no se ha terminado).

Se ha arreglado un *bug* por el cual al hacer click fuera del *navigation drawer* el estado de este no cambiaba y era necesario pulsar dos veces el botón para abrirlo. Para ello se va a gestionar el estado del *drawer* internamente y se va a modificar a partir de funciones.

También se ha estado trabajando en la vista de Partidos. Se han creado todos los componentes con información relativa a partidos y un sistema que permite a la vista cargar estos componentes de forma dinámica para que cada uno se muestre en la pestaña adecuada. Además se han añadido las peticiones de la API referentes a partidos.

Se ha reestructurado el sistema de carpetas de las vistas. Ahora se encuentra organizado según si las vistas son vistas de la aplicación o vistas de fútbol. Se han creado subcarpetas para las vistas de fútbol según su tipo (partidos, ligas, equipos, etc).

Tarea	Tipo	Descripción	Horas	Estado
T - 033	DEV	Añadir API-Football	2h	Completada
T - 042	DOC	Documentación del diseño de la aplicación	5h	Completada
T - 043	DOC	Documentación del sprint 5	1h	Completada
T - 044	HU-3	Realización de bocetos	0.5h	Completada
T - 045	HU-1	Realización de bocetos	0.5h	Completada
T - 046	HU-3	Crear vista Fixture	6h	Completada
T - 047	HU-1	Crear vista Leagues	1h	En progreso
T - 048	HU-3	Crear componente FixtureHeader	3h	Completada
T - 049	HU-3	Crear componente FixtureEvents	2h	En progreso
T - 050	HU-3	Crear componente FixtureListItem	4h	Completada
T - 051	HU-3	Crear componente FixtureStatistics	1h	En progreso
T - 052	HU-3	Crear componente StatsBar	3h	Completada
T - 053	HU-3	Crear componente Lineups	2h	En progreso
T - 054	DEV	Añadir las vistas creadas a router	0.5h	Completada
T - 055	HU-19	Internacionalización de las cadenas de texto añadidas en el Sprint 5	0.5h	Completada
T - 056	FIX	Arreglar bug del NavigationDrawer	3h	Completada
Total			**35h**	

Tabla 6.5: Tareas establecidas para el sprint 5

6.1.7. Sprint 6

Las horas de trabajo durante este sprint han sido menores que en los anteriores debido a limitaciones en la disponibilidad de los miembros del equipo de desarrollo.

También se ha trabajado en los componentes Lineups y Head2Head (Sección 7.2.1) referentes a alineaciones y partidos cara a cara respectivamente. Para ello se ha trabajado en añadir a las peticiones de la API las consultas relativas a los cara a cara.

Posteriormente se ha trabajado en añadir estos dos componentes a la vista Fixtures (Sección 7.2.1).

Tarea	Tipo	Descripción	Horas	Estado
T - 047	HU-1	Crear vista Leagues	0h	En progreso
T - 049	HU-3	Crear componente FixtureEvents	0h	En progreso
T - 051	HU-3	Crear componente FixtureStatistics	0h	En progreso
T - 057	DOC	Documentación del sprint 6	1h	Completada
T - 058	HU-1	Realización de bocetos	0.5h	Completada
T - 059	HU-2	Realización de bocetos	0.5h	Completada
T - 060	HU-4	Realización de bocetos	0.5h	Completada
T - 061	HU-3	Crear componente Lineups	2h	Completada
T - 062	HU-3	Crear componente LineupList	3h	Completada
T - 063	HU-3	Crear componente LineupListItem	4h	Completada
T - 064	HU-3	Crear componente Head2Head	4h	Completada
T - 065	HU-3	Refactorizar FixtureListItem	1h	Completada
T - 066	HU-3	Añadir peticiones a la API referentes a los Cara a cara	0.5h	Completada
T - 067	HU-3	Añadir componente Head2Head a la vista Fixture	2h	Completada
T - 068	HU-3	Añadir componente Lineups a la vista Fixture	2h	Completada
T - 069	DEV	Internacionalización de las cadenas de texto añadidas en el Sprint 6	0.5h	Completada
Total			**21.5h**	

Tabla 6.6: Tareas establecidas para el sprint 6

6.1.8. Sprint 7

Durante el desarrollo de este Sprint se ha trabajado en crear las vistas League, Team y Player (Sección 7.2.1) pertenecientes a las historias de usuario 1, 2 y 4 respectivamente (Tabla 3.2). Para cada una de ellas se ha implementado una serie de componentes que facilitan la visualización de los datos mostrados. El desglose de las tareas realizadas puede consultarse en la tabla 6.7.

Las tareas 47, 49 y 51 vuelven a quedar en progreso ya que se ha decidido priorizar la implementación de las vistas League, Team y Player.

Tarea	Tipo	Descripción	Horas	Estado
T - 047	HU-1	Crear vista Leagues	0h	En progreso
T - 049	HU-3	Crear componente FixtureEvents	0h	En progreso
T - 051	HU-3	Crear componente FixtureStatistics	0h	En progreso
T - 070	DOC	Documentación del sprint 7	1h	Completada
T - 071	HU-1	Crear vista League	2h	Completada
T - 072	HU-1	Crear componente LeagueGrid	3h	Completada
T - 073	HU-1	Crear componente LeagueGridCard	2h	Completada
T - 074	HU-1	Refactorizar componente LeagueCard	0.5h	Completada
T - 075	HU-1	Crear componente LeagueHeader	2h	Completada
T - 076	HU-6	Crear componente Country	1h	Completada
T - 077	HU-3	Refactorizar FixtureListItem	0.5h	Completada
T - 078	HU-1	Crear componente LeagueStandings	1h	Completada
T - 079	HU-1	Crear componente Standings	2h	Completada
T - 080	HU-2	Crear vista Team	2h	Completada
T - 081	HU-2	Crear componente TeamHeader	3h	Completada
T - 082	HU-2	Crear componente TeamList	2h	Completada
T - 083	HU-2	Crear componente TeamListItem	2h	Completada
T - 084	HU-2	Crear componente VenueCard	1h	Completada
T - 085	HU-4	Crear vista Player	2h	Completada
T - 086	HU-4	Crear componente PlayerHeader	2h	Completada
T - 087	HU-4	Crear componente PlayerStatistics	1h	En progreso
T - 088	HU-4	Crear componente TransfersList	2h	Completada
T - 089	HU-4	Crear componente TransfersListItem	1h	Completada
T - 090	HU-4	Crear componente SidelinedTable	2h	Completada
T - 091	HU-4	Crear componente TrophieList	1h	Descartada
T - 092	HU-4	Crear componente TrophieListItem	2h	Descartada
T - 093	HU-4	Crear componente TrophieTable	3h	Completada
T - 094	HU-3	Añadir peticiones a la API referentes a partidos	0.5h	Completada
T - 095	HU-1	Añadir peticiones a la API referentes a ligas	0.5h	Completada
T - 096	HU-1	Añadir peticiones a la API referentes a clasificaciones	0.5h	Completada
T - 097	HU-2	Añadir peticiones a la API referentes a equipos	0.5h	Completada
T - 098	HU-4	Añadir peticiones a la API referentes a jugadores	0.5h	Completada
T - 099	HU-4	Añadir peticiones a la API referentes a trofeos	0.5h	Completada
T - 100	DEV	Internacionalización de las cadenas de texto añadidas en el Sprint 7	1h	Completada
T - 101	DEV	Añadir las vistas creadas a router	0.5h	Completada
Total			**45.5h**	

Tabla 6.7: Tareas establecidas para el sprint 7

6.1.9. Sprint 8

Durante este sprint se ha trabajado en las tareas relativas a la historia de usuario 5 (Tabla 3.2). Para ello se construye la vista de entrenador y todos los componentes relacionados con esta.

Al igual que en el sprint anterior se vuelven a aplazar las tareas 47, 49 y 51. Además las tareas 108 y 109 quedan en progreso para el siguiente Sprint.

Tarea	Tipo	Descripción	Horas	Estado
T - 047	HU-1	Crear vista Leagues	0h	En progreso
T - 049	HU-3	Crear componente FixtureEvents	0h	En progreso
T - 051	HU-3	Crear componente FixtureStatistics	0h	En progreso
T - 102	DOC	Documentación del sprint 8	1h	Completada
T - 103	HU-5	Realización de bocetos	0.5h	Completada
T - 104	HU-5	Crear vista Coach	2h	Completada
T - 105	HU-5	Crear componente CareerList	2h	Completada
T - 106	HU-5	Crear componente CareerListItem	3h	Completada
T - 107	HU-5	Crear componente CoachHeader	4h	Completada
T - 108	HU-5	Crear componente CoachList	2h	En progreso
T - 109	HU-5	Crear componente CoachListItem	3h	En progreso
T - 110	HU-5	Añadir peticiones a la API referentes a los entrenadores	0.5h	Completada
T - 111	DEV	Actualizar router	0.5h	Completada
T - 112	DEV	Internacionalización de las cadenas de texto añadidas en el Sprint 8	0.5h	Completada
Total			**19h**	

Tabla 6.8: Tareas establecidas para el sprint 8

6.2. Planificación extendida

Al finalizar estos 8 sprints planteados inicialmente el estado de desarrollo del proyecto no era el esperado. Por ello se añadieron nuevos sprints para intentar finalizar el trabajo restante y llevar al proyecto a un estado más cercano al resultado deseado. A continuación (tabla 6.9) se enumeran los sprints no pertenecientes a la planificación inicial utilizados para finalizar el proyecto.

Sprint	**Comienzo**	**Finalización**	**Observaciones**
Sprint 9	05/07/2021	19/07/2021	Sprint complementario
Sprint 10	19/07/2021	02/08/2021	
Sprint 11	02/08/2021	16/08/2021	
Sprint 12	16/08/2021	30/08/2021	
Sprint 13	30/08/2021	13/09/2021	

Tabla 6.9: Planificación extendida

6.2.1. Sprint 9

Este sprint se ha dedicado en mayor medida a mejorar los componentes ya creados añadiendo más funcionalidad y traducciones más amplias del contenido.

Se han terminado las tareas 108 y 109 que se quedaron a medias en el sprint anterior. Las tareas 47, 49 y 51 se vuelven a retrasar para próximos Sprints.

Tarea	Tipo	Descripción	Horas	Estado
T - 047	HU-1	Crear vista Leagues	0h	En progreso
T - 049	HU-3	Crear componente FixtureEvents	0h	En progreso
T - 051	HU-3	Crear componente FixtureStatistics	0h	En progreso
T - 108	HU-5	Crear componente CoachList	1h	Completada
T - 109	HU-5	Crear componente CoachListItem	2h	Completada
T - 113	DOC	Documentación del sprint 9	1h	Completada
T - 114	HU-3	Refactorizar componente Lineup	1h	Completada
T - 115	HU-3	Refactorizar componente LineupList	1h	Completada
T - 116	HU-3	Refactorizar componente LineupListItem	2h	Completada
T - 117	HU-3	Añadir paginación al componente Head2Head	2h	Completada
T - 118	HU-3	Refactorizar componente FixtureHeader	3h	Completada
T - 119	HU-3	Refactorizar componente StatsBar	4h	Completada
T - 120	HU-3	Añadir paginación al componente FixtureList	1h	Completada
T - 121	HU-4	Refactorizar componente PlayerHeader	3h	Completada
T - 122	HU-4	Crear componente PlayerList	1h	Completada
T - 123	HU-4	Crear componente PlayerListItem	2h	Completada
T - 124	HU-4	Refactorizar componente TransferListItem	1h	Completada
T - 125	HU-4	Crear componente PlayerStatistics	0.5h	En progreso
T - 126	HU-2	Crear componente TeamTransfers	3h	Completada
T - 127	HU-2	Refactorizar componente TeamHeader	2h	Completada
T - 128	HU-2	Refactorizar componente TeamListItem	2h	Completada
T - 129	DEV	Refactorizar componente NavigationDrawer	2h	Completada
T - 130	DEV	Actualizar router	0.5h	Completada
T - 131	DEV	Internacionalización de las cadenas de texto añadidas en el Sprint 9	2h	Completada
Total			**37h**	

Tabla 6.10: Tareas establecidas para el sprint 9

6.2.2. Sprint 10

Durante este sprint se ha añadido un servidor Dyson (Sección 5.1.5) para simular los resultados obtenidos de la API. También se ha configurado de manera que para las rutas consultadas se devuelva información coherente y facilite las tareas de desarrollo.

También se ha trabajado en mejorar los componentes relativos a la navegación del proyecto. Se ha renovado el *footer* completamente y se ha modificado la forma de manejar los enlaces en el componente NavigationDrawer.

Se han realizado tareas de solución de errores sobre los componentes Eventos, MatchStatistics y FixtureHeader (Sección 7.2.1).

Se ha revisado el comportamiento de las vistas Fixture y League (Sección 7.2.1) realizando algunos cambios significativos.

Por último, se ha realizado una tarea de diseño de iconos con InckScape (Sección 5.1.10) para reemplazar la cadena de texto "form", utilizada dentro del componente Standings para mostrar el estado de forma de un equipo, es decir, si han ganado, perdido o empatado cada uno de los ultimo partidos jugados.

Tarea	Tipo	Descripción	Horas	Estado
T - 047	HU-1	Crear vista Leagues	0h	En progreso
T - 049	HU-3	Crear componente FixtureEvents	0h	En progreso
T - 051	HU-3	Crear componente FixtureStatistics	0h	En progreso
T - 125	HU-4	Crear componente PlayerStatistics	0h	En progreso
T - 132	DOC	Documentación del sprint 10	1h	Completada
T - 133	HU-7	Añadir selector de temporada a LeagueHeader	2h	En progreso
T - 134	DEV	Refactorizar Footer	2h	Completada
T - 135	DEV	Refactorizar NavigationDrawer	3h	Completada
T - 136	DEV	Cambios en NavigationHandler	1h	Completada
T - 137	DEV	Añadir distintos tipos de .env	1h	Completada
T - 138	DEV	Instalación y configuración de Dyson	6h	Completada
T - 139	FIX	Solución de un error en Eventos	0.5h	Completada
T - 140	FIX	Solución de un error en MatchStatistics	1h	Completada
T - 141	FIX	Solución de un error en FixtureHeader	0.5h	Completada
T - 142	DEV	Agregar configuración de temas a Vuetify	2h	Completada
T - 143	HU-3	Refactorizar vista Fixture	4h	Completada
T - 144	HU-1	Refactorizar vista League	1h	Completada
T - 145	HU-2	Crear componente TeamGrid	3h	Completada
T - 146	HU-1	Diseño de los iconos de Form	2h	Completada
T - 147	HU-1	Añadir iconos a Form	3h	Completada
T - 148	HU-1	Refactorizar Standings	2h	Completada
T - 149	HU-2	Crear componente TeamCard	2h	Completada
T - 150	DEV	Internacionalización de las cadenas de texto añadidas en el Sprint 10	1h	Completada
Total			**38h**	

Tabla 6.11: Tareas establecidas para el sprint 10

6.2.3. Sprint 11

Durante este sprint se ha finalizado la funcionalidad de la vista Fixture (Sección 7.2.1). Toda la información relativa a partidos se muestra ahora de la forma deseada. Con esto se han terminado las tareas 49 y 51 que se llevaban retrasando desde hacer varios sprints.

Se han añadido varias mejoras sobre componentes relacionados con las vistas de equipos, jugadores y entrenadores (Sección 7.2.1).

También se ha arreglado una serie de errores existentes en las rutas del proyecto

Por último, una de las tareas relacionadas con los partidos ha consistido en diseñar un gráfico vectorial que represente un campo de fútbol para mostrar las alineaciones de manera dinámica sobre él.

Tarea	Tipo	Descripción	Horas	Estado
T - 047	HU-1	Crear vista Leagues	0h	En progreso
T - 049	HU-3	Crear componente FixtureEvents	5h	Completada
T - 051	HU-3	Crear componente FixtureStatistics	4h	Completada
T - 151	HU-4	Crear componente PlayerStatistics	2h	Completada
T - 152	DOC	Documentación del sprint 11	1h	Completada
T - 153	HU-3	Añadir cabecera a FIxtureListItem	1h	Completada
T - 133	HU-7	Añadir selector de temporada a LeagueHeader	1h	Completada
T - 154	HU-4	Refactorizar sidelined table	0.5h	Completada
T - 155	HU-1	Refactorizar componente Standings	0.5h	Completada
T - 156	HU-4	Crear PlayerCard	2h	Completada
T - 157	HU-4	Crear PlayerGrid	1h	Completada
T - 158	HU-2	Refactorizar TeamHeader	2h	Completada
T - 159	HU-2	Refactorizar vista Team	2h	Completada
T - 160	HU-4	Crear TransferGrid	1h	Completada
T - 161	HU-4	Crear Transfercard	2h	Completada
T - 162	HU-2	Refactorizar TeamTransferListItem	2h	Completada
T - 163	HU-5	Crear CoachGrid	1h	Completada
T - 164	HU-5	Crear CoachCard	2h	Completada
T - 165	HU-3	Diseñar campo de fútbol para Lineup	3h	Completada
T - 166	HU-3	Refactorizar FixtureHeader	1h	Completada
T - 167	HU-3	Añadir botón SaveFixture	2h	Completada
T - 168	HU-5	Crear CarreerGrid	1h	Completada
T - 169	HU-5	Crear lista de trofeos	2h	Completada
T - 170	HU-4	Crear componente TopPlayers	3h	Completada
T - 171	HU-5	Refactorizar CoachHeader	2h	Completada
T - 172	HU-4	Refactorizar PlayerHeader	2h	Completada
T - 173	FIX	Arreglar rutas	3h	Completada
T - 174	HU-2	Crear TeamStatistics	3h	Completada
T - 175	DEV	Internacionalización de las cadenas de texto añadidas en el Sprint 11	1h	Completada
Total			**53h**	

Tabla 6.12: Tareas establecidas para el sprint 11

6.2.4. Sprint 12

Durante este sprint se ha trabajado en añadir la funcionalidad relacionada con la interacción del usuario. Para ello se ha añadido un sistema de *signup* y *login* que permita a los usuarios registrarse y un número de acciones que permitan a un usuario interactuar con los distintos tipos de datos de la aplicación, así como modificar sus propios datos.

También se ha trabajado en permitir filtrar y ordenar algunos de los datos mostrados por la aplicación para componentes como FixtureList o LeaguesGrid (Sección 7.2.1).

Por último, se han arreglado errores existentes en la visualización de algunas de las vistas.

Tarea	Tipo	Descripción	Horas	Estado
T - 047	HU-1	Crear vista Leagues	2h	Completada
T - 176	DOC	Documentación del sprint 12	1h	Completada
T - 177	FIX	Añadir gráficas a TeamStatistics	2h	Descartada
T - 178	DEV	Añadir Firebase	1h	Completada
T - 179	HU-8	Configurar Vuex para LogIn y SignUp	5h	Completada
T - 180	HU-9	Configurar Navegación para adaptarla a LogIn	3h	Completada
T - 181	HU-6	Crear vista Country	3h	Completada
T - 182	HU-1	Añadir paginación a LeaguesGrid	2h	Completada
T - 183	HU-1	Añadir filtros a LeaguesGrid	3h	Completada
T - 184	HU-1	Fix Standings	2h	Completada
T - 185	HU-9	Funcionalidad de LogIn	4h	Completada
T - 186	HU-8	Funcionalidad de SignUp	4h	Completada
T - 187	HU-2	Añadir paginación a TeamGrid	2h	Completada
T - 188	FIX	Style Fixes	1h	Completada
T - 189	HU-2	Añadir búsqueda a TeamGrid	2h	Completada
T - 190	HU-3	Añadir búsqueda y filtros a FixtureList	4h	Completada
T - 191	HU-2	Refactorizar de la vista Team	1h	Completada
T - 192	FIX	Fix TeamStatistics	2h	Completada
T - 193	HU-11	Crear vistas de usuario e interacción	4h	En Progreso
T - 194	DEV	Añadir persistencia a la store	1h	Completada
T - 195	DEV	Internacionalización de las cadenas de texto añadidas en el Sprint 12	1h	Completada
Total			**50h**	

Tabla 6.13: Tareas establecidas para el sprint 12

6.2.5. Sprint 13

Este es el último sprint del proyecto. En el se ha terminado la funcionalidad de las vistas de usuario y se ha construido la página *home* (Sección 7.2.1).

También se ha trabajado en corregir pequeños errores y se ha realizado la traducción final al español.

Por último se ha trabajado en finalizar las partes restantes de la documentación del proyecto.

Tarea	Tipo	Descripción	Horas	Estado
T - 196	DOC	Documentación del sprint 13	0.5h	Completada
T - 193	HU-11	Crear vistas de usuario e interacción	4h	Completada
T - 197	FIX	Fix de la rutas de Player	4h	Completada
T - 198	DEV	Crear Carrousel	2h	Completada
T - 199	DEV	Crear vista Home	4h	Completada
T - 200	FIX	Arreglar enlaces	2h	Completada
T - 201	DEV	Internacionalización de las cadenas de texto añadidas en el Sprint 13	0.5h	Completada
T - 202	DEV	Añadir traducción al español	3h	Completada
Total			**20**	

Tabla 6.14: Tareas establecidas para el sprint 13

Capítulo 7

Diseño

7.1. Model-View-ViewModel

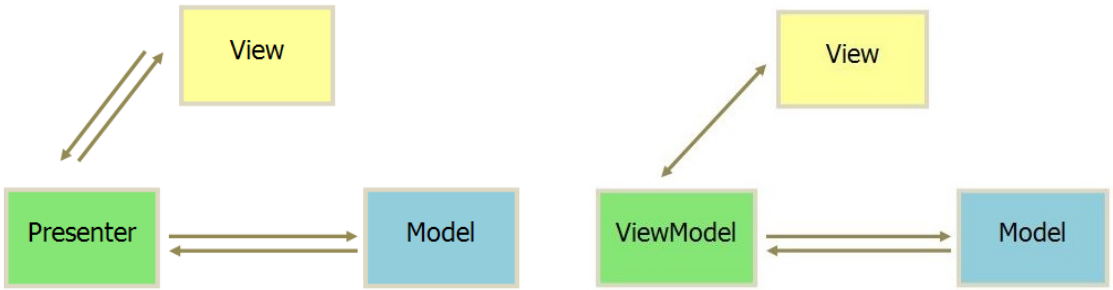
Como se indica en la documentación de Vue.js, el diseño de este framework se encuentra inspirado en el patrón arquitectónico Model-View-ViewModel (MVVM). Este patrón es una evolución del Modelo-Vista-Presentador (MVP) que es a su vez una derivación del patrón arquitectónico Modelo-Vista-Controlador (MVC). La principal intención de este patrón es tratar de desacoplar lo máximo posible la interfaz de usuario de la lógica de negocio de la aplicación. Para ello se sirve de las siguientes capas.

- **Modelo (*Model*)**: Representa la capa de datos y la lógica de negocio.
- **Vista (*View*)**: Es la capa encargada de representar la información de manera visual y capturar las interacciones de los usuarios.
- **Modelo de vista (*ViewModel*)**: El modelo de vista funciona a modo de intermediario entre el modelo y la vista y contiene toda la lógica de presentación.

La principal diferencia entre los patrones MVP y MVVM es que mientras que en el patrón MVP la comunicación entre la Vista y el Presentador, aun siendo bidireccional, se realiza a través de dos canales de comunicación diferentes de manera independiente (el presentador se comunica con la vista a través de un canal diferente al que utiliza la vista para comunicarse con él. Fig. 7.1a) en el patrón MVVM la Vista y el ViewModel se encuentran conectados a través de un único enlace de datos bidireccional (Fig. 7.1b) que permite que los cambios realizados en la vista se reflejen de manera automática en el ViewModel y viceversa permitiendo una comunicación mucho más ágil y sin intermediarios.

7.1.1. Vue.js y MVVM

Como he comentado anteriormente Vue.js permite implementar el patrón MVVM (Fig. 7.2) gracias a que la instancia Vue fue concebida para actuar a modo de modelo de vista proporcionando múltiples herramientas que permiten la comunicación bidireccional entre Vista (etiqueta template en *Single File Components*) y ViewModel (instancia Vue).



(a) Esquema MVP (b) Esquema MVVM

Figura 7.1: Comparativa entre MVP y MVVM [71].

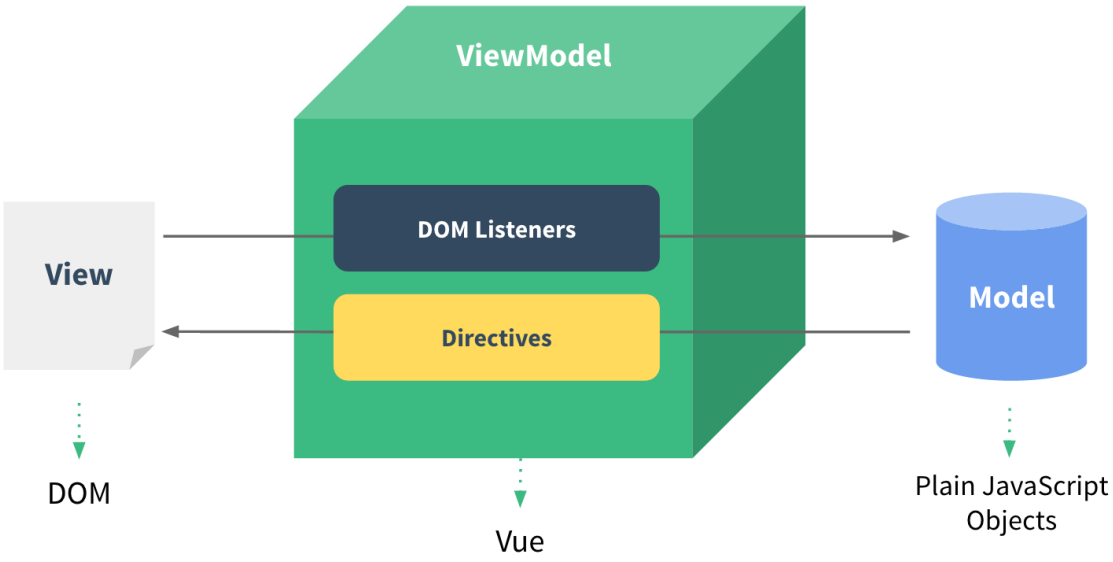


Figura 7.2: Implementación de MVVM en Vue [71].

Este patrón de diseño se ve reflejado en Vue en herramientas como las directivas o el *Double mustache binding* que se describirán a continuación.

- **Directivas**: Las directivas son atributos HTML especiales precedidos por la cadena "v-". La principal función de estas es realizar cambios sobre el DOM de manera dinámica cuando el valor de su expresión cambia. Algunas de las directivas Vue más destacadas son las siguientes.
  - **v-bind**: La directiva v-bind permite enlazar una variable de Vue con un atributo HTML. Es decir, se mantiene el atributo actualizado con la propiedad de la instancia.
  - **v-model**: La directiva v-model permite crear un enlace bidireccional entre un atributo HTML y una propiedad de la instancia. Es decir, si el valor del atributo HTML se modifica este cambio se verá reflejado inmediatamente en la propiedad de la instancia y viceversa.
  - **v-if (v-else/v-else-if)**: Estas directivas permiten la renderización condicional de elementos del DOM. Al igual que en la programación convencional, si la expresión se evalúa a *true* el elemento se mostrará, si se evalúa a *false* el elemento no se mostrará.

  - **v-for**: La directiva v-for permite realizar renderizados de lista. Al igual que en la programación convencional permite renderizar el mismo elemento repetidas veces (con o sin variaciones).
  - **v-on**: La directiva v-on permite capturar eventos del DOM y realizar acciones cuando estos suceden.
- **Double mustache binding**: El Double mustache binding es un tipo de sintaxis a través del cual podemos mostrar datos de forma dinámica en el DOM. Recibe este nombre porque se utilizan llaves dobles para delimitar los términos sobre los cuales queremos realizar el *binding* (p. ej. "$< p > \{\{variable\}\} < /p >$").

Además de las directivas comentadas anteriormente existe una gran variedad de directivas adicionales así como la posibilidad de crear nuestras propias directivas Vue personalizadas.

7.2. Atomic Design

El Atomic Design o Diseño Basado en Componentes es un patrón de diseño web que pretende acabar con las inconsistencias y optimizar al máximo el desarrollo de productos digitales [72] [73]. Este patrón consiste en descomponer las interfaces web en partes más pequeñas conocidas como átomos y reutilizarlas a través de la composición para generar estructuras más complejas hasta terminar formando la web final.

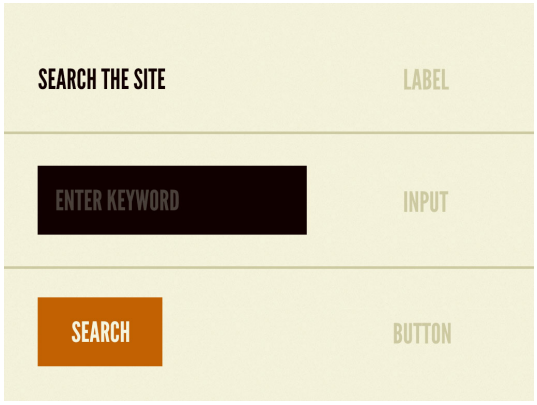
Las principales estructuras reconocidas por este patrón son las siguientes.

- **Átomos (*atoms*)**: Son los componentes básicos de la aplicación. Aplicado el diseño web serían aquellos elementos que tienen alguna funcionalidad por si solos como serían los botones, cards, inputs, labels, etc. (Fig. 7.3a).
- **Moléculas (*molecules*)**: Una molécula es la unión de dos o más átomos. En el diseño web esto se traduce por un conjunto de componentes como podría ser una composición formada por un título y un subtítulo o un campo de búsqueda junto al botón buscar. (Fig. 7.3b).
- **Organismos (*organisms*)**: Un organismo es un conjunto de moléculas. Estos elementos comienzan a ser más complejos y podrían ser cosas como una lista de avatares y sus nombres o un grupo de botones para crear una barra de navegación. (Fig. 7.3c).
- **Plantillas (*templates*)**: Las plantillas son mucho más concretas que los elementos mencionados anteriormente y proporcionan un contexto a dichos elementos. En el diseño web serían el equivalente a un *wireframe*, es decir, una especie de boceto del contenido final de la vista con *placeholders* en vez de la información real de la aplicación. (Fig. 7.3d).
- **Páginas (*pages*)**: Las páginas son las instancias finales concretas de las plantillas. En estas los *placeholders* de las plantillas se reemplazan por el contenido final de la página. (Fig. 7.3e).

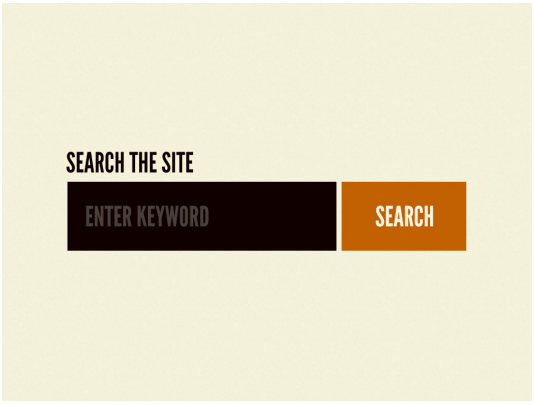
Se ha decidido utilizar el Atomic Design porque proporciona una metodología clara para llevar a cabo el diseño y desarrollo de sistemas. Además este patrón proporciona gran consistencia y escalabilidad a las aplicaciones.

7.2.1. Aplicación al proyecto

Para aplicar este patrón al proyecto se ha utilizado la funcionalidad aportada por vue para construir componentes de un único archivo (*Single File Components*). A continuación se detallará una lista de los componentes utilizados en el proyecto, así como un diagrama de componentes disponible en la figura 7.4.

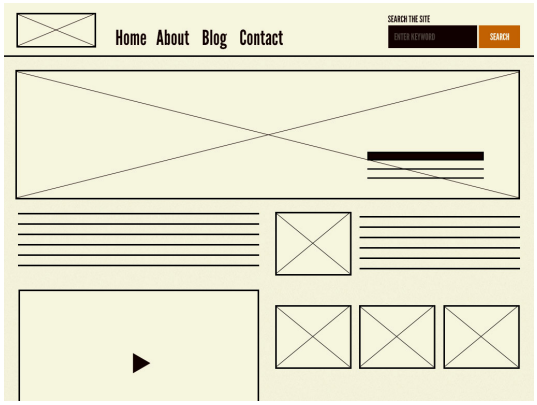


(a) Átomos

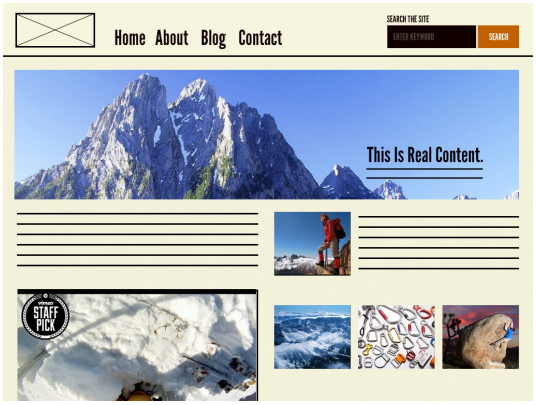


(b) Molécula

(c) Organismo



(d) Plantilla



(e) Página

Figura 7.3: Estructuras de Atomic Design [72].

Átomos

Los átomos son componentes básicos que sirven para crear el resto de componentes de mayor tamaño. Dentro de nuestra aplicación todos estos componentes básicos son los elementos utilizados de la biblioteca de componentes Vuetify.

Moléculas

En esta sección describiremos los principales componentes de la aplicación que se pueden catalogar como moléculas. Estos componentes son elementos de tamaño mediano que sirven como base para crear componentes mayores. Con el fin de simplificar el diagrama de componentes (Fig. 7.4) se ha decidido omitir estos componentes en la versión final del diagrama. Los componentes descritos son los siguientes.

- **CareerCard**: Proporciona la estructura para mostrar información sobre un equipo en el que ha trabajado un entrenador (nombre y escudo del equipo, fecha de inicio y fecha de fin).
- **CareerListItem**: Proporciona la estructura para mostrar un elemento de la carrera de un entrenador en forma de lista.
- **CoachCard**: Proporciona la estructura para mostrar una carta de entrenador.
- **CoachListItem**: Proporciona la estructura para mostrar información de un entrenador en forma de lista.
- **CountryCard**: Proporciona la estructura para mostrar una carta de país.
- **Country**: Proporciona la estructura para mostrar un enlace a un país determinado formado por el nombre del país y su bandera.
- **FixtureListHeader**: Proporciona la estructura para crear una barra de búsqueda y filtros dentro de una lista de partidos.
- **FixtureListItem**: Proporciona la estructura para mostrar partidos en forma de lista.
- **LineupList**: Proporciona la estructura para generar una tabla de alineaciones.
- **LineupListItem**: Proporciona la estructura para mostrar un determinado jugador dentro de una lista de alineaciones.
- **StatsBar**: Proporciona la estructura para generar una barra de estadísticas dentro de un partido.
- **TimeLine**: Proporciona la estructura para generar una *timeline* a partir de los eventos de un partido.
- **LeagueGridCard**: Proporciona la estructura para mostrar una carta de liga vertical.
- **LeagueCard**: Proporciona la estructura para mostrar una carta de liga horizontal.
- **LineupField**: Proporciona la estructura para generar un campo de fútbol para visualizar alineaciones.
- **LineupPlayer**: Proporciona la estructura para mostrar un jugador dentro del campo de fútbol de alineaciones.
- **PlayerCard**: Proporciona la estructura para mostrar una carta de jugador.
- **PlayerList**: Proporciona la estructura para mostrar una lista de jugadores.
- **PlayerListItem**: Proporciona la estructura para mostrar un jugador dentro de una lista de jugadores.

- **PlayerStatisticsTable**: Proporciona la estructura para mostrar las estadísticas de un jugador.
- **TransferCard**: Proporciona la estructura para mostrar una carta de traspasos de un jugador.
- **TransferListItem**: Proporciona la estructura para mostrar un traspaso de jugador dentro de una lista.
- **Standings**: Proporciona la estructura para mostrar un *ranking* de una liga.
- **TeamCard**: Proporciona la estructura para mostrar una carta de equipo.
- **TeamListItem**: Proporciona la estructura para mostrar un equipo dentro de una lista de equipos.
- **TeamStatisticsTable**: Proporciona la estructura para mostrar una tabla de estadísticas de un equipo.
- **TeamTransferListItem**: Proporciona la estructura para mostrar un elemento de lista de un traspaso dentro de un equipo.
- **VenueCard**: Proporciona la estructura para mostrar una carta de estadio.
- **TrophieListItem**: Proporciona la estructura para mostrar un elemento de lista de trofeos.
- **LocaleSwitcher**: Proporciona un sistema para cambiar dinámicamente el idioma de la aplicación.
- **AvatarMenu**: Proporciona la estructura del menú de avatar de la aplicación.

Organismos

Son elementos complejos (generalmente compuestos por múltiples moléculas) que cuentan con funcionalidad propia y sirven como base para las páginas. Dentro del diagrama de componentes (Fig. 7.4) son los elementos marcados con color amarillo. Los componentes descritos son los siguientes.

- **PlayerStatistics**: Proporciona la estructura para mostrar información sobre las estadísticas de un determinado jugador.
- **TransferGrid**: Proporciona la estructura para mostrar los traspasos entre equipos de un determinado jugador.
- **PlayerHeader**: Proporciona la estructura de la cabecera de la vista Player.
- **TrophieTable**: Proporciona la estructura de la tabla de trofeos que puede ser utilizada en la vista Player y la vista Coach
- **SidelinedTable**:Proporciona la estructura de la tabla de lesiones y sanciones que puede ser utilizada por la Vista Player y la vista Coach.
- **CareerGrid**: Proporciona la estructura para visualizar los distintos equipos en los que ha trabajado un entrenador.
- **CoachHeader**: Proporciona la estructura de la cabecera de la vista Coach.
- **Events**: Proporciona la estructura para visualizar los eventos de un partido.
- **FixtureHeader**: Proporciona la estructura de la cabecera de la vista Fixture.
- **Head2Head**: Proporciona la estructura para mostrar una lista de partidos cara a cara.
- **Lineups**: Proporciona la estructura para visualizar las alineaciones de un partido.

- **MatchStatistics**: Proporciona la estructura para visualizar información relativa a las estadísticas de un partido.
- **CountryGrid**: Proporciona la estructura para mostrar una cuadricula de países.
- **TeamTransferList**: Proporciona la estructura para mostrar los traspasos de un equipo.
- **PlayerGrid**: Proporciona la estructura para mostrar una cuadricula de jugadores.
- **CoachGrid**: Proporciona la estructura para mostrar una cuadricula de entrenadores.
- **TeamStatistics**: Proporciona la estructura para mostrar las estadísticas de un equipo.
- **TeamHeader**: Proporciona la estructura de la cabecera de la vista Team.
- **TeamGrid**: Proporciona la estructura para mostrar una cuadricula de equipos.
- **TeamLeagues**: Proporciona la estructura para mostrar las ligas de un equipo.
- **CountryHeader**: Proporciona la estructura de la cabecera de la vista Country.
- **CountryLeagues**: Proporciona la estructura para visualizar las ligas de un país.
- **TopPlayers**: Proporciona la estructura para visualizar los jugadores más destacados.
- **StandingsList**: Proporciona la estructura para mostrar un *ranking* de equipos.
- **LeagueHeader**: Proporciona la estructura de la cabecera de la vista League.
- **FixtureList**: Proporciona la estructura para mostrar una lista de partidos.
- **TeamList**: Proporciona la estructura para mostrar una lista de equipos.
- **HorizontalScrollContainers**: Proporciona la estructura para mostrar distintas tarjetas en un contenedor de *scroll* horizontal.
- **FixtureCarrousel**: Proporciona la estructura para mostrar información sobre distintos partidos.
- **LeagueGrid**: Proporciona la estructura para mostrar una cuadricula de ligas.
- **UserHeader**: Proporciona la estructura de la cabecera de la vista User.
- **EditUserCard**: Proporciona un formulario para editar los datos de un usuario.
- **SignUpCard**: Proporciona un formulario de inicio de sesión.
- **SignInCard**: Proporciona un formulario de registro.

Plantillas

Debido al diseño de Vue, este tipo de componentes resulta un poco redundante dentro de la aplicación, ya que serían las páginas detalladas anteriormente pero separando la funcionalidad de comunicación con la API y la base de datos de el diseño visual de la página. Por ello se ha decidido omitir este tipo de componentes, dado que teniendo en cuenta la estructura de los *Single File Components* (separados en *template*, *script* y *style*) esta separación resulta intrínseca a cualquier vista Vue.

Páginas

En esta sección pasaremos a describir las páginas o vistas utilizadas en la aplicación. Para facilitar la catalogación de estas vistas se han separado los componentes en grupos dependiendo de su tipo que puede ser Football, User o Aplicación. En estos componentes se realiza toda la funcionalidad relativa a la comunicación con la API y la base de datos. Estas vistas se corresponden a los componentes marcados en naranja dentro del diagrama de componentes (Fig. 7.4). Los componentes descritos son los siguientes.

- **Football**: Categoría para agrupar a todas las páginas relativas al fútbol.
  - **Coach**: Página en la que se permite visualizar información relativa a entrenadores.
  - **Counties**: Página en la que se permite ver los distintos países disponibles en la aplicación.
  - **Country**: Página que permite visualizar información relativa a un país determinado.
  - **Fixture**: Página en la que se permite visualizar información sobre un partido.
  - **Leagues**: Página que permite visualizar todas las distintas ligas disponibles de la aplicación.
  - **League**: Página en la que se puede consultar información sobre una liga concreta.
  - **Player**: Página que permite consultar información sobre un jugador concreto.
  - **Team**: Página que permite visualizar información sobre un equipo concreto.
- **User**: Categoría para agrupar a todas las páginas relativas a la funcionalidad de usuario.
  - **User**: Página en la que se permite visualizar información relativa al usuario.
  - **EditUser**: Página en la que se permite editar las características del usuario.
- **Aplicación**: Categoría para agrupar a todas las páginas relativas a la aplicación.
  - **Home**: Página principal de la aplicación.
  - **SignUp**: Página de inicio de sesión.
  - **SignIn**: Página de registro.

Navegación

Los componentes de navegación podrían considerarse organismos, ya que comparten muchas de las características definidas para ese grupo de componentes. Sin embargo, debido a una característica de estos, heredada de la estructura de tipo SPA (*Single-page application*), he decidido catalogarlas en su propio grupo de componentes. A diferencia de los organismos, estos componentes se renderizan una única vez (siempre que no se produzcan cambios sobre ellos) y se colocan en la capa más alta de la aplicación a modo de marco, de manera que las páginas se van deslizando por debajo de estos a medida que se navega por la aplicación.

Dentro del diagrama de componentes (Fig. 7.4) son los elementos representados en color azul. Estos componentes son los siguientes.

- **NavigationHandler**: Es un componente que agrupa todos los componentes relacionados con la navegación. Gestiona los enlaces que se muestran de manera dinámica en las barras de navegación y los eventos de estas.
- **AppBar**: Es la barra superior de la aplicación. Contiene los enlaces de navegación dentro de la aplicación, los botones de login y signin y el botón de cambio de idioma.

- **NavigationDrawer**: Es la barra lateral izquierda de la aplicación. Esta se muestra en dispositivos de pequeño tamaño en los que los enlaces no entran dentro de la AppBar.
- **Footer**: Es la barra de navegación inferior de la aplicación contiene enlaces a redes sociales e información relevante sobre la web.

Raíz

Este componente constituye el elemento raíz de la aplicación. En el se colocan los componentes de navegación y se crea un contenedor donde se mostrarán los componentes referentes a vistas de la web. He decidido separarlo del resto de componentes porque no se alinea con ninguna de las estructuras descritas por *Atomic Design* ya que crea una estructura superior a las páginas.

- **App**: Componente raíz de la aplicación.

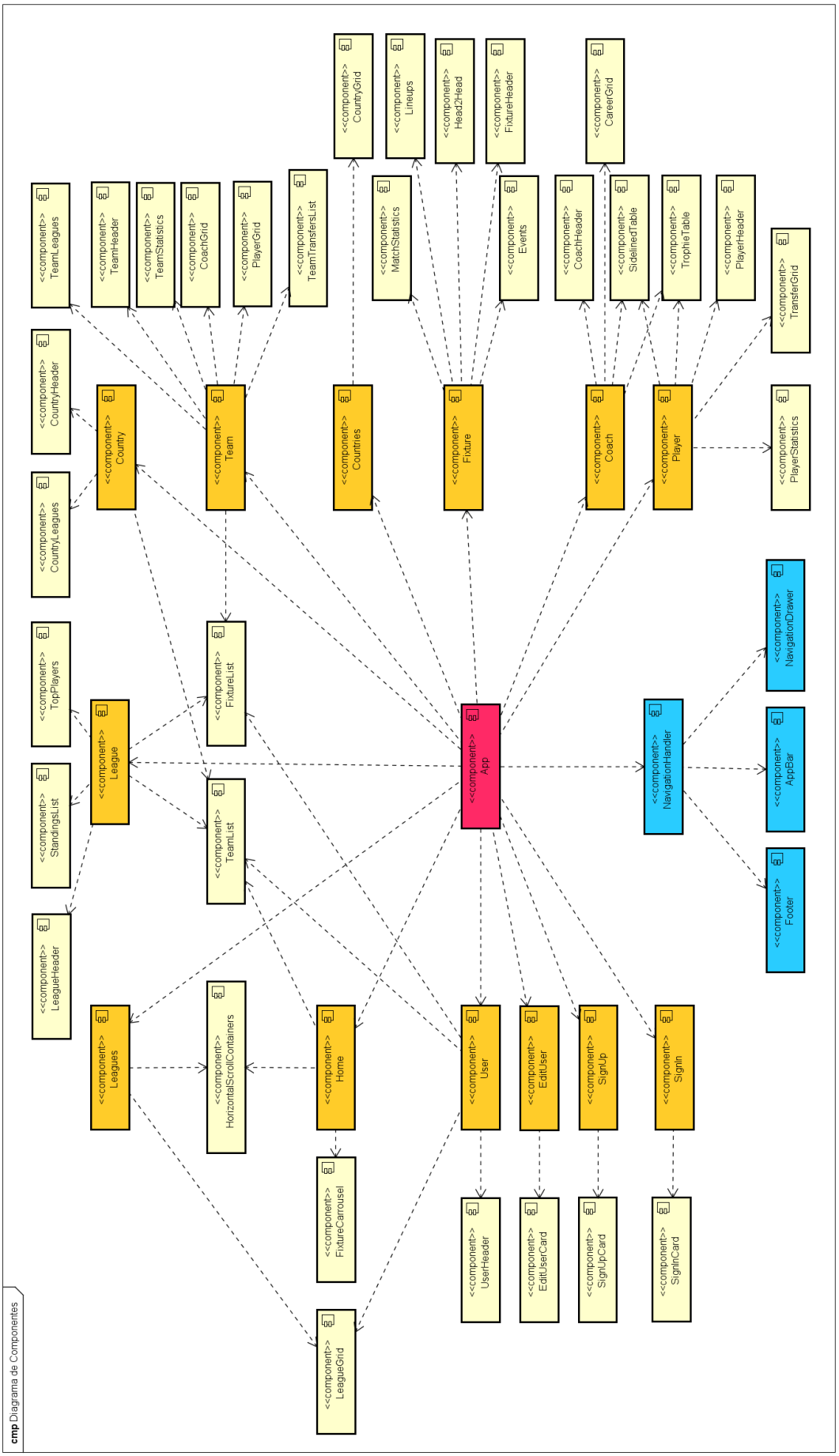


Figura 7.4: Diagrama de componentes.

7.3. Diseño de la interfaz de usuario

El objetivo de la interfaz de usuario diseñada es proporcionar una interfaz simple y de uso sencillo para todo tipo de usuarios. Por ello, el diseño de la aplicación se ha inspirado en las propuestas realizadas por otras alternativas del mercado que ya han demostrado ser efectivas, pero intentando generar una identidad propia.

7.3.1. Bocetos

Todos los bocetos aquí presentados son ideas previas al desarrollo de las vistas, por lo que el resultado final de algunas de ellas puede haber sido modificado tras su planteamiento. Por otra parte, estos bocetos son diseños muy simplificados de las vistas, construidos para tener una idea general del resultado final esperado, por lo que muchas de las características de las mismas (como textos y colores) han sido abstraídas del boceto.

Todas las imágenes que se van a mostrar a continuación se han realizado utilizando la herramienta MockFlow comentada en la sección 5.1.11. Se ha decidido que el diseño de gran parte de estas vistas sea similar para lograr tanto dar uniformidad al aspecto de toda la aplicación como para facilitar la accesibilidad de la misma, al tiempo que se reduce la cantidad de trabajo necesaria en su desarrollo.

Navegación

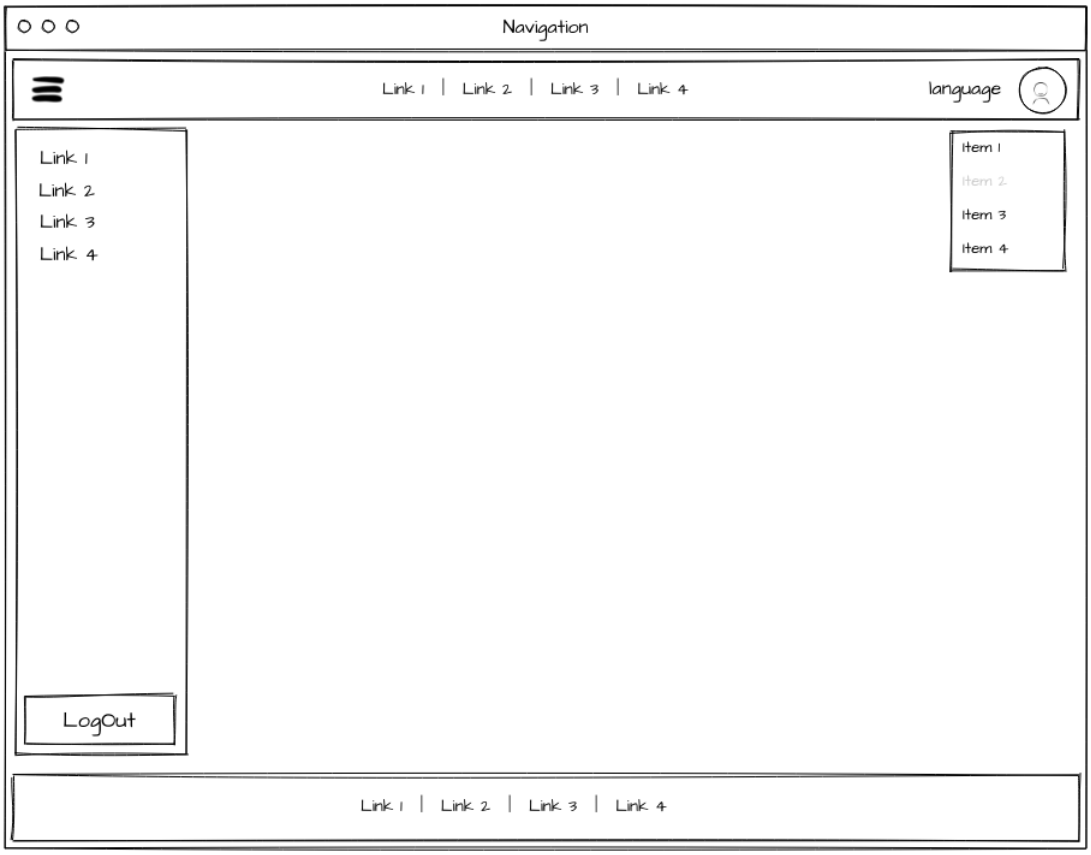


Figura 7.5: Boceto navegación.

El primer boceto en realizarse fue el de la navegación (Fig. 7.5). La idea de este boceto es mostrar la navegación dentro de la aplicación de manera independiente al contenido de las vistas. Esta cuenta con tres partes principales:

- AppBar: Barra superior de la aplicación. En ella se muestra el botón para abrir la barra de navegación, los enlaces de navegación dentro de la app, el componente para cambiar de idioma y dependiendo de si el usuario está logueado o no, botones de login y signin o un avatar de usuario que al pulsarlo despliega un menú con acciones de usuario.
- NavigationDrawer: Barra lateral izquierda de la aplicación. Se abre al pulsar sobre el botón de la barra principal de la aplicación y contiene los mismos enlaces de navegación que la AppBar.
- Footer: Barra inferior de la aplicación. En ella se muestran enlaces a redes sociales e información sobre la aplicación.

La principal influencia de diseño de las barras de navegación consiste en utilizar el principio de ubicar las cosas donde el usuario está acostumbrado a encontrarlas. Al no tratarse de un proyecto que planea ser rompedor en el medio podemos valernos de este principio para favorecer la accesibilidad de la página y la comodidad que aporta un entorno de navegación familiar.

Login & SignUp

Las vistas de login y SignUp son muy sencillas. Ambas se encuentran formadas por una tarjeta con su formulario correspondiente dentro. En los diseños se tiene en cuenta también que solo es posible acceder a estas vistas si no se está *logueado* en la aplicación (más información al respecto en la sección 8.2).

La vista de login (Fig. 7.6) contiene un formulario para acceder a la aplicación a través de un correo electrónico y contraseña.

La vista de SignUp (Fig. 7.7) contiene un formulario de registro. Este formulario está compuesto por 5 campos de texto (nombre completo, nombre de usuario, dirección de correo electrónico, contraseña y un campo de texto adicional para comprobar que la contraseña introducida es correcta), un selector de países y un *checkbox* para confirmar que se aceptan los términos y condiciones de nuestro servicio.

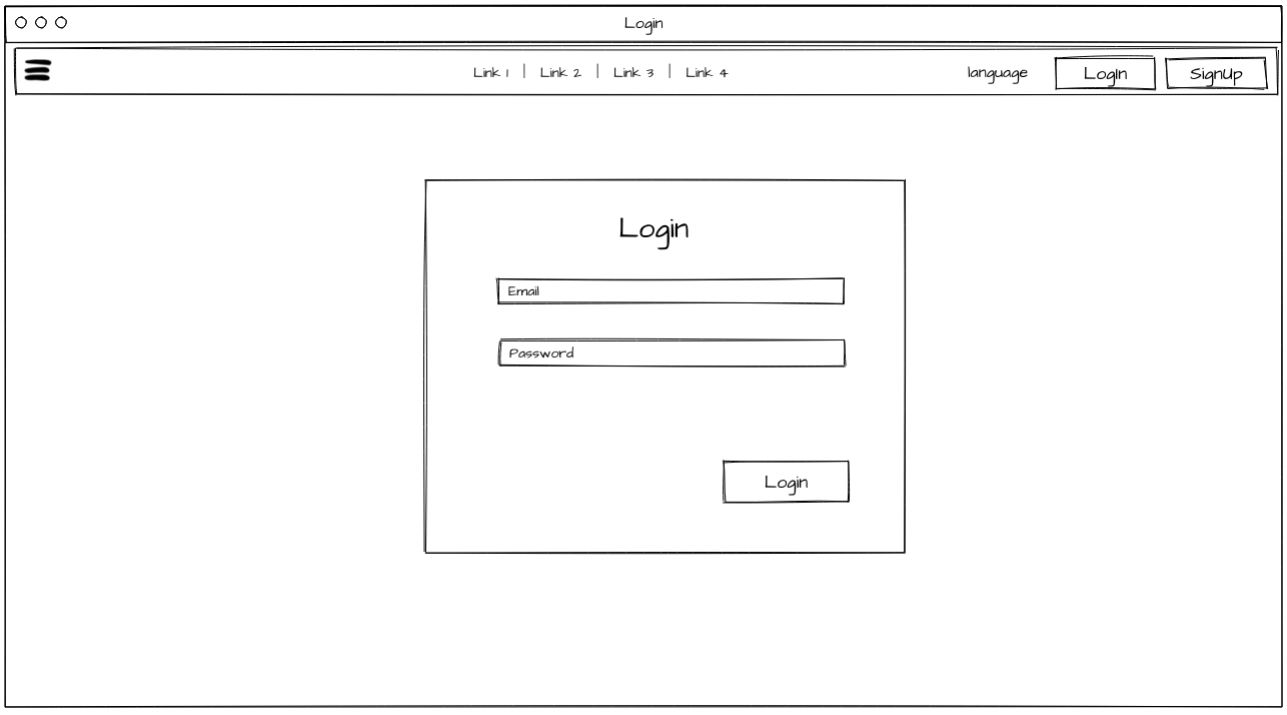


Figura 7.6: Boceto vista login.

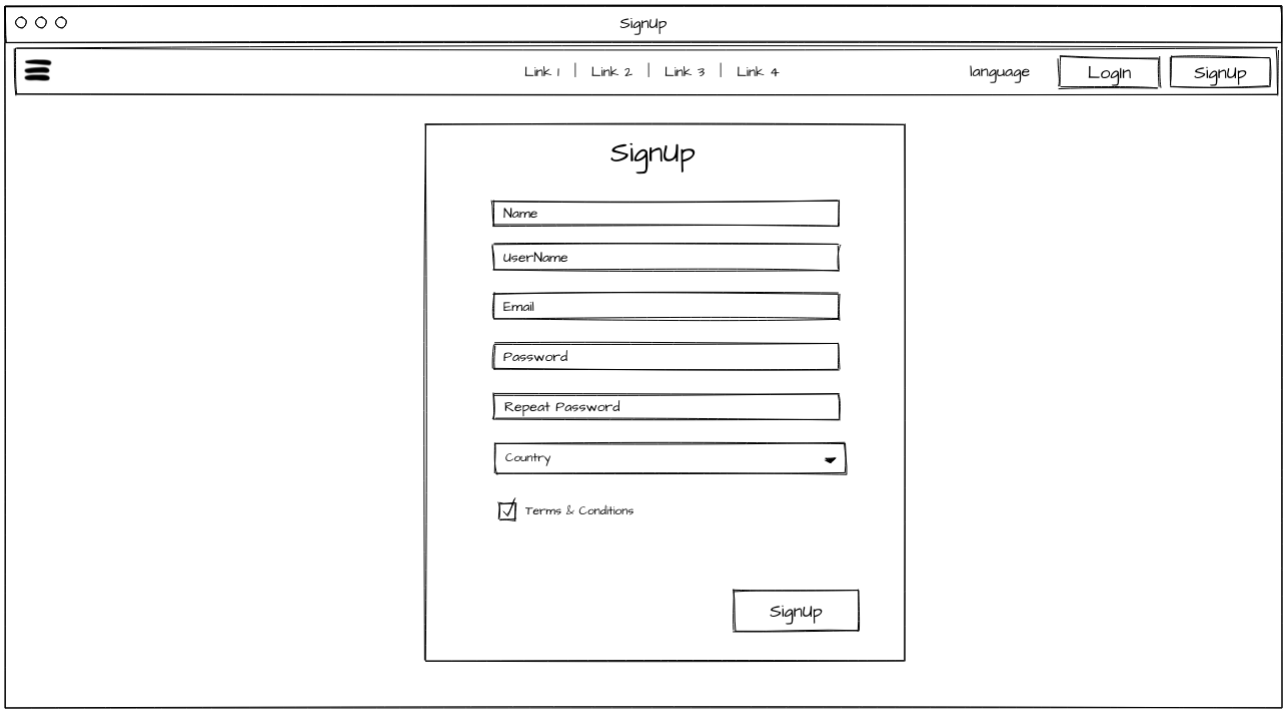


Figura 7.7: Boceto vista SignUp

Vista League

La vista League (Fig. 7.8) nos permite visualizar la información de una liga determinada. Esta vista se encuentra dividida en dos secciones: una cabecera, donde se muestra el logo de la liga, su nombre e información relativa a esta y un grupo de pestañas donde se mostrarán los componentes con información más detallada.

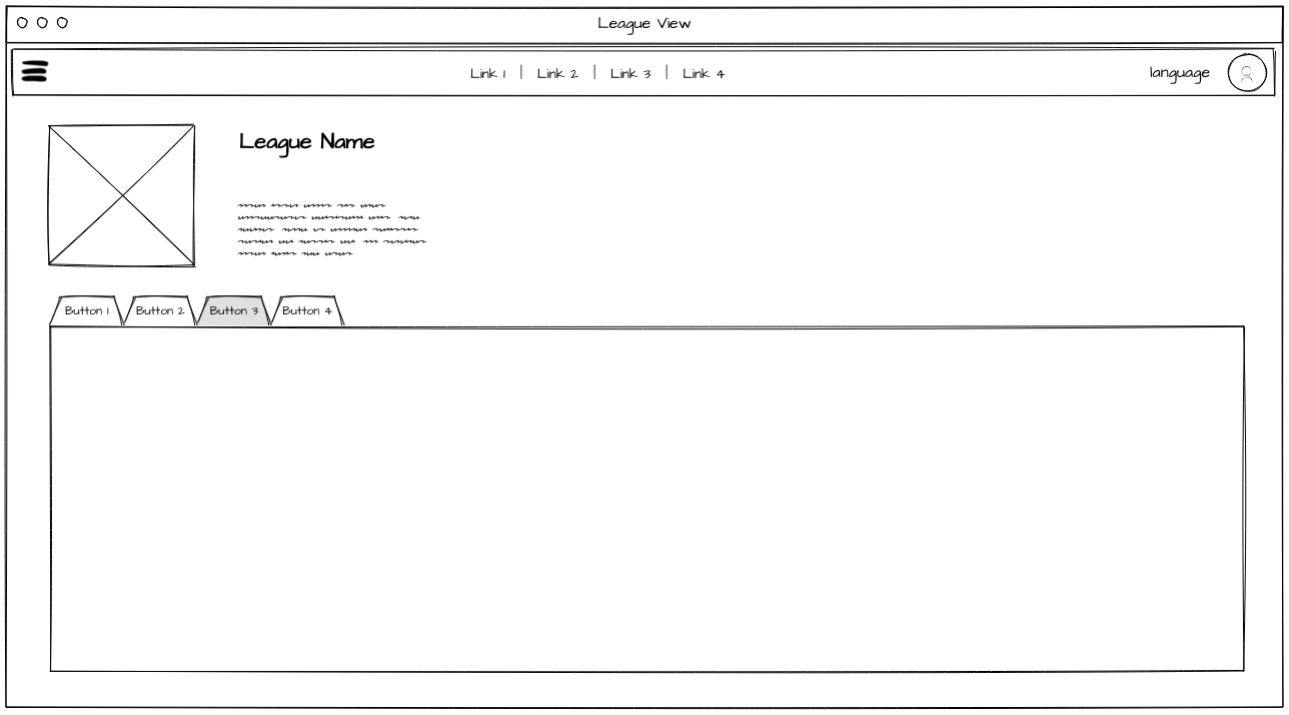


Figura 7.8: Boceto vista League.

Vista Fixture

La vista Fixture (Fig. 7.9) permitirá visualizar la información relativa a un partido. Al igual que la vista League (y que el resto de vistas de información deportiva) se encuentra dividida en dos secciones: la cabecera, mostrando los dos equipos que rivalizan, la fecha del partido, su estado y el resultado de este y el cuerpo de la vista, formado por un grupo de pestañas que contienen los componentes que utilizará.

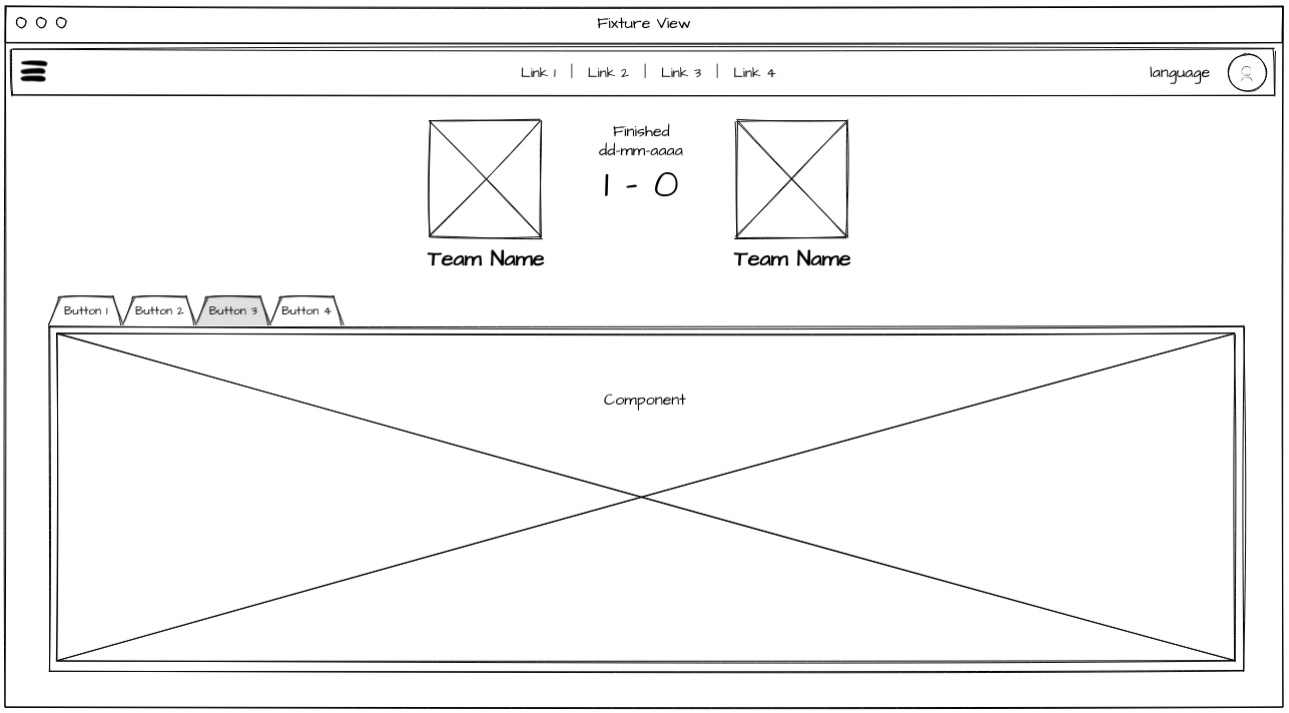


Figura 7.9: Boceto vista Fixture.

Vista Team

La vista Team (Fig. 7.10) permite visualizar información sobre un equipo determinado. Al lado izquierdo de la cabecera contiene el escudo, nombre y descripción del equipo mientras que al lado derecho de la cabecera contiene una tarjeta con información sobre el estadio del equipo. Debajo de la cabecera, en el cuerpo de la vista, se ha construido un grupo de pestañas que contendrán componentes específicos con información relacionada con los equipos de fútbol.

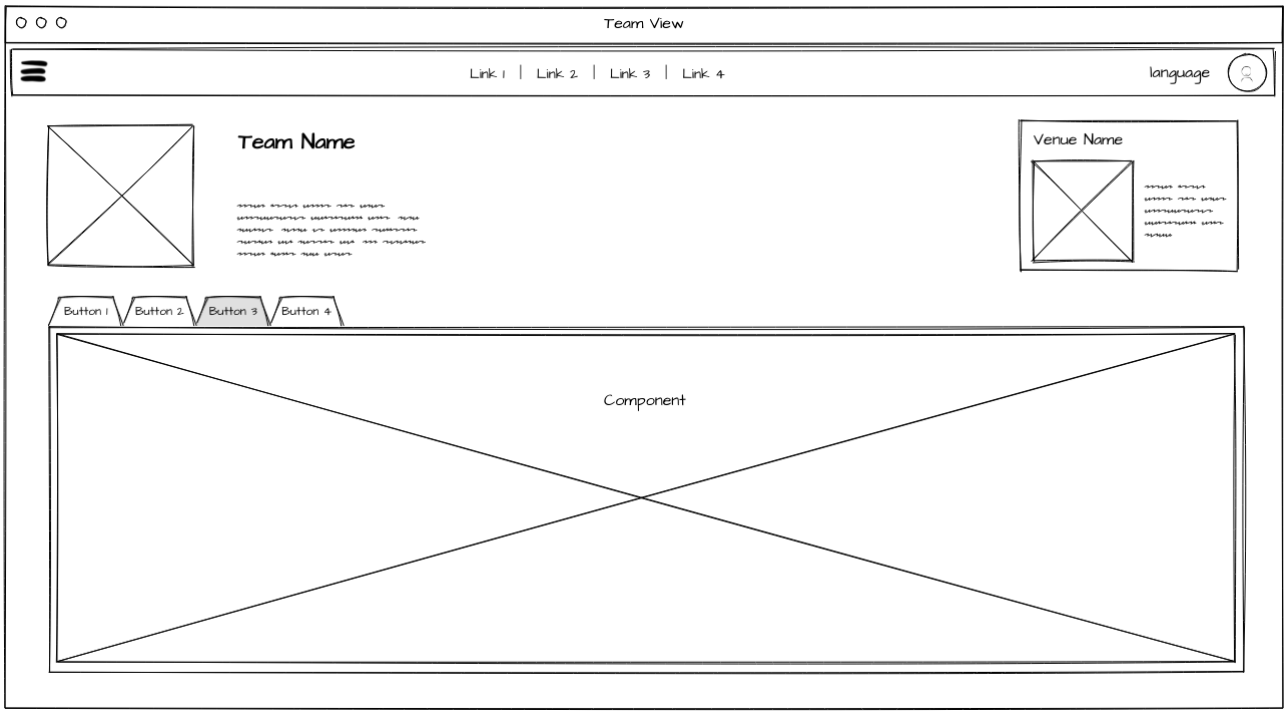


Figura 7.10: Boceto vista Team.

Vista Player

La vista Player (Fig. 7.11) permite visualizar información relacionada con jugadores de fútbol específicos. En la cabecera se muestra una imagen del jugador junto a su nombre y una breve descripción del mismo mientras que en la parte inferior se encuentra un sistema de pestañas con información más detallada sobre el jugador.

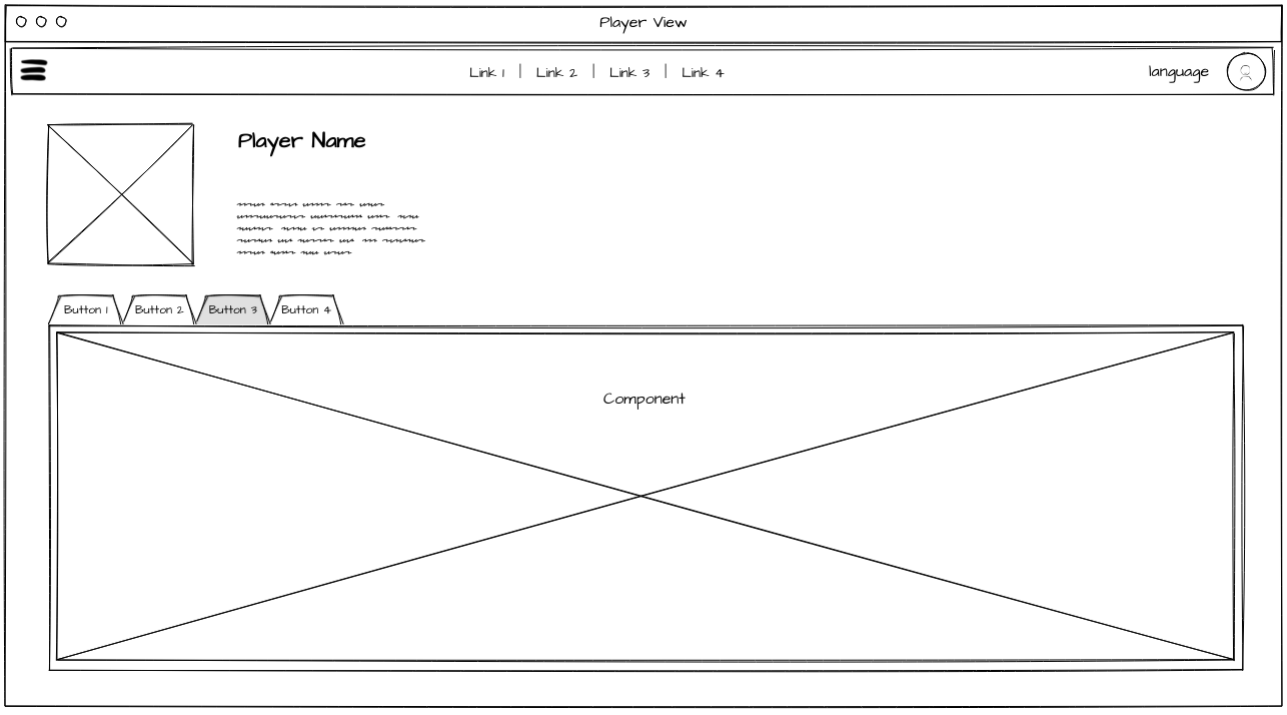


Figura 7.11: Boceto vista Player.

Vista Coach

La vista Coach (Fig. 7.12) permite consultar información relativa a entrenadores de fútbol. Al igual que en las anteriores vistas se ha dividido en cabecera y cuerpo. En la cabecera se muestra una imagen del entrenador, su nombre y su descripción. En el cuerpo, al igual que n todas las vistas anteriores, se encuentra un sistema de pestañas para organizar los componentes con información más específica relativos a un entrenador.

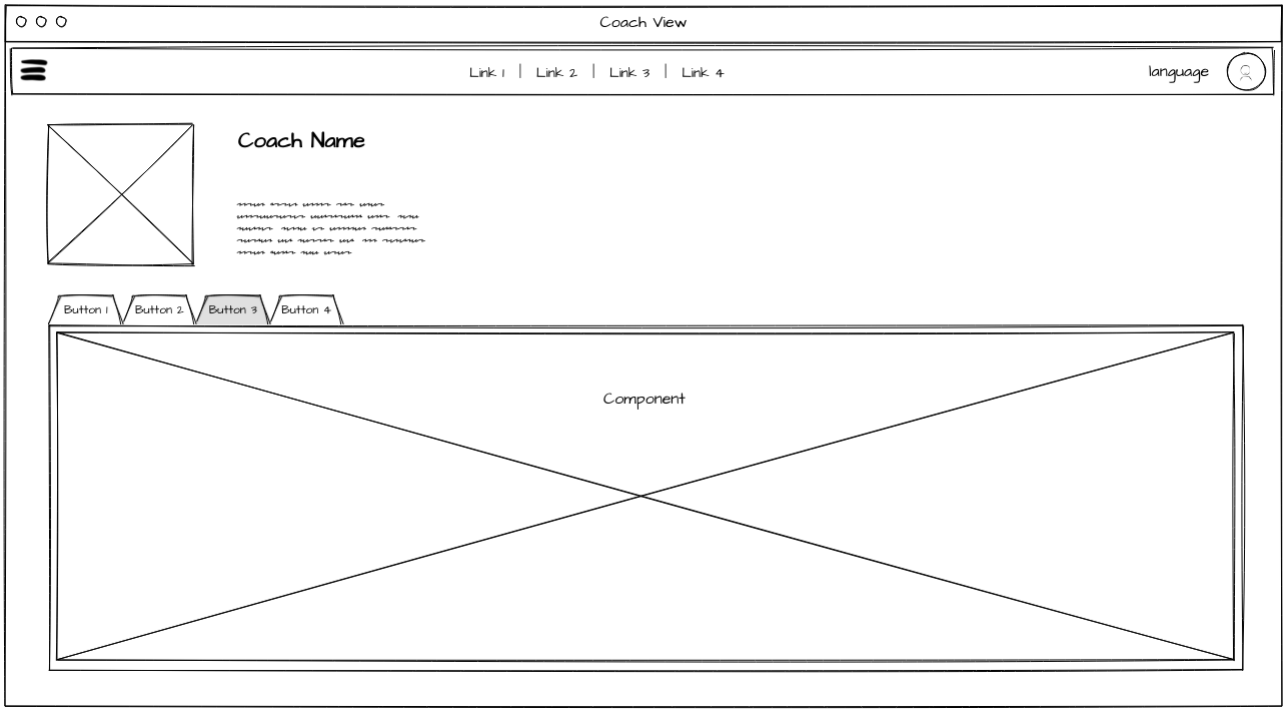


Figura 7.12: Boceto vista Coach.

Vistas de usuario

La vista de usuario (Fig. 7.11) es muy similar a todas las anteriores. También se encuentra dividida en cabecera y cuerpo donde la cabecera contiene información sobre el usuario y el cuerpo contiene un grupo de pestañas con información más detallada sobre las ligas, equipos y partidos que ha guardado el usuario.

El diseño de la vista de edición de perfil (Fig. 7.14) es similar al de registro ya que gran parte de su funcionalidad es similar. Este contiene una *card* con campos de texto para editar el nombre completo, nombre de usuario, contraseña y repetir contraseña además de un selector de países para modificar el país seleccionado en el registro.

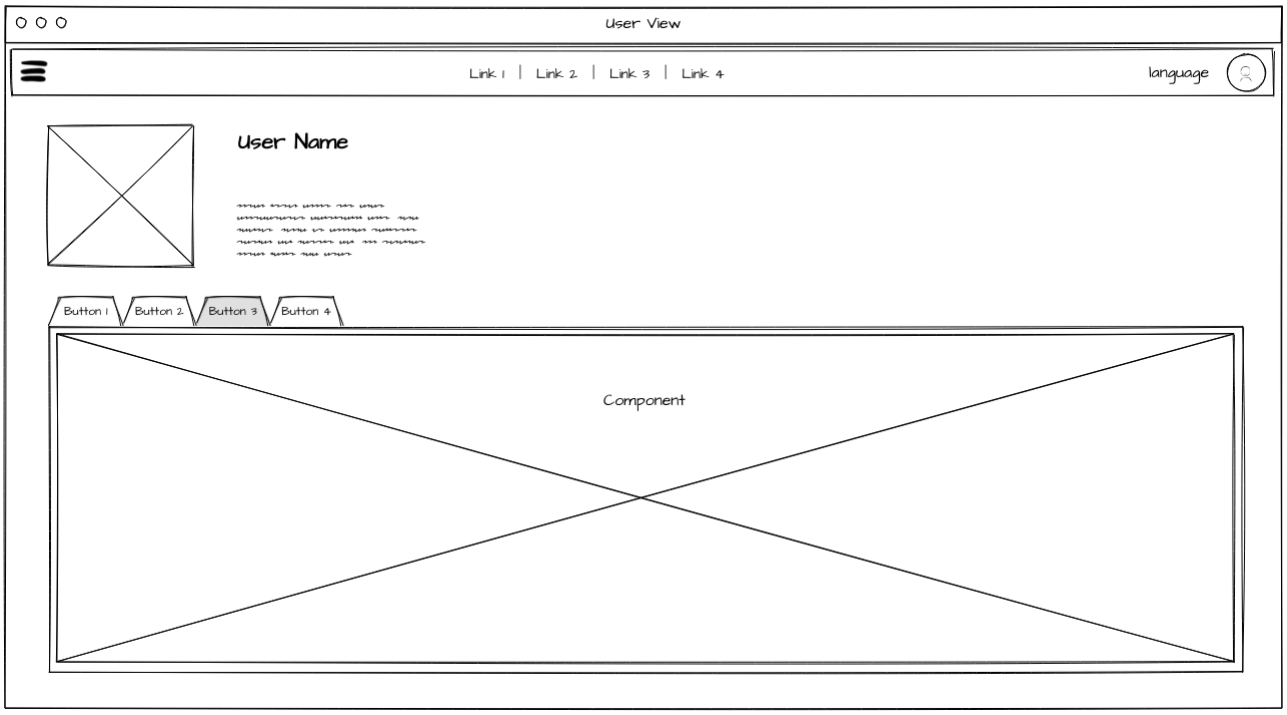


Figura 7.13: Boceto vista User.

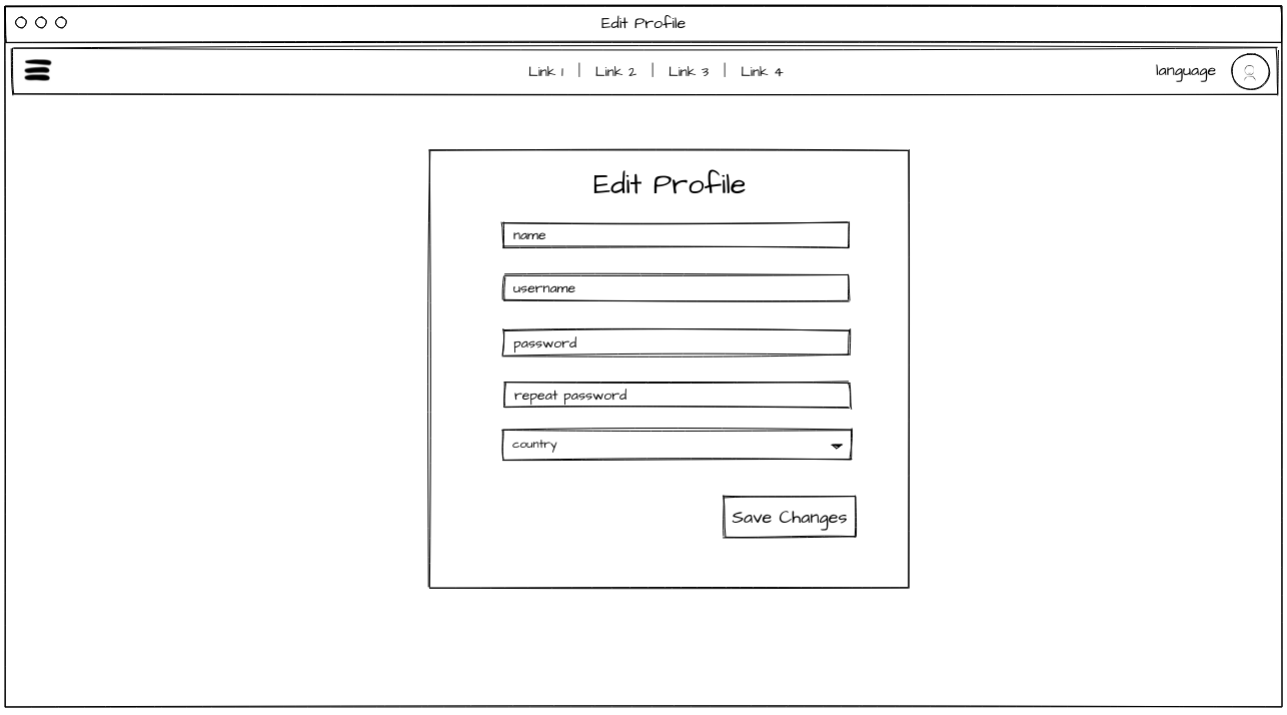


Figura 7.14: Boceto vista EditUser.

7.3.2. Paleta de Colores

Se ha decidido utilizar una paleta de 5 colores para la aplicación, porque mantener una paleta de colores simple además de facilitar las tareas de desarrollo, suele generar aplicaciones más intuitivas y homogéneas en su aspecto

[74]. Partiendo de que generalmente suele ser imprescindible utilizar los colores blanco y negro nos queda una paleta a escoger formada por un color primario y dos colores secundarios.

Como color primario se ha escogido el color *Bright Navy Blue* (segundo color de la paleta de colores de la figura 7.15). Se ha escogido este color porque, de acuerdo con la psicología del color [75] los colores azules suelen asociarse con la seriedad y la confianza que precisamente es lo que se busca transmitir con esta aplicación.

Como colores secundarios se ha optado por Maximum Blue Green y Red Munsell (tercer y cuarto colores respectivamente de la paleta de colores de la figura 7.15). Se han escogido estos dos colores porque al formar una triada con un tono anaranjado se encuentran a la misma distancia, de esta manera, al asignarlos a cada uno de los equipos de un partido se transmite su rivalidad pero sin presentar preferencias por ninguno de los dos equipos.

Por último se ha escogido el color blanco como color de fondo de la aplicación y el color negro como color de las fuentes de texto (a excepción de los textos que se encuentren sobre los colores primarios o secundarios en los que el color blanco favorece la proporción de contraste entre el texto y el fondo). También se han utilizado algunos tonos grises para romper la monotonía de estos dos colores sobre los fondos y los textos.

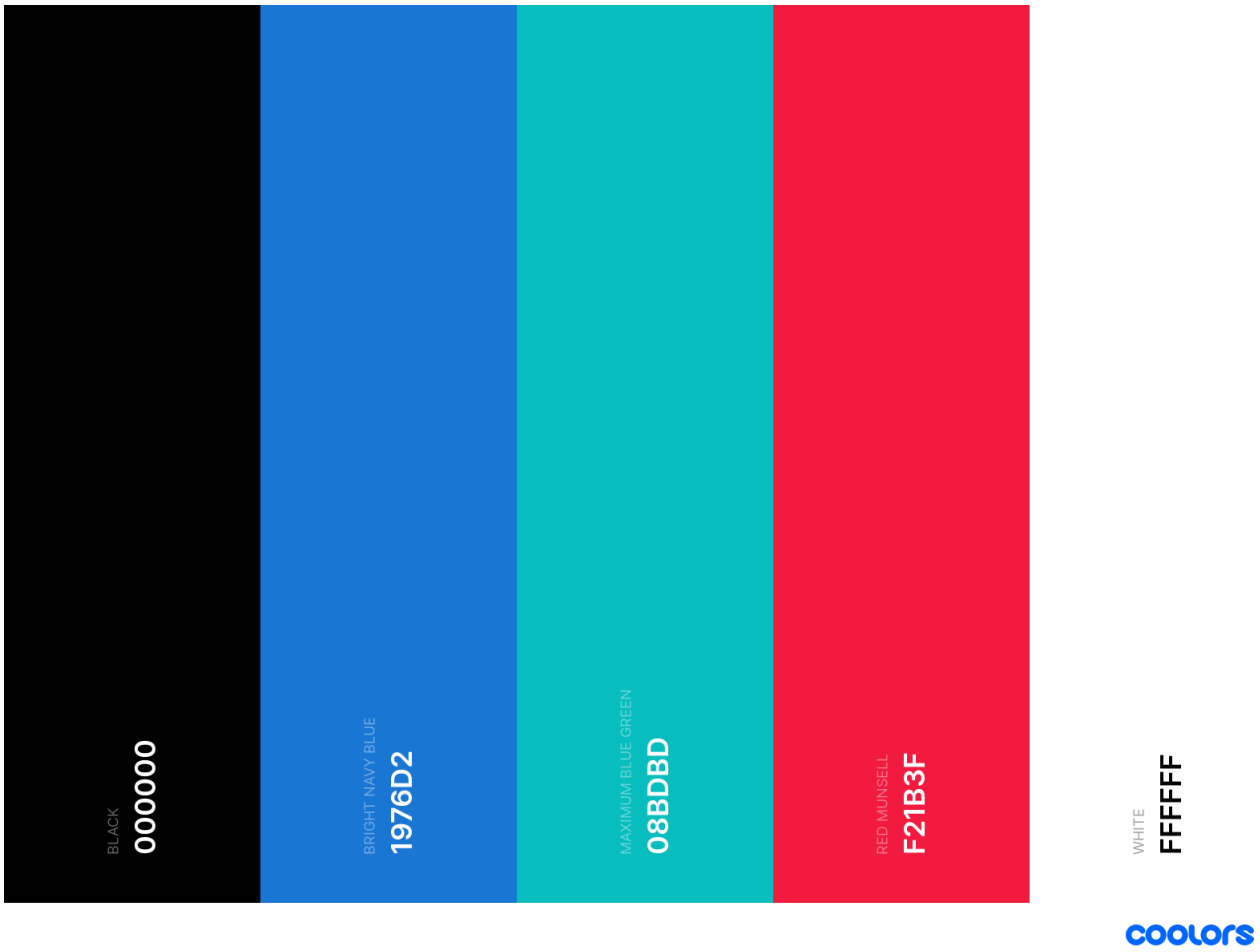


Figura 7.15: Paleta de Colores.

7.3.3. Fuentes Utilizadas

La fuente de texto utilizada para los textos de la aplicación es la fuente Roboto. Roboto es una familia de fuentes del tipo san-serif (sin serifas) diseñada por Google como fuente predeterminada para los sistemas operativos

Andorid. Esta es la fuente por defecto de Vuetify y se ha decidido mantenerla en la aplicación por su legibilidad (en pantallas las tipografías sin serifas se ven más limpias que aquellas con adornos) y porque al ser la fuente utilizada por Google resulta más familiar y cómoda de leer respecto otras fuentes.

7.4. Diseño de la base de datos

7.4.1. Diseño inicial

Durante en inicio del desarrollo se realizó un diseño conceptual de la estructura de la base de datos necesaria para el proyecto, ya que antes de decidir utilizar una API para los resultados deportivos se planteó recabar dicha información a través de *web scraping*. Este diseño inicial (disponible en la figura 7.16) cubre únicamente los aspectos relacionados con resultados deportivos y no los de interacción de usuarios.

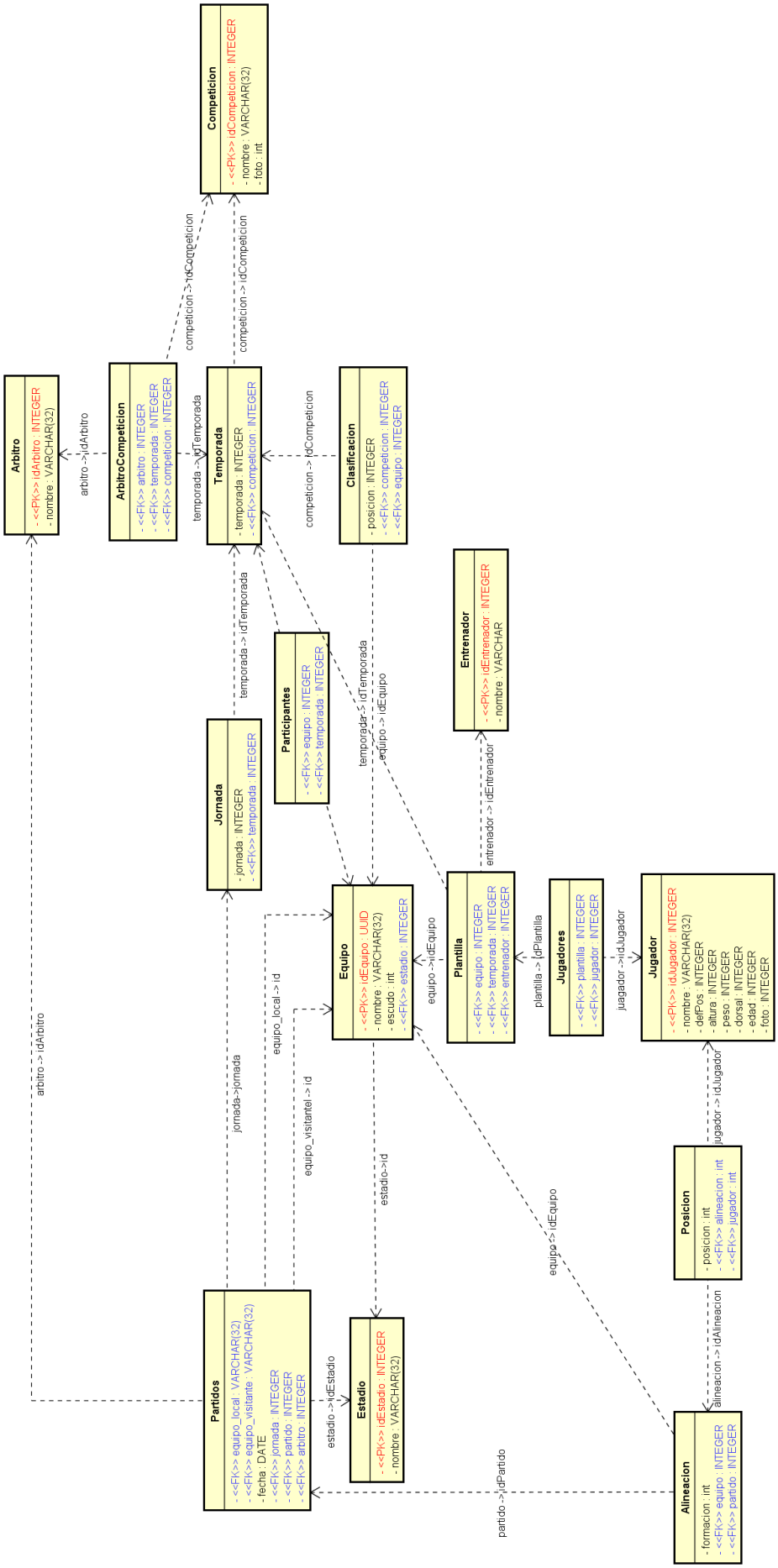


Figura 7.16: Diagrama conceptual DB de fútbol.

7.4.2. Diseño Firebase

También se ha realizado un diseño de la base de datos creada para la interacción de los usuarios con la aplicación mas allá de la visualización de los datos (Fig. 7.17). Esta base de datos se ha creado utilizando el servicio de bases de datos en tiempo real Firestore proporcionado por Firebase.

Firestore es un servicio de base de datos en tiempo real con una estructura no relacional. Esto quiere decir que en vez de centrarse en las relaciones entre tablas se encuentra organizado en Colecciones y Documentos. Las colecciones son agrupaciones de documentos mientras que los documentos son archivos con formato JSON que pueden almacenar cualquier información (no es necesario que dos Documentos tengan la misma estructura interna para que pertenezcan a la misma colección). Un documento puede a su vez contener una colección de documentos (sub-colecciones) con lo que se pueden generar estructuras similares a las proporcionadas por una base de datos relacional pero de manera más sencilla.

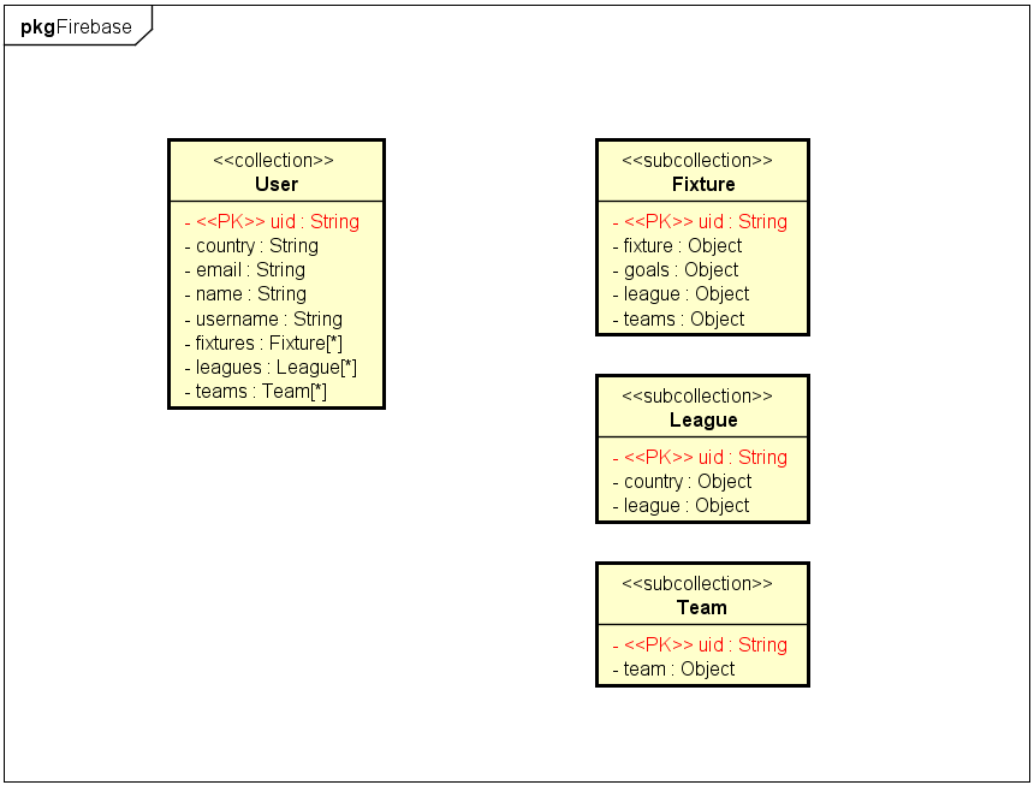


Figura 7.17: Diagrama conceptual DB de usuarios.

Aunque este diseño de base de datos se podría haber simplificado aun más almacenando únicamente listas de ids de los partidos, ligas y equipos se ha decidido copiar estos objetos enteros a nuestra propia base de datos para intentar ahorrar peticiones a la API de manera que estas se hagan solo cuando sean estrictamente necesarias.

7.5. Modelo de despliegue de la web

El despliegue de la aplicación se ha realizado utilizando el servicio de *hosting* proporcionado por Firebase. Por ello se ha elaborado un diagrama de despliegue (Fig. 7.18) con el fin de mostrar el resultado esperado tras el despliegue de la aplicación.

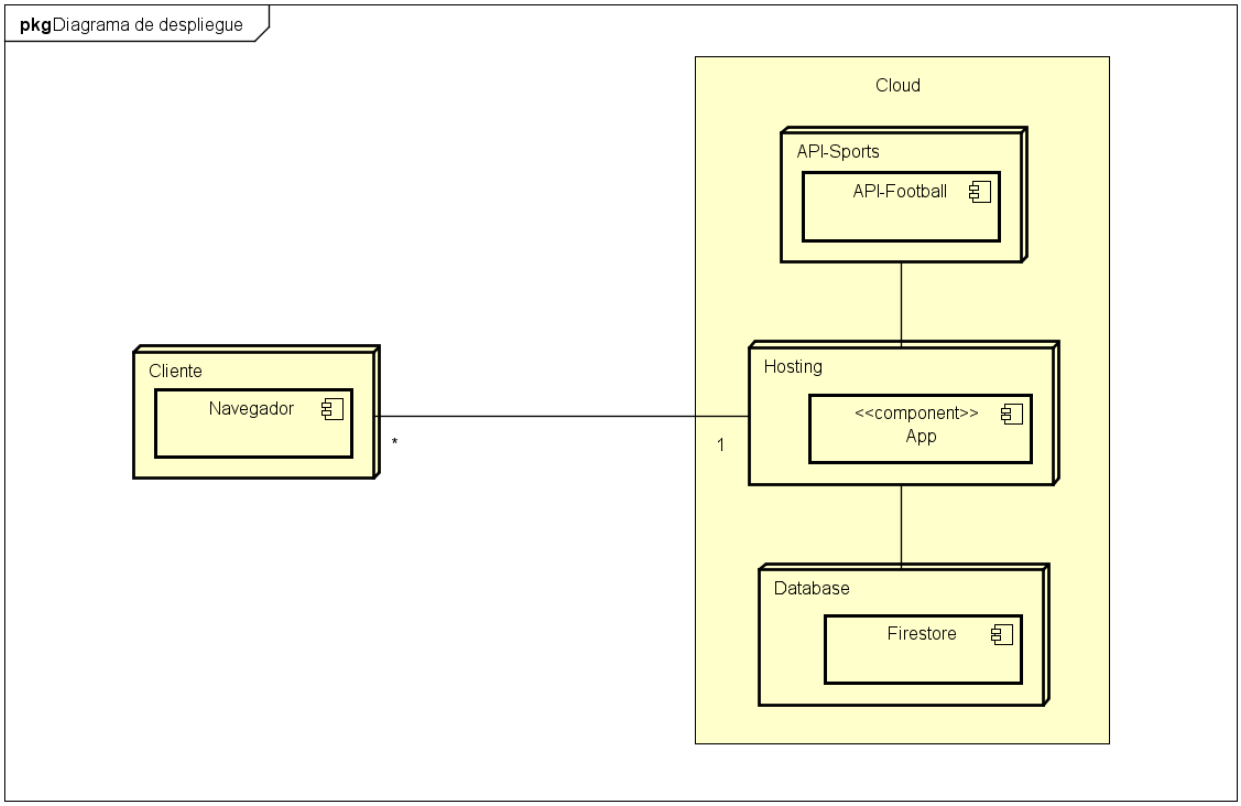


Figura 7.18: Diagrama de despliegue.

Capítulo 8

Implementación

En este capítulo se tratará de comentar la implementación del proyecto comenzando por los detalles técnicos relativos a la estructura del mismo y posteriormente detallando todos los cambios surgidos durante la etapa de desarrollo respecto al diseño planteado en el capítulo 7.

8.1. Estructura del directorio del proyecto

A continuación se muestra la estructura de ficheros del proyecto:

```
- Estructura del directorio del proyecto -

WatchFootball
│   .editorconfig
│   .env.development
│   .env.mock
│   .env.production
│   .eslintignore
│   .gitignore
│   babel.config.js
│   cypress.json
│   LICENSE
│   package-lock.json
│   package.json
│   README.md
│   vue.config.js
│
├──dyson
│   │   index.js
│   │
│   ├──assets
│   └──responses (Omitido por brevedad)
│
├──node_modules
├──public
│       favicon.ico
```

```
        index.html

├──src
    │   App.vue
    │   i18n.js
    │   main.js
    │
    ├──api
    │   │   axiosDefault.js
    │   │
    │   ├──firebase (Omitido por brevedad)
    │   │
    │   └──logic (Omitido por brevedad)
    │
    ├──assets (Omitido por brevedad)
    │
    ├──components (Omitido por brevedad)
    │
    ├──locales
    │       en.json
    │       es.json
    │
    ├──plugins
    │       firebase.js
    │       firebaseConfig.js
    │       vuetify.js
    │
    ├──router
    │       index.js
    │
    ├──store
    │   │   index.js
    │   │
    │   └──modules
    │           user.js
    │
    ├──styles
    │       variables.scss
    │
    └──views (Omitido por brevedad)

└──tests
    ├──e2e
    │
    └──unit
```

La estructura de ficheros presentada anteriormente es una derivación de la estructura inicial creada por Vue CLI surgida a partir de las distintas necesidades manifestadas durante la etapa de desarrollo.

En el directorio raíz del proyecto nos encontramos inicialmente con el fichero `.editorconfig` (fichero de configuración del IDE) y los distintos ficheros de entorno (`.env`) para cada una de las distintas etapas de desarrollo del proyecto: desarrollo, producción y mock (este último surge de la necesidad de iniciar la aplicación en modo de

pruebas para alguna de las tareas de desarrollo). Posteriormente nos encontramos con los ficheros `.eslintignore` y `.gitignore` que contienen una serie de reglas con las que se puede omitir el funcionamiento de los programas ESLint y Git respectivamente sobre ciertos ficheros de nuestro proyecto. También contamos con los ficheros de configuración de babel (`babel.config.js`), cypress (`cypress.json`), NPM (`package.json`, `package-lock.json`), y Vue (`vue.config.js`). Por último, en la estructura raíz del proyecto encontraremos el fichero `README.md` de nuestra aplicación que es auto-generado por Vue CLI y modificado en función de las propiedades del proyecto y el fichero `LICENSE` con el código legal de la licencia aplicada al proyecto.

A continuación realizaremos un desglose de las distintas carpetas contenidas en el directorio raíz de nuestro proyecto [76]:

- `dyson`: Contiene la configuración del servidor mock además de una carpeta "`responses`" con los distintos fichero JSON que simulan la respuesta de al API utilizada.
- `node_modules`: Este directorio contiene todas las dependencias NPM necesarias para el correcto funcionamiento de la app.
- `public`: Esta carpeta contiene todos los ficheros estáticos que no serán procesados por Vue. Esta contiene el fichero `index.html` que sirve como punto de entrada a nuestra aplicación y el fichero con el logo de la app que se mostrará en la pestaña del navegador en la que se encuentre nuestra app.
- `src`: En este directorio se almacena todo el código fuente de nuestro proyecto. En la raíz de este directorio se encuentran los ficheros `App.vue` (componente base de la aplicaci'on Vue), `i18n.js` (encargado de ejecutar la lógica de localización de la aplicación) y `main.js` (el fichero encargado de lanzar la aplicación y generar la instancia Vue inicial). Además, este directorio está formado a su vez por las siguientes carpetas:
  - `api`: Contiene la lógica de la comunicación entre la aplicación y las fuentes de datos (API-Football y Firebase).
  - `assets`: Contiene archivos estáticos utilizados en la aplicación como imágenes, vídeos, iconos, etc.
  - `components`: Dentro de este directorio se encuentran todos los componentes de nuestra aplicación.
  - `locales`: Contiene los ficheros JSON con las variables de localización del proyecto. En nuestro caso contiene dos ficheros: `en.json` y `es.json`.
  - `plugins`: Contiene los ficheros de configuración de los distintos plugins utilizados, en nuestro caso la configuración de Firebase (`firebase.js` y `firebaseConf.js`) y Vuetify (`vuetify.js`).
  - `router`: Contiene los archivos necesarios para manejar la navegación (cambio entre las distintas vistas) dentro de la aplicación. Contiene el fichero `index.js` que contiene información sobre las distintas vistas (su ruta, componente que utilizan, nombre, etc) y las reglas de acceso a cada una de ellas.
  - `store`: Contiene toda la lógica encargada de gestionar los estados de la aplicación.
  - `styles`: Contiene el fichero `variables.scss` en el que se establecen una serie de variables scss para utilizar en el estilo de las vistas de la app. De esta manera podemos centralizar cosas como los colores para que si en un futuro queremos realizar alguna modificación sobre la paleta de colores utilizada nos sea mucho más sencillo hacerlo.
  - `views`: Contiene todos los componentes de la aplicación que funcionan a modo de vista.
- `test`: Directorio que contiene los directorios para incluir los test unitarios y end-to-end de la aplicación.

8.2. Medidas de seguridad implementadas

Durante el desarrollo del proyecto se divisaron algunos puntos vulnerables de la aplicación que podrían ser utilizados para acceder a funciones no previstas durante la etapa de planificación.

Una forma común de vulnerar una aplicación web es a través de los formularios de registro e inicio de sesión. En ellos se utilizan los distintos campos de texto del formulario para introducir pequeños scripts en el sistema y esperar que estos se ejecuten al ser evaluados. La solución utilizada para lidiar con este problema es añadir una serie de reglas a cada formulario de manera que se compruebe que el contenido introducido en dicho campo es un contenido válido.

Otra posible vulnerabilidad de la aplicación es que ciertas páginas no deberían ser accesibles para usuarios no registrados. Por ejemplo, la página de edición de un usuario no debería ser accesible si dicho usuario no ha iniciado sesión. Gestionar este tipo de vulnerabilidades es posible gracias al plugin Vue Router (Sección 5.1.5) que permite incorporar comprobaciones antes de visitar ciertas páginas de manera que en el caso de que la comprobación no se cumpla el usuario sea redirigido a otra página distinta.

Por último, otro punto común de vulnerabilidad es la autenticación de los usuarios. Al tratarse de información privada es necesario manejar dicha información con precaución. Una de las manera más comunes de hacer esto es almacenar la información sensible de forma cifrada. Por suerte, la herramienta utilizada para la autenticación en la aplicación, Firebase Authentication (sección 5.1.4), tiene esto en cuenta y gestiona el cifrado de los campos de usuario de manera interna. También se han tenido en cuenta los principios aprendidos durante el grado, por los cuales, durante el registro solo se solicita información indispensable para el correcto funcionamiento de la funcionalidad planeada para la aplicación.

8.3. Decisiones tomadas a lo largo del proyecto

Durante el desarrollo del proyecto se realizaron una serie de decisiones que creemos que merecen ser destacadas.

La primera decisión importante que se tomo fue la de reemplazar un *back-end* propio para toda la información deportiva por una API externa que nos proporcionara toda la información necesaria para nuestro proyecto. Tras un tiempo investigando distintas formas de hacernos con los datos necesarios para el correcto desarrollo del proyecto nos encontramos con la API utilizada. Esta contaba con unas cuantas ventajas frente al resto de alternativas:

- La API cuenta con una versión gratuita que permite trabajar con ella con un límite de 100 peticiones diarias por lo que toda la fase de desarrollo se podría realizar utilizando la versión final de la API.
- La API cuenta con datos fiables que se mantienen actualizados a lo largo del tiempo.
- Los datos provienen de un suministrador oficial por lo que nos quitamos los posibles problemas legales que podrían conllevar otras opciones como el *web scraping*.
- La API cuenta con distintos planes de contratación sustancialmente más baratos que los de la competencia por lo que es posible adaptarse dependiendo de el número de visitas al proyecto.

Esta decisión también trae consigo una serie de desventajas asociadas, como la necesidad de adaptar nuestra aplicación la estructura ofrecida por la API, depender de una fuente externa al proyecto para que la aplicación pueda funcionar de manera correcta o la necesidad de pagar por mejores servicios en función del numero de peticiones que obtenga la web, por citar algunas.

Además, tras agregar una API externa fue necesario implementar un servidor mock que funcionara a modo de respaldo en caso de que las peticiones a la API fallaran o en el caso de que nos quedáramos sin peticiones antes de terminar una tarea. En estos casos, usamos este servidor para completarla.

También cabe destacar que la versión de la API utilizada se actualizó a mitad del proyecto ya que facilitaba el acceso a algunos datos que anteriormente eran difíciles de recuperar.

Otra decisión importante fue la de utilizar Firebase como plataforma de desarrollo. En ella se ha realizado el despliegue de la app y contiene la base de datos de usuarios del proyecto. Al utilizar una API externa como fuente de datos la mayor parte de la carga de trabajo se centra en el apartado *front-end* por lo que la necesidad de un contar con un *back-end* propio se reduce y puede ser remplazada por las herramientas propuestas por Firebase. Además de todo esto, Firebase cuenta con una gran extensibilidad y multitud de herramientas útiles para tareas de gestión, monitorización y prevención de amenazas.

Por último se ha utilizado Vuex como herramienta para gestionar el estado de la aplicación posibilitando que los distintos componentes puedan compartir información entre ellos de forma segura y controlada. También se ha utilizado el *SessionStorage* para mantener dicha información cargada mientras la sesión actual en el navegador permanece abierta.

8.4. Cambios realizados a lo largo del proyecto

8.4.1. Cambios en la interfaz de usuario

La interfaz de usuario ha sufrido una gran cantidad de cambios a lo largo del desarrollo del proyecto. Muchas de estas modificaciones no han sido sobre las vistas sino sobre los componentes de menor nivel de los cuales no se realizaron bocetos durante el diseño. De cualquier manera, muchas de estas pequeñas modificaciones se pueden observar en vistas como Fixture o Team donde se realizaron cambios en las cabeceras de dichas vistas por lo que si es posible visualizar las diferencias entre la versión original y la versión final.

8.4.2. Cambios en implementación del modelo conceptual

Como se ha comentado anteriormente, el modelo conceptual de dominio relativo a usuarios se vio modificado al incorporar Firebase al proyecto. Firebase proporciona servicios de base de datos de tipo NoSQL, por lo que no es posible implementar una base de datos con una estructura relacional. Todo esto se puede ver en la sección 7.4.2 del capítulo sobre el diseño.

8.5. Licencia

Este proyecto se distribuye bajo una licencia Creative Commons de tipo Atribución/Reconocimiento-NoComercial-SinDerivados [77], según la cual es posible copiar y redistribuir el material siempre y cuando se atribuya al autor de manera adecuada, dicho material no se utilice con fines comerciales y no se distribuyan copias del mismo si ha sido modificado de algún modo.

Capítulo 9

Pruebas

Con la intención de asegurar el correcto funcionamiento de la aplicación web se han llevado una serie de pruebas de software sobre el código implementado.

9.1. Introducción

Las pruebas de software consisten en verificar la calidad de un producto software a partir de métodos empíricos. Esto se consigue gracias a una serie de tests formalizados que se ejecutan dentro de un entorno de pruebas controlado de manera que los resultados obtenidos sean objetivos y reproducibles.

Existen distintas formas de clasificar las pruebas de software en función de sus características [78].

- Según su forma de interactuar con el software:
  - **Pruebas Estáticas**: Son aquellas que se realizan sin ejecutar el código de la aplicación.
  - **Pruebas Dinámicas**: Son aquellas que necesitan de la ejecución del software testeado para que se puedan llevar a cabo.
- Según su tipo de ejecución:
  - **Pruebas manuales**: Son aquellas pruebas que necesitan de la presencia de un individuo para su desarrollo.
  - **Pruebas automáticas**: Son todas aquellas pruebas que se sirven de un determinado software especialmente diseñado para la ejecución de las mismas.
- Según su enfoque:
  - **Pruebas de Caja Blanca**: Se centran en probar el funcionamiento interno de sistemas software.
  - **Pruebas de Caja Negra**: Se centran en testear un sistema software a partir de sus entradas y salidas, sin tener en cuenta el funcionamiento interno de este.
- Según lo que verifican:
  - **Pruebas Funcionales**: Verifican toda la funcionalidad propuesta por el sistema software.
    - **Pruebas unitarias**: Son pruebas que consisten en comprobar el correcto funcionamiento de una unidad de código.

  - **Pruebas de componentes**: Son pruebas que consisten en verificar el funcionamiento de un componente software.
  - **Pruebas de integración**: Estas pruebas buscan comprobar el correcto funcionamiento de todos los distintos componentes software trabajando en conjunto e interactuando entre sí.
  - **Pruebas end-to-end**: Estas pruebas se centran en testear acciones competas de los posibles usuarios dentro de la plataforma.
- **Pruebas No Funcionales**: Verifican requisitos no relacionados con la funcionalidad (requisitos no funcionales) que pueden utilizarse para juzgar el sistema.
  - **Pruebas de compatibilidad**: Son las pruebas que verifican el funcionamiento del sistema en diferentes entornos (un ejemplo en el caso de aplicaciones web sería probarlas en distintos navegadores).
  - **Pruebas de seguridad**: Las pruebas de seguridad son aquellas que tratan de buscar posibles brechas de seguridad en un sistema con el fin de prevenirlas y corregirlas.
  - **Pruebas de estrés**: Consisten en probar el software con cantidades de carga crecientes hasta alcanzar el punto en el que el sistema colapsa de manera que se pueda conocer dicho punto y la forma que tiene el sistema de lidiar con el colapso.
  - **Pruebas de usabilidad**: Las pruebas de usabilidad se enfocan en medir la capacidad de un producto de cumplir el propósito para el cual fue diseñado.
  - **Pruebas de rendimiento**: Las pruebas de rendimiento consisten en medir los tiempos de respuesta de un producto software bajo unas condiciones específicas de manera que se pueda medir la capacidad de reacción de la aplicación ante determinados eventos.

9.2. Pruebas en este proyecto

Como se propone en la sección sobre pruebas del manual de uso de Vue.js [79], en este proyecto se ha trabajado en implementar pruebas *end-to-end* y pruebas de componentes.

9.2.1. Pruebas de Componentes

Las pruebas de componentes se ha realizado de manera manual. La intención inicial era utilizar la herramienta Storybook (Sec. 5.2.1) para las tareas de diseño y prueba de los componentes. Lamentablemente, debido a problemas de compatibilidad durante la configuración se dejó de lado esta herramienta y se procedió a probar dichos componentes de manera manual.

Para poder realizar las pruebas de componentes de modo satisfactorio, independientemente del *famework* utilizado, es necesario montar dichos componentes dentro del DOM (ya sea de manera real o virtual) y así asegurar que los componentes se están renderizando de la manera esperada.

En nuestro caso particular, para poder probar el funcionamiento de estos componentes se ha habilitado una vista especial en la se montan los componentes que están siendo testeados de manera que se puedan realizar las pruebas para verificar el correcto renderizado de los componentes. En el caso de haber podido continuar utilizando la herramienta Storybook este testeo se habría realizado montando cada uno de los componentes en un DOM virtual y se habría comprobado el correcto funcionamiento de estos gracias a las herramientas proporcionadas por Storybook.

En este proyecto, estas pruebas se han realizado sobre todos y cada uno de los distintos componentes de la aplicación una vez se habían diseñado e implementado.

9.2.2. Pruebas End-to-end

A modo de complemento del testeo de componentes también se ha realizado un testeo *end-to-end* de las partes más importantes de la aplicación.

Este tipo de pruebas trata de simular la interacción de un usuario con un sistema informático a través de la realización de distintas tareas previstas para el uso final de la aplicación. Estas tareas suelen estar relacionadas con las historias de usuario descritas en la sección 3.2.2.

Para este tipo de pruebas se ha utilizado la herramienta de de testeo automatizado Cypress (comentada en la sección 5.1.5). Esta herramienta permite construir conjuntos de pruebas automatizadas que verifiquen funciones concretas de una aplicación. Para ello se ejecuta un navegador que es manejado a partir de reglas y acciones muy sencillas y como describen en la página inicial del framework, semejantes al lenguaje natural.

Las principales desventajas de este framework es que la ejecución de este tipo de pruebas es bastante lenta (sobre todo para tareas complejas) y que para su ejecución es necesario el correcto funcionamiento de la aplicación que se está probando, incluyendo las interacciones con la base de datos. Por estas razones, algunas de las pruebas elaboradas solo pueden ejecutarse de manera correcta una única vez (como por ejemplo las tareas de registro). Estos errores podrían solucionarse utilizando distintas bases de datos independientes para cada una de las etapas del desarrollo, pero debido a las limitaciones de recursos de este proyecto no ha sido posible implementarlas.

El numero de pruebas creadas alcanza un total de 29, divididos en 8 grupos distintos. Con esto se ha conseguido cubrir algunas de las partes más importantes de la aplicación (principalmente todas las acciones relacionadas con la interacción entre la base de datos y la web).

A continuación se muestra una descripción de todas las pruebas realizadas y los grupos a los que pertenecen:

- **Fixture**:
  - El botón “Add Fixture” no se muestra sin iniciar sesión.
  - El botón “Add Fixture” se muestra con la sesión iniciada.
  - Añadir partido correcto.
  - Eliminar partido correcto.
- **Forgot Email**:
  - Enviar email correcto.
  - Enviar email error.
- **League**:
  - El botón “Add League” no se muestra sin iniciar sesión.
  - El botón “Add League” se muestra con la sesión iniciada.
  - Añadir liga correcto.
  - Eliminar liga correcto.
- **Localization**:
  - Cambio de idioma correcto.
- **SignIn**:

  - SignIn correcto.
  - LogOut correcto.
  - SignIn error en el email.
  - SignIn error en la contraseña.
  - Visitar la página de SignUp desde la página de SignIn.
  - Iniciar sesión desde la barra de navegación correcto.

- **SignUp**:
  - SignUp correcto.
  - Pulsar el botón “Already Registered” redirige al usuario a la página SignIn.
  - Pulsar el botón “Agree” del texto de términos y condiciones marca la casilla.
  - Pulsar el botón “Disagree” del texto de términos y condiciones no marca la casilla.

- **Team**:
  - El botón “Add Team” no se muestra sin iniciar sesión.
  - El botón “Add Team” se muestra con la sesión iniciada.
  - Añadir equipo correcto.
  - Eliminar equipo correcto.

- **User**:
  - Editar usuario correcto.
  - Editar email correcto.
  - Editar email error ya en uso.
  - Editar contraseña correcto.

Capítulo 10

Conclusiones

10.1. Conclusiones

Una vez concluido el proyecto se considera que se han alcanzado todos los objetivos planteados al inicio del mismo, implementando de manera correcta una aplicación web totalmente funcional y con utilidades similares a las webs más importante del mercado. Se ha logrado conectar con una API externa, así como poder desplegar la aplicación en un servidor público. La aplicación final ofrece todas las funcionalidades inicialmente planteadas.

A continuación se muestra un desglose de los hitos logrados:

- Se ha construido una aplicación web de tipo SPA (*Single-page application*) utilizando el framework *Vue.js*.
- La aplicación muestra información de ligas, equipos, partidos, jugadores y entrenadores.
- La aplicación permite a los usuarios registrarse e iniciar sesión.
- La aplicación permite a los usuarios registrados guardar sus ligas, equipos, y partidos favoritos.
- La aplicación mantiene su información actualizada gracias a la conexión realizada con la API.
- La visualización se adapta a toda clase de públicos.

Con respecto a los objetivos personales, también se consideran alcanzados:

- Se han ampliado los los conocimientos sobre el funcionamiento de los *frameworks* de JavaScript (en concreto Vue.js), sus herramientas, plugins externos de utilidad y funcionamiento interno.
- Se han mejorado mis habilidades en el desarrollo *front-end*.
- He entendido la verdadera importancia que tiene la correcta planificación de un proyecto y la utilización de una metodología para el correcto desarrollo del mismo, así como la importancia que estas técnicas tienen a la hora de facilitar el trabajo en equipo.
- Se han ampliado mis conocimientos en relación al diseño de interfaces, conociendo nuevas técnicas y como cosas tan sencillas como las fuentes o colores pueden tener un enorme impacto sobre la calidad visual final de una aplicación.

- He conseguido mejorar mis habilidades en el uso de herramientas que ya conocía como HTML5, CSS o JavaScript.
- He tenido la oportunidad de trabajar con plataformas de desarrollo como Firebase por primera vez y ver la cantidad de herramientas y oportunidades que estas ofrecen.
- Se ha aprendido a trabajar con fuentes de datos externas.

Personalmente, este proyecto ha supuesto todo un reto ya que se ha estudiado de cero una nueva tecnología que no conocía con anterioridad y se ha utilizado para desarrollar un proyecto completo de inicio a fin por primera vez. Enfrentarse a un proyecto de estas características por primera vez y hacerlo en solitario supone todo un desafío, pero al final, tras verlo finalizado, supone una experiencia muy gratificante y considero que ha potenciado mis habilidades más allá de lo que habría esperado.

10.2. Lineas Futuras

Tras haber finalizado este proyecto aun existen algunas características en las que este podría mejorar. Algunas de estas características son las siguientes.

- Incluir un buscador dentro de la web.
- Mayor interacción entre la web y los usuarios.
- Agregar interacción entre usuarios.
- Mejorar visualización de la aplicación para dispositivos pequeños.
- Encontrar alguna solución al problema de la limitación en las peticiones.
- Incluir funcionalidad relativa a la visualización de partidos en directo.
- Mejorar el funcionamiento de la aplicación en distintos navegadores.

Apéndice A

Manual de usuario

En este apéndice veremos una pequeña guía de uso de la aplicación y realizaremos un recorrido por todas las distintas vistas disponibles en la misma.

A.1. Vista Home

Esta es la página principal de la aplicación, es la primera página visible al entrar en la web. En ella se muestra un carrusel con partidos en vivo, una sección de ligas destacadas y una cuadricula de equipos españoles. El resultado final de esta vista se puede observar en la figura A.1.

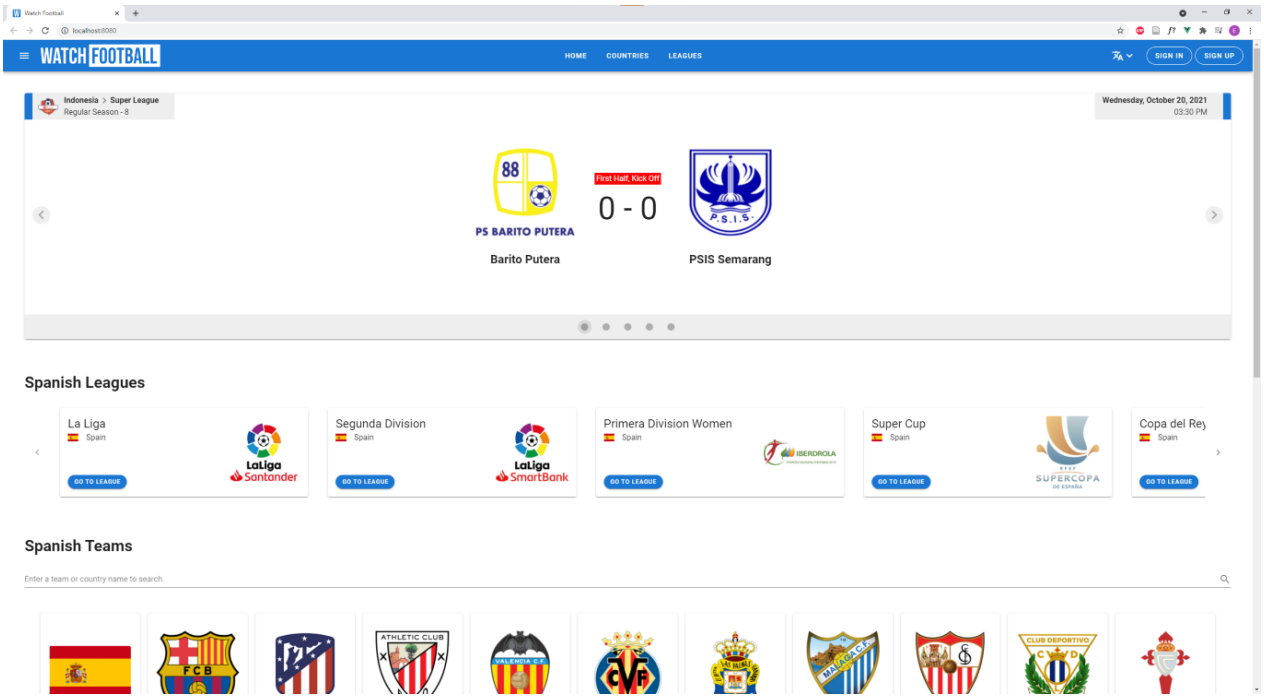


Figura A.1: Apariencia final de la página Home.

A.2. Vista Countries

En esta página se pueden visualizar los distintos países disponibles en la web. Esta vista también cuenta con una herramienta a partir de la cual podemos filtrar el país buscado a través de su nombre o código de país. El resultado final de la vista se puede observar en la figura A.2.

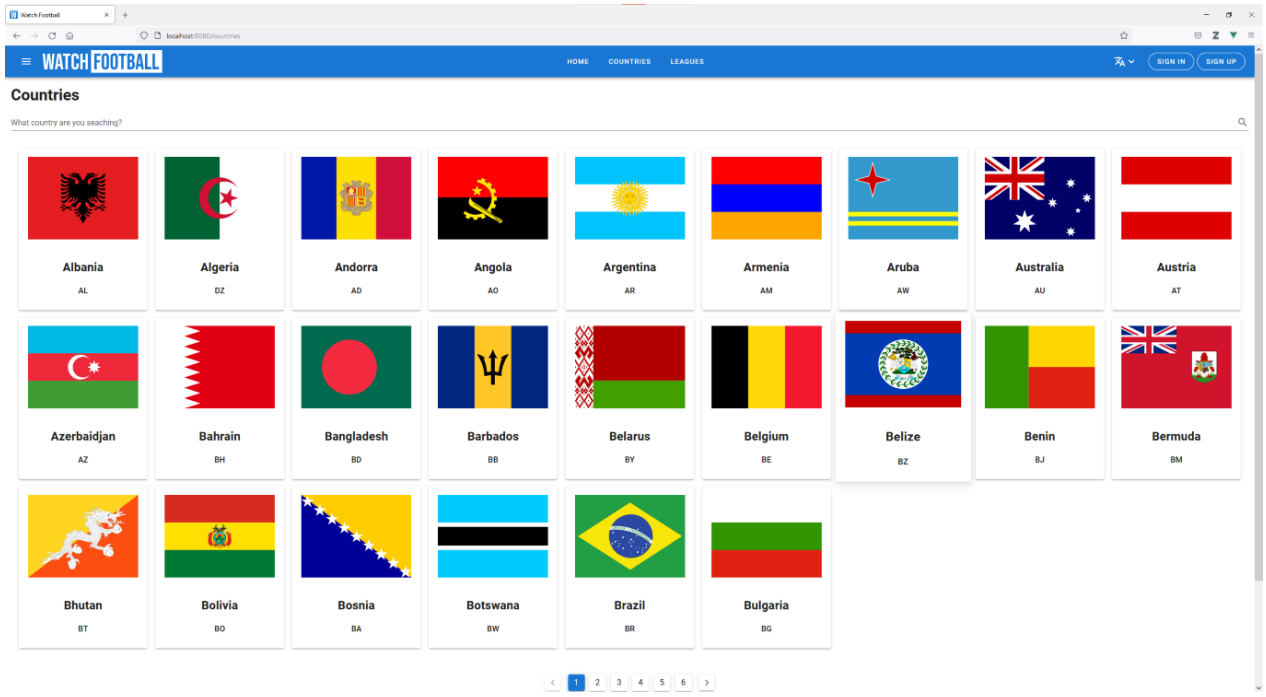


Figura A.2: Apariencia final de la página Countries.

A.3. Vista Country

En esta página se puede ver información sobre un país en concreto. Esta se encuentra dividida en dos partes: la cabecera, donde se puede ver una imagen de la bandera del país seleccionado junto a su nombre y código de país y el cuerpo, donde se puede consultar información más detallada.

En el cuerpo de esta vista se encuentra un sistema de pestañas que permite escoger entre visualizar las ligas disponibles para el país seleccionado (Fig. A.3) o los equipos de dicho país (Fig. A.4). Ambas pestañas cuentan con una barra de búsqueda a partir de la cual se puede filtrar la información buscada.

A.4. Vista Leagues

En esta página se pueden visualizar las distintas ligas disponibles dentro de la web. El resultado final de la misma se puede observar en la figura A.5.

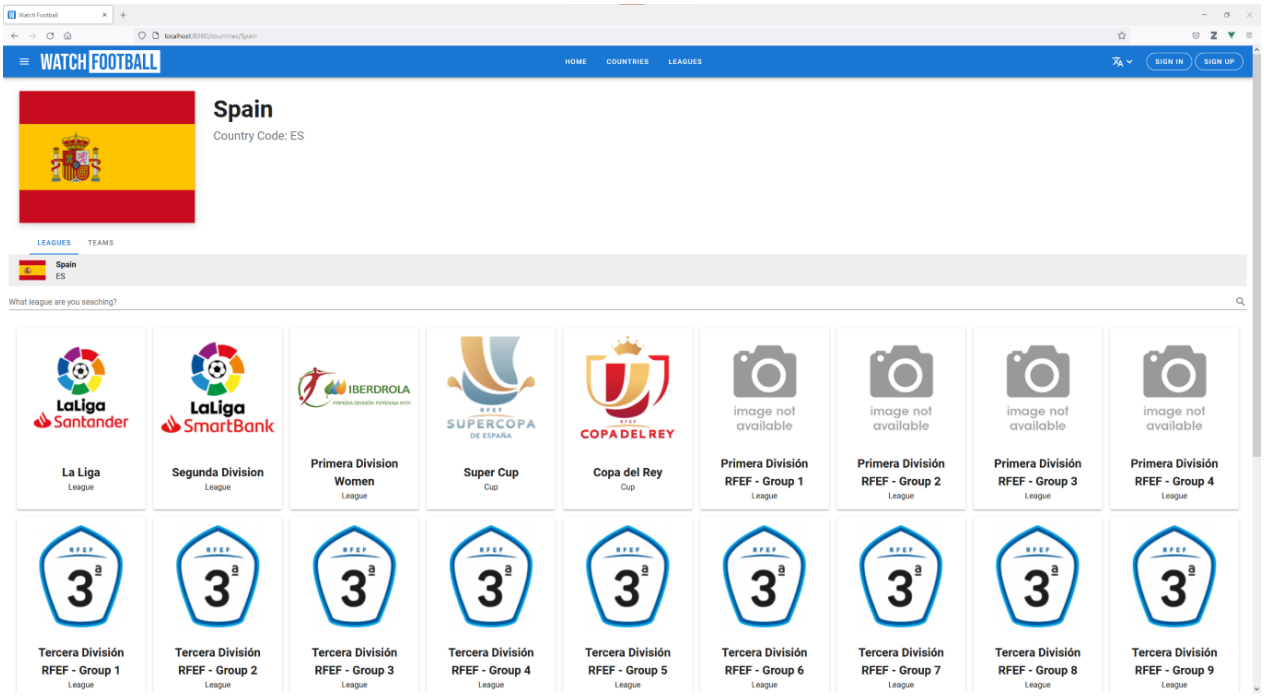


Figura A.3: Apariencia final de la pestaña Leagues en la vista Country.

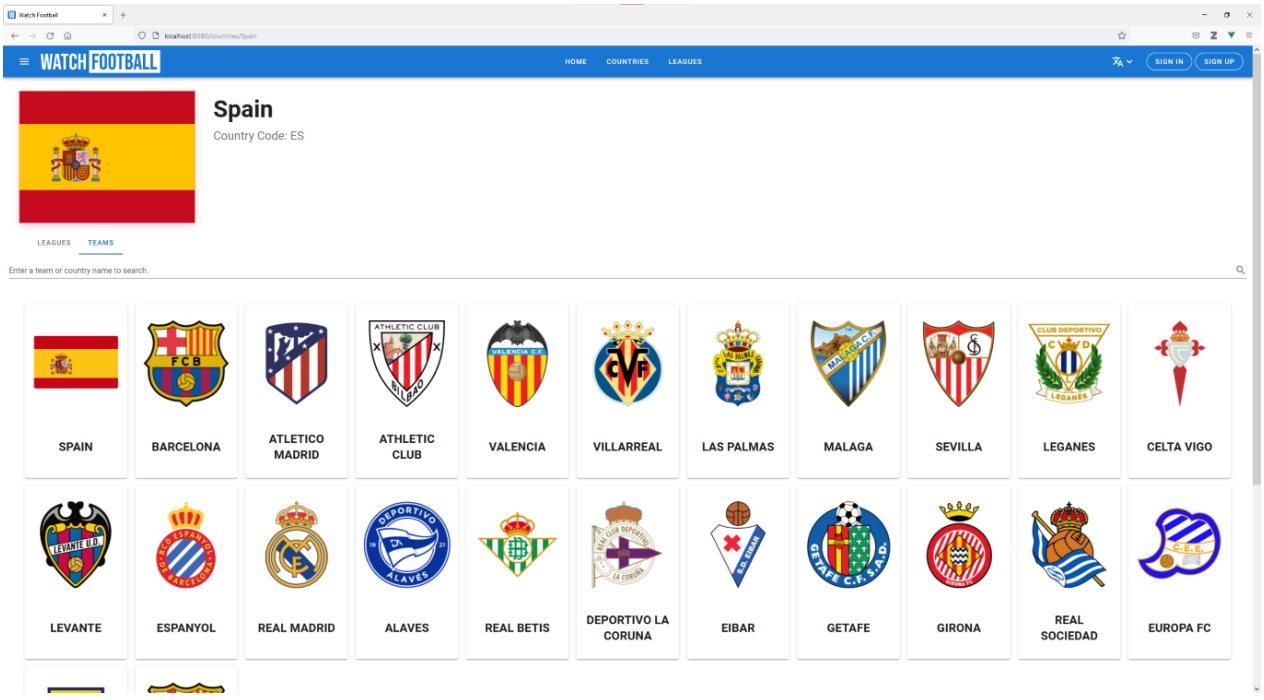


Figura A.4: Apariencia final de la pestaña Teams en la vista Country.

A.5. Vista League

En esta página es posible visualizar información relativa a una liga específica. La página, al igual que otras páginas, se encuentra dividida en: cabecera donde se puede encontrar el nombre, país, logo o selector de temporada de la liga (Fig. A.7) y el cuerpo donde se muestra información mas específica relativa a la liga. El resultado final de esta página se puede ver en la figura A.6.

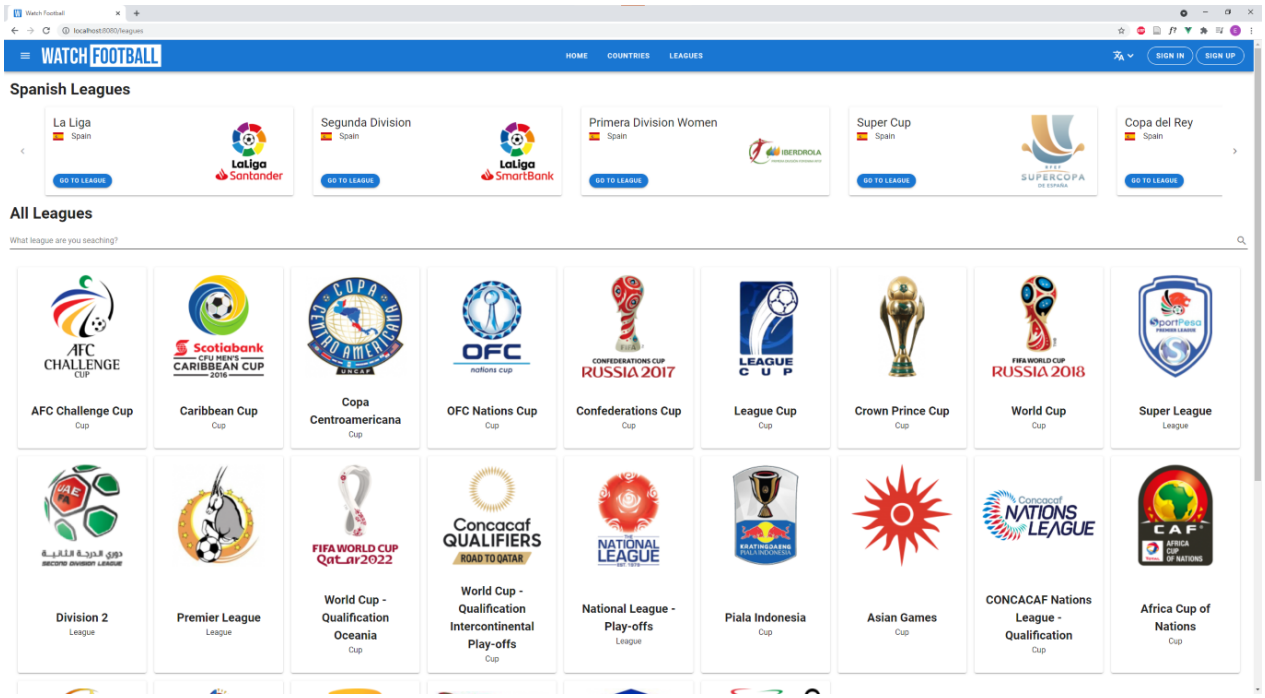


Figura A.5: Apariencia final de la página Leagues.

Dentro del cuerpo de la página nos encontramos con la siguiente información disponible:

- **Pestaña Fixtures**: Esta pestaña muestra los distintos partidos disponibles de una liga (Fig. A.6). Esta pestaña cuenta con una herramienta de filtrado (Fig. A.8) y otra de ordenación (A.9).
- **Pestaña Standings**: Esta pestaña muestra la clasificación actual de la liga para la temporada seleccionada (Fig. A.10)
- **Pestaña Teams**: Esta pestaña muestra los equipos disponibles para la liga y temporada seleccionados (Fig. A.11)
- **Pestaña Top Players**: Esta pestaña muestra el top 20 jugadores más destacados en las categorías: mayores goleadores, mayores asistentes, jugadores con más tarjetas amarillas y jugadores con más tarjetas rojas (Fig. A.12).

A.6. Vista Fixture

En esta página es posible visualizar información relativa a un partido de fútbol concreto (Fig. A.13). La página se encuentra dividida en cabecera y cuerpo, de estos podemos destacar lo siguiente.

- **Cabecera**: Muestra información general del partido.
  - **Información de la liga**: En la parte superior izquierda de la cabecera encontramos un cartel con información de la liga a la que pertenece el partido. En ella podemos ver el escudo de la liga, el país al que pertenece esta y la jornada en la que se jugó el partido.
  - **Fecha del partido**: En la parte superior izquierda de la cabecera encontramos la fecha y hora en la que se jugó o jugará el partido.

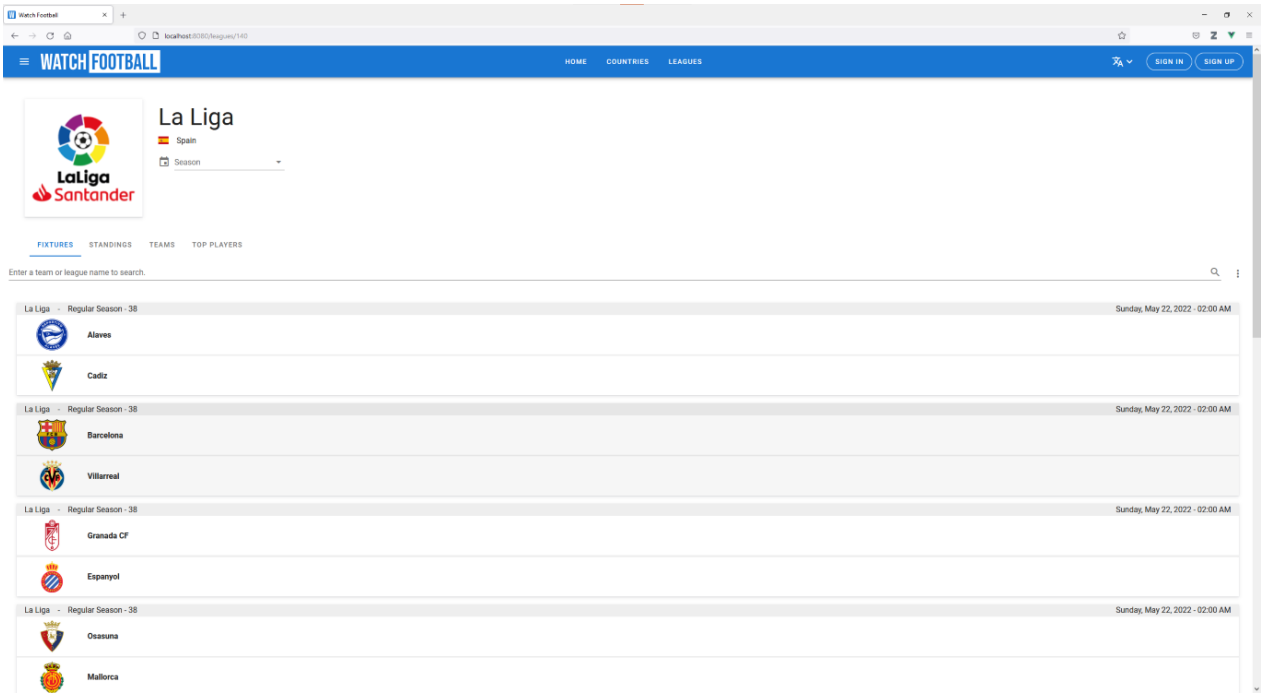


Figura A.6: Apariencia final de la página League.

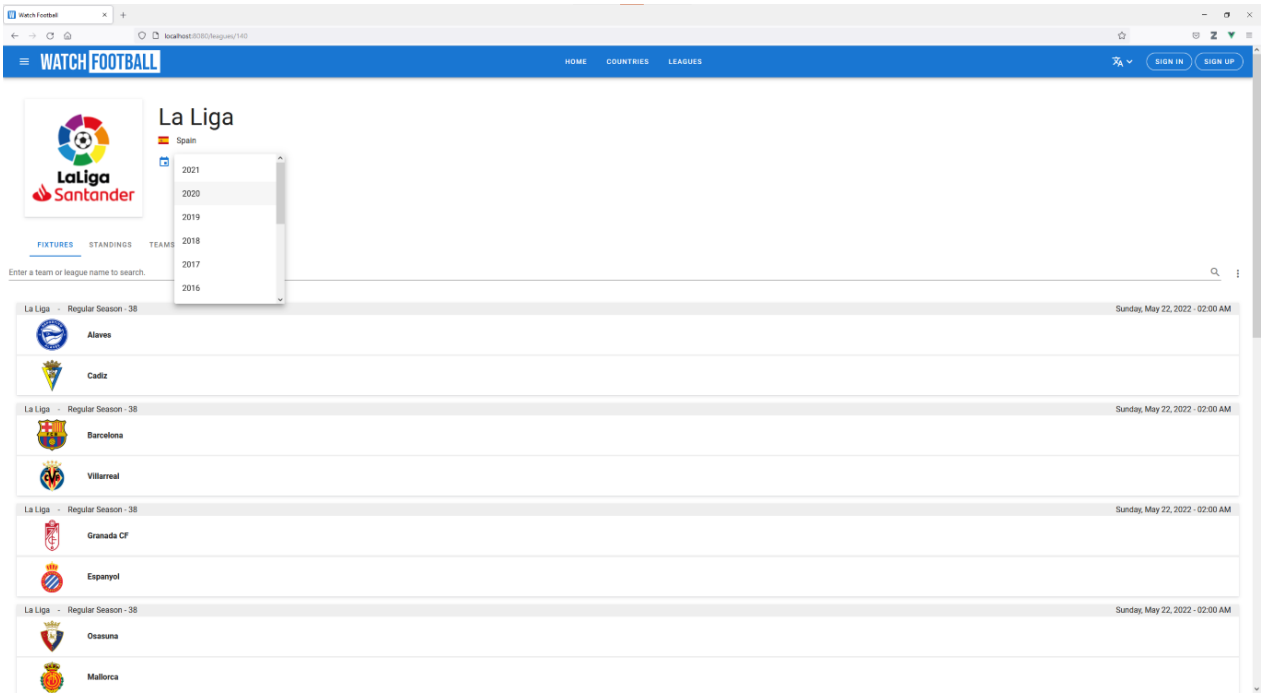


Figura A.7: Visualización de las herramientas de cambio de temporada.

- **Botón de guardar el partido**: Al situar el cursor sobre el cartel de la fecha la barra que se muestra a la derecha de esta se moverá mostrando un botón para guardar el partido. Esto solo ocurrirá si el usuario está logueado.
- **Información del partido**:En el centro de la cabecera se encuentra la información más importante del partido que son los equipos que lo jugaron, junto con el estado del partido y el resultado de este.

- **Cuerpo**: Muestra información más específica del partido.

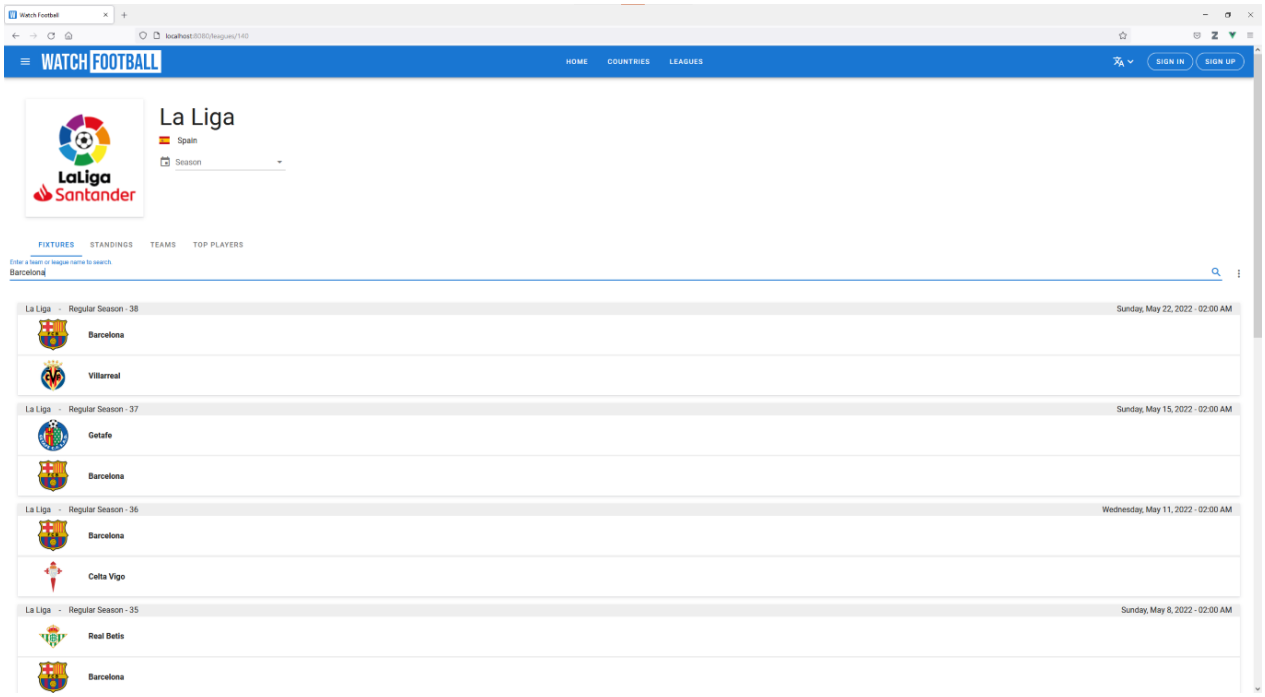


Figura A.8: Visualización de las herramientas de filtrado.

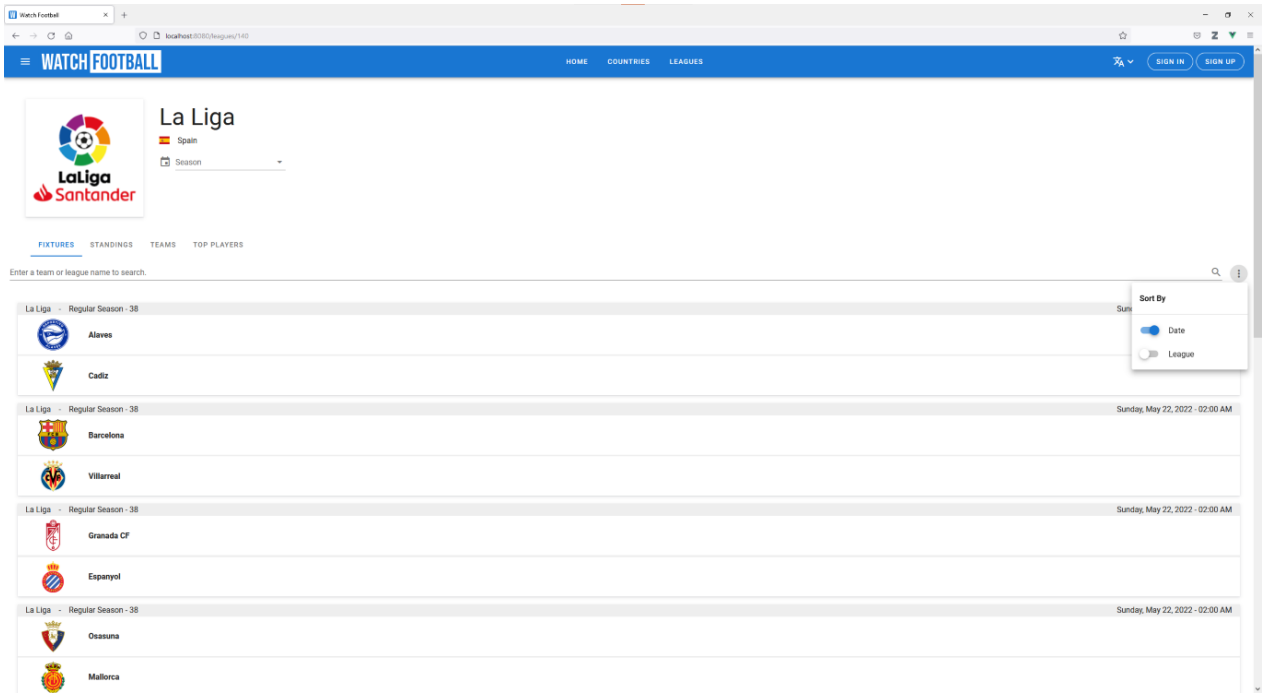


Figura A.9: Visualización de las herramientas de organizar.

- **Pestaña eventos**: En esta pestaña se muestran los distintos eventos surgidos durante el transcurso del partido. En ellos se muestra el tipo de evento junto con el jugador o jugadores afectados. Los eventos pueden ser del tipo tarjeta (amarilla, roja o segunda amarilla), gol (gol normal, gol en propia, penalti o penalti fallado), cambio (El número de cambio junto al jugador que entra y el que sale) o VAR (con eventos surgidos del VAR como gol cancelado o penalti confirmado). El resultado final de esta pestaña se puede ver en la figura A.14.
- **Head to Head**: Con todos los distintos partidos que han jugado estos dos equipos entre ellos. Esta

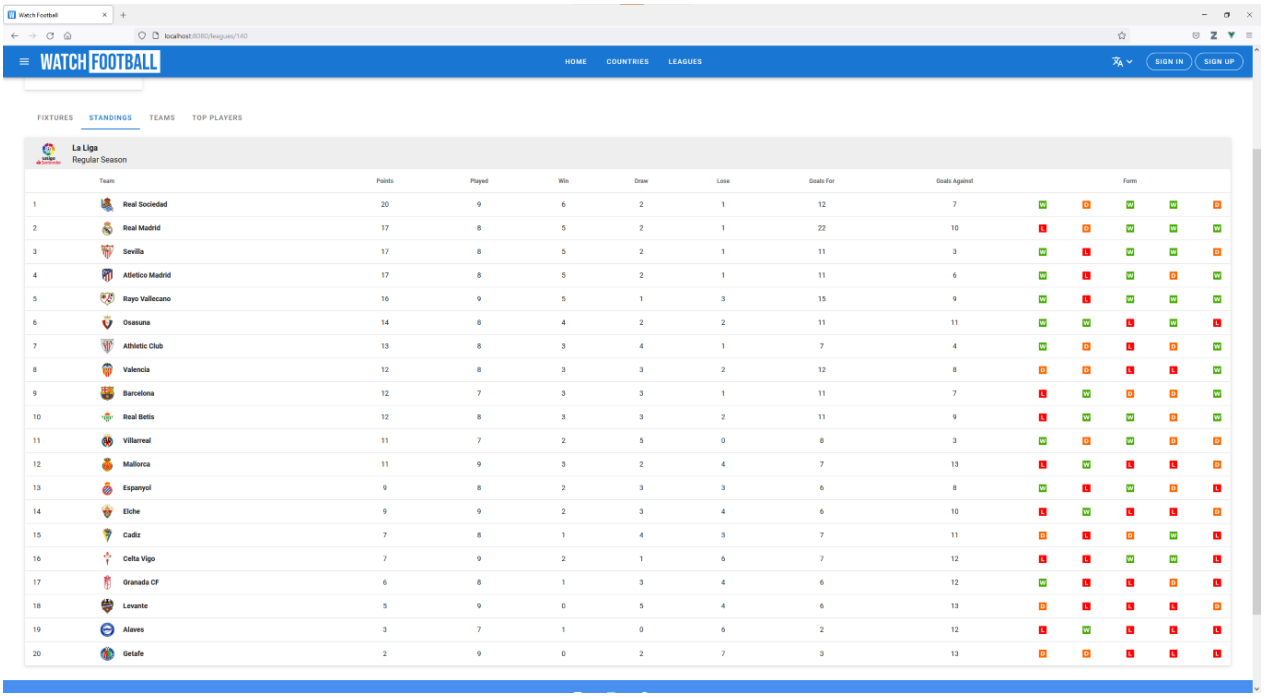


Figura A.10: Apariencia final de la pestaña Standings en la vista League.

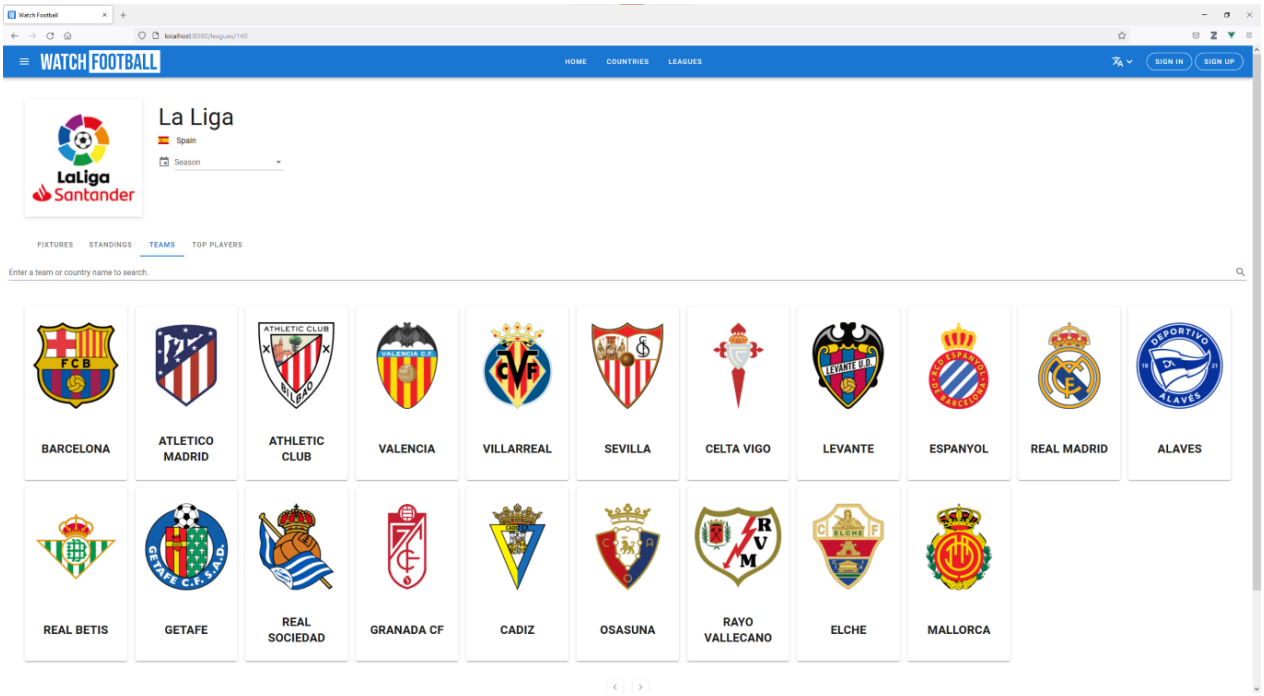


Figura A.11: Apariencia final de la pestaña Teams en la vista League.

pestaña cuenta también con herramienta de filtro y orden de los partidos. El resultado final de la pestaña se puede observar en la página A.15.

- **Lineups**: En esta pestaña se pueden observar las alineaciones planteadas por ambos equipos para el encuentro. Para las alineaciones se muestran dos tablas con los jugadores que participan en el encuentro, y un campo de fútbol en el que se distribuyen de manera correcta los jugadores dependiendo de la alineación propuesta por cada equipo. El resultado final de esta pestaña se puede observar en la imagen A.16.

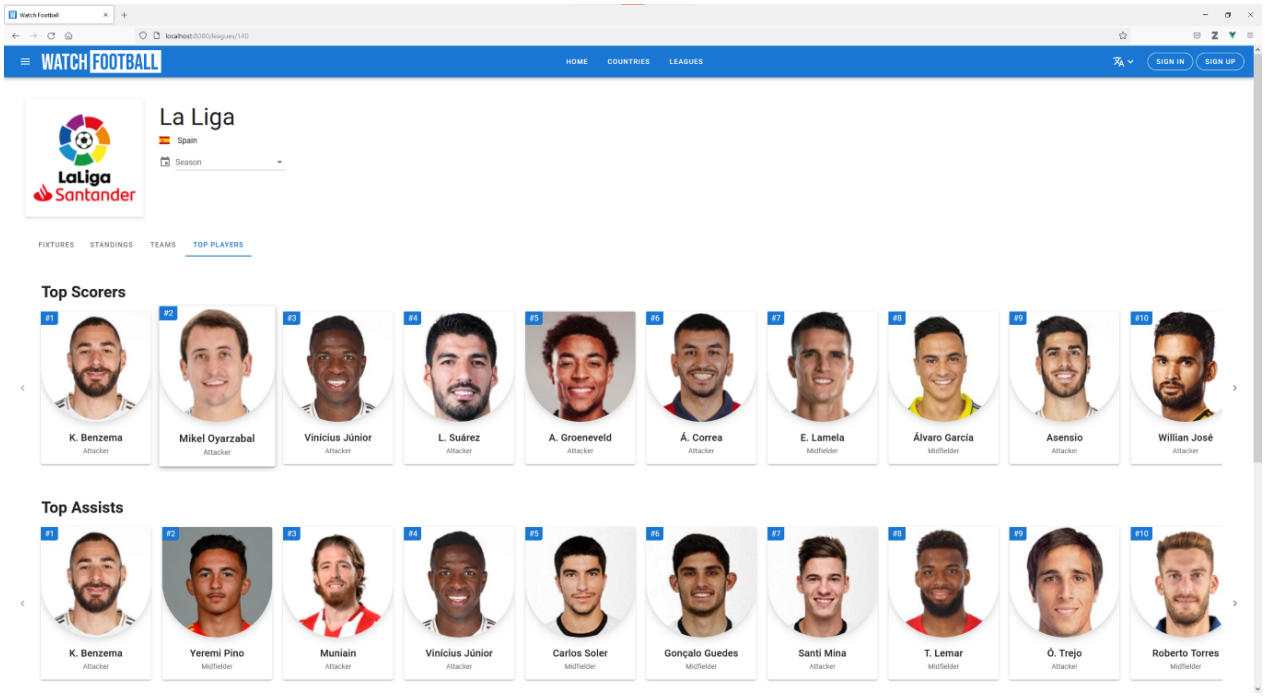


Figura A.12: Apariencia final de la pestaña Top Players en la vista League.

- **Match Statistics**: En esta pestaña se pueden observar las estadísticas del partido. Las estadísticas se muestran a partir de barras horizontales que se rellenan en función de las estadísticas obtenidas por cada equipo. El resultado final de esta pestaña se puede observar en la imagen A.17.

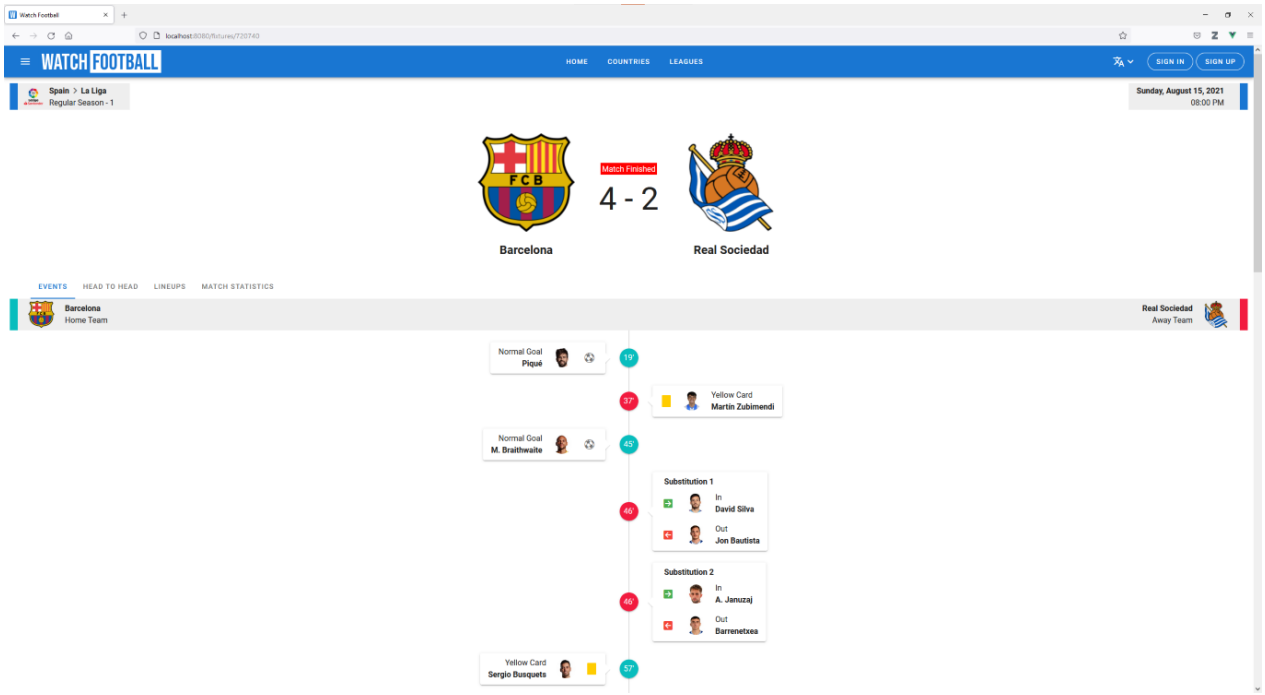


Figura A.13: Apariencia final de la página Fixture.

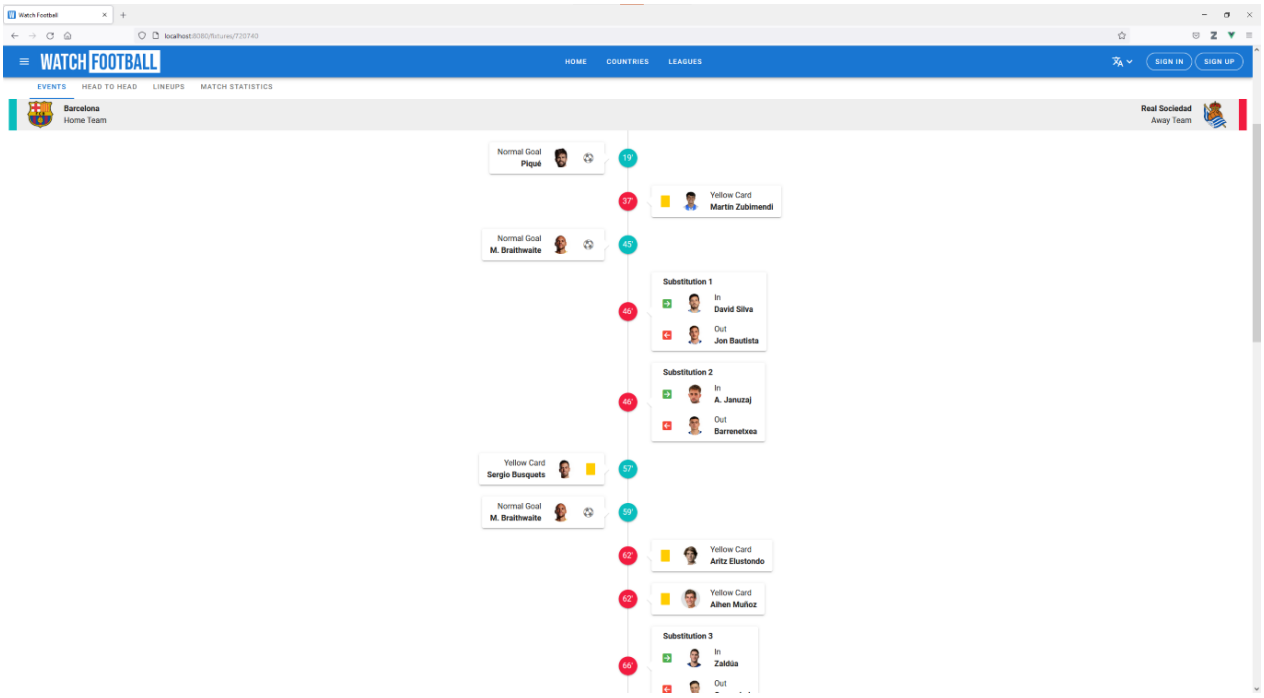


Figura A.14: Apariencia final de la pestaña Events en la vista Fixture.

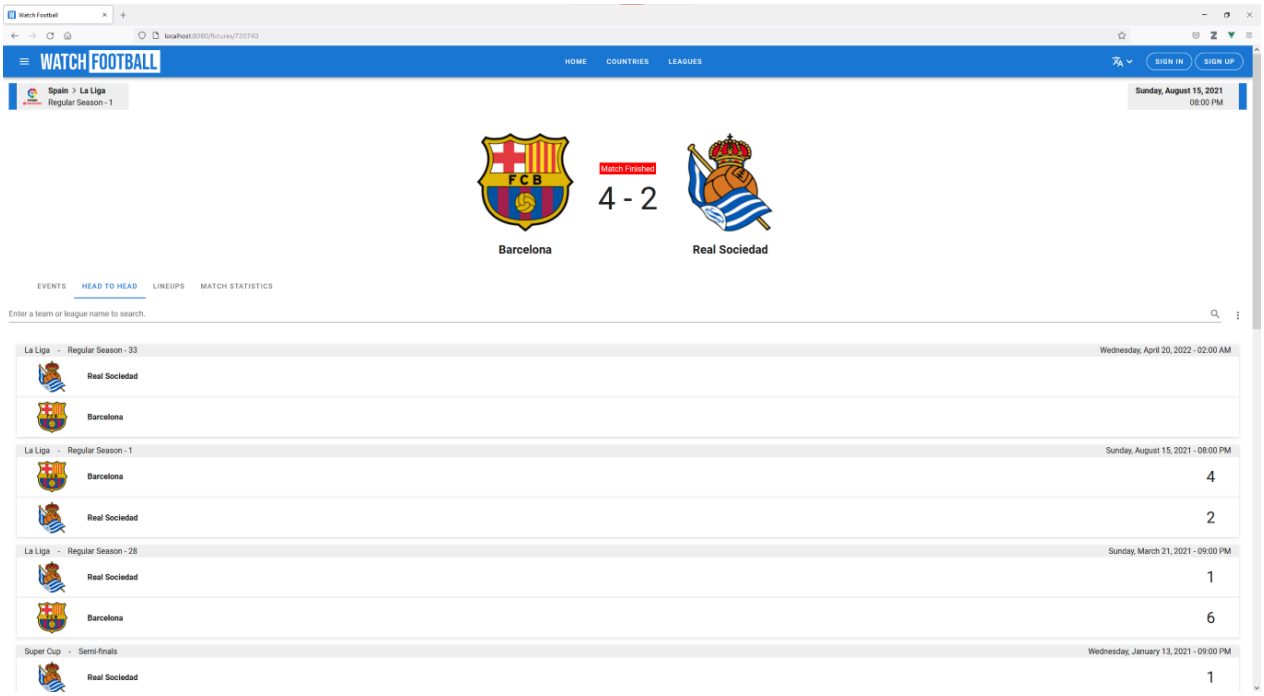


Figura A.15: Apariencia final de la pestaña Head To Head en la vista Fixture.

A.7. Vista Team

En esta página es posible visualizar información relativa a un determinado equipo de fútbol (Fig. A.13). La página se encuentra dividida en cabecera y cuerpo, de estos podemos destacar lo siguiente.

- **Cabecera**: Muestra información general del equipo.

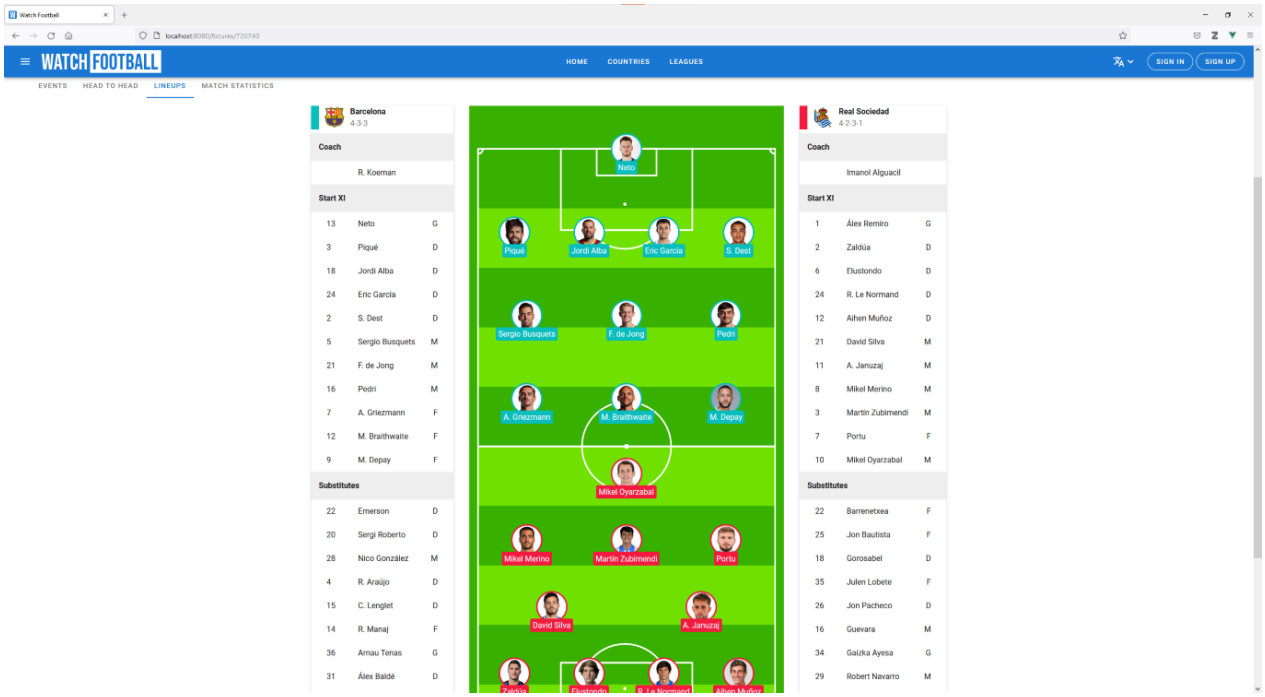


Figura A.16: Apariencia final de la pestaña Lineups en la vista Fixture.

Figura A.17: Apariencia final de la pestaña Statistics en la vista Fixture.

- **Escudo del Equipo**: En la parte izquierda de la cabecera se muestra el escudo del equipo que estamos consultando.
- **Descripción del Equipo**: A la derecha del escudo nos encontramos con una descripción del equipo que en la parte superior muestra el nombre del mismo seguido por la fecha de fundación, el país al que pertenece y una herramienta de selección de temporada.

- **Cuerpo**: Muestra información más específica del equipo.

- **Statistics**: En esta pestaña se muestran las estadísticas del equipo. Para mostrarlas es necesario seleccionar la temporada y la liga sobre las que queremos consultar las estadísticas.Una vez rellenados estos campos se mostrarán una serie de tablas y estadísticas sobre la liga seleccionada. El resultado final de la pestaña puede consultarse en la imagen A.20.
- **Leagues**: Muestra una cuadrícula con todas las ligas en las que participa el equipo consultado. El resultado final de la pestaña se puede observar en la página A.21.
- **Fixtures**: Muestra una lista de todos los partidos que ha jugado el equipo durante la temporada seleccionada. Esta herramienta cuenta con filtro y selector de orden al igual que las demás listas de partidos. El resultado final de esta pestaña se puede observar en la imagen A.22.
- **Players**: En esta pestaña se pueden consultar los distintos jugadores que se encuentran en la plantilla del equipo durante la temporada seleccionada. El resultado final de esta pestaña se puede observar en la imagen A.23.
- **Transfers**: En esta pestaña se pueden consultar todos los traspasos que ha realizado el equipo. El resultado final de esta pestaña se puede observar en la imagen A.24.
- **Coachs**: En esta pestaña se pueden observar los distintos entrenadores que han formado parte de la plantilla del equipo seleccionado. El resultado final de esta pestaña se puede observar en la imagen A.25.

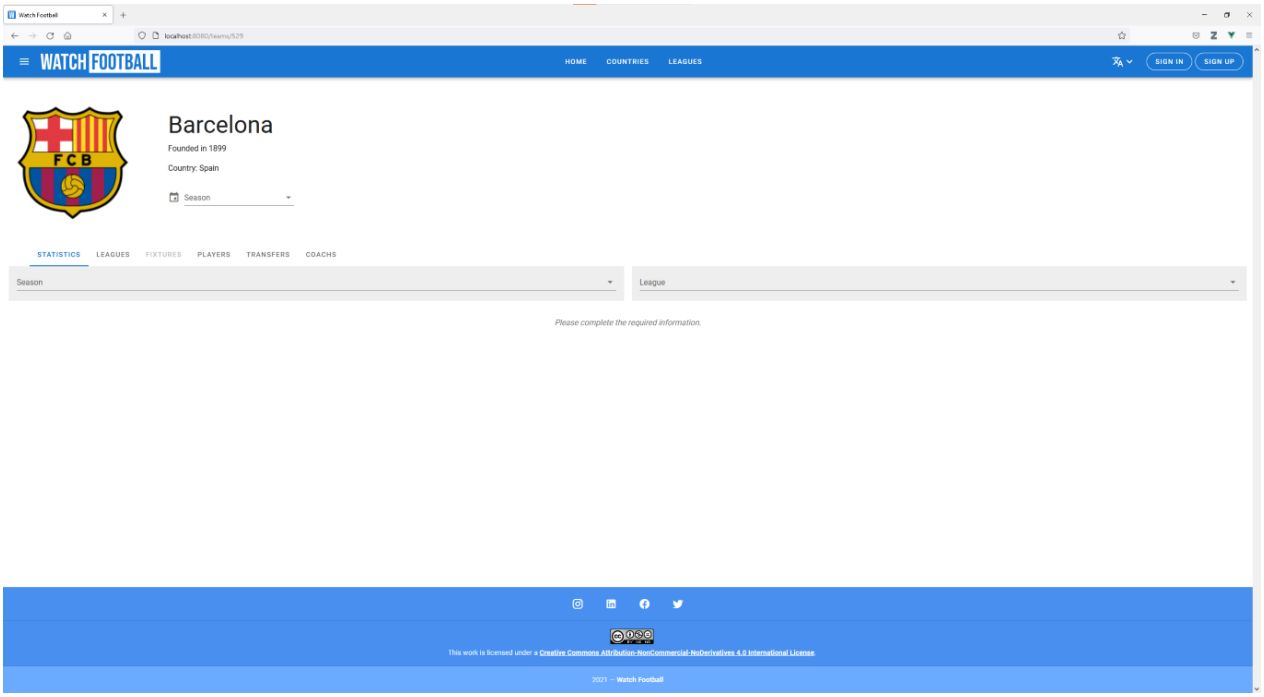


Figura A.18: Apariencia final de la página Team.

A.8. Vista Player

En esta página se puede visualizar información sobre jugadores concretos disponibles en la web. En la cabecera de la página se puede consultar información general sobre el jugador seleccionado como su nombre, edad, fecha de nacimiento o nacionalidad. En el cuerpo de la página se pueden consultar sus estadísticas en los distintos equipos en los que juega (Fig. A.26), los traspasos en los que ha participado el jugador (Fig. A.27), los trofeos que ha ganado (Fig. A.28) o las veces que no ha estado disponible para jugar (Fig. A.29)

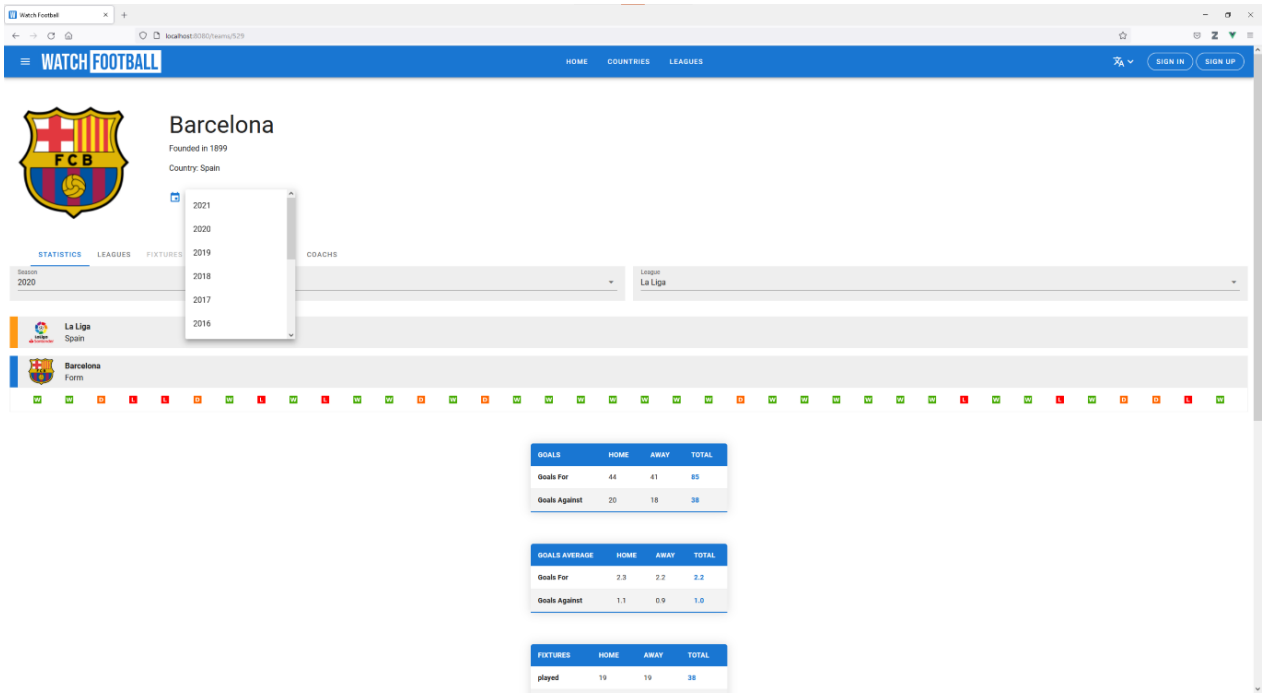


Figura A.19: Visualización de las herramientas de selección de temporadas.

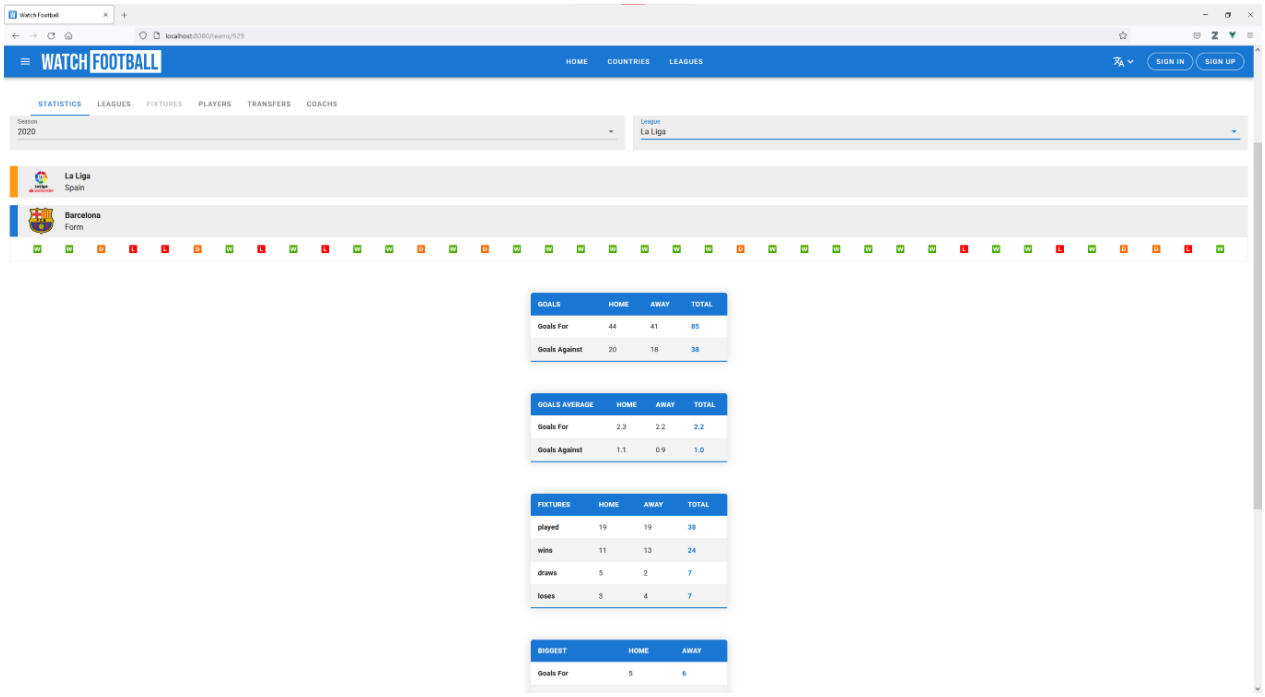


Figura A.20: Apariencia final de la pestaña Statistics en la vista Team.

A.9. Vista Coach

En esta página es posible consultar información sobre un entrenador concreto. Para ello en la cabecera se muestra información de carácter general sobre el entrenador, como podría ser su nombre, fecha de nacimiento, edad o nacionalidad. En el cuerpo de esta vista se muestran información más específica como podrían ser los equipos en los que ha sido entrenador (Fig. A.30), los trofeos que ha obtenido (Fig. A.31) o las veces que no ha estado disponible por lesiones o expulsiones (Fig. A.32).

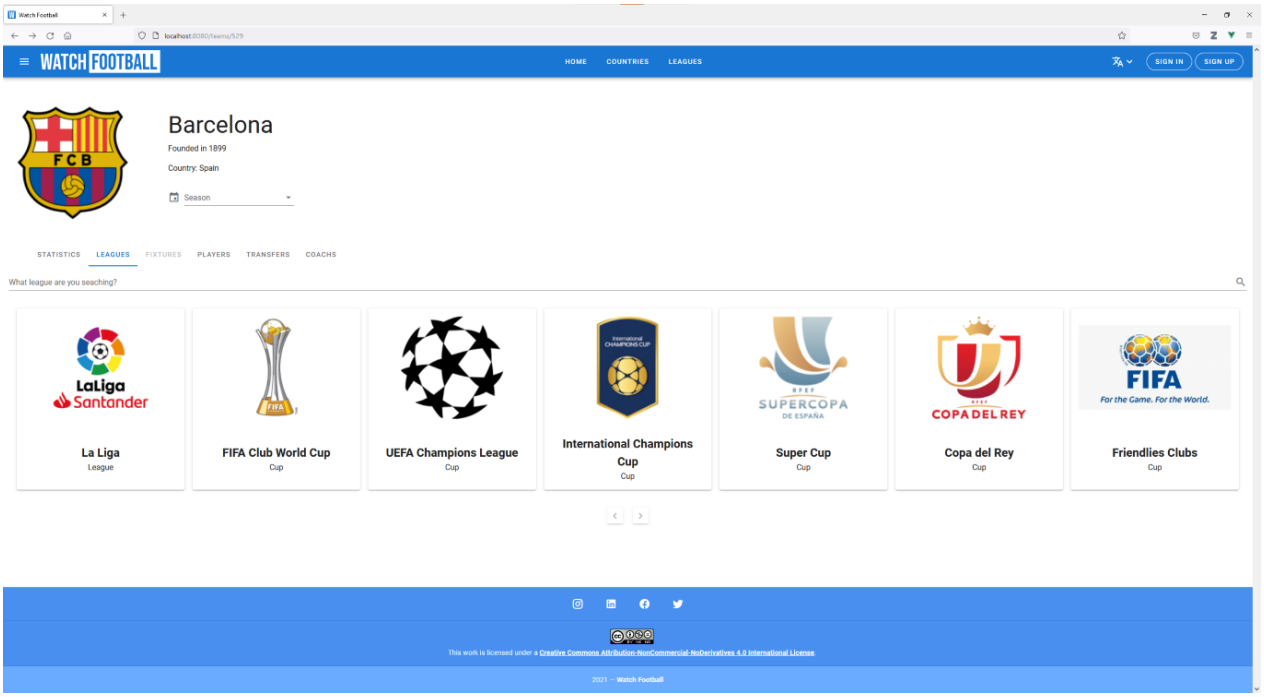


Figura A.21: Apariencia final de la pestaña Leagues en la vista Team.

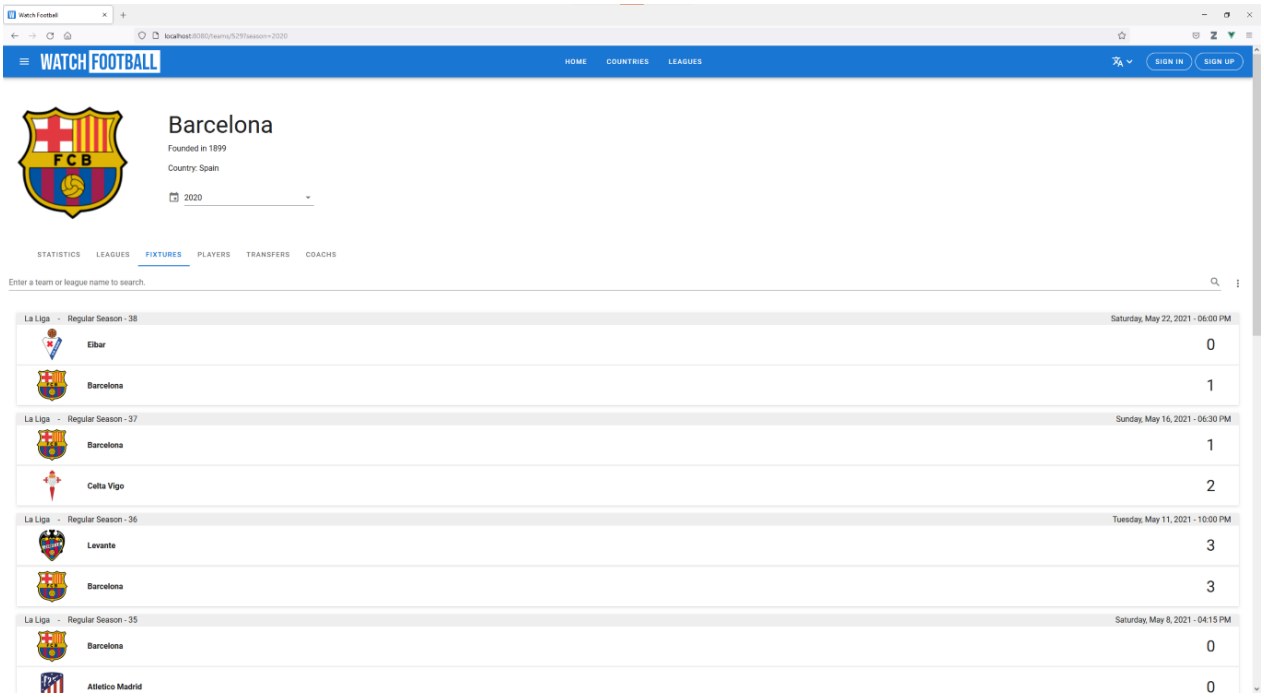


Figura A.22: Apariencia final de la pestaña Fixtures en la vista Team..

A.10. Cambio de idioma

Se ha implementado una herramienta a través de la cual se puede modificar el idioma de la aplicación en tiempo real. Para acceder a esta herramienta es necesario pulsar sobre el icono de traducción en la parte derecha de la barra superior de la aplicación, esto mostrará un menú con los idiomas disponibles (Fig. A.33), en este momento solo los idiomas inglés y español. Una vez hecho esto, los textos de la aplicación se habrán traducido al idioma seleccionado y este se deshabilitará en el menú de selección. La apariencia de la aplicación tras traducir los textos

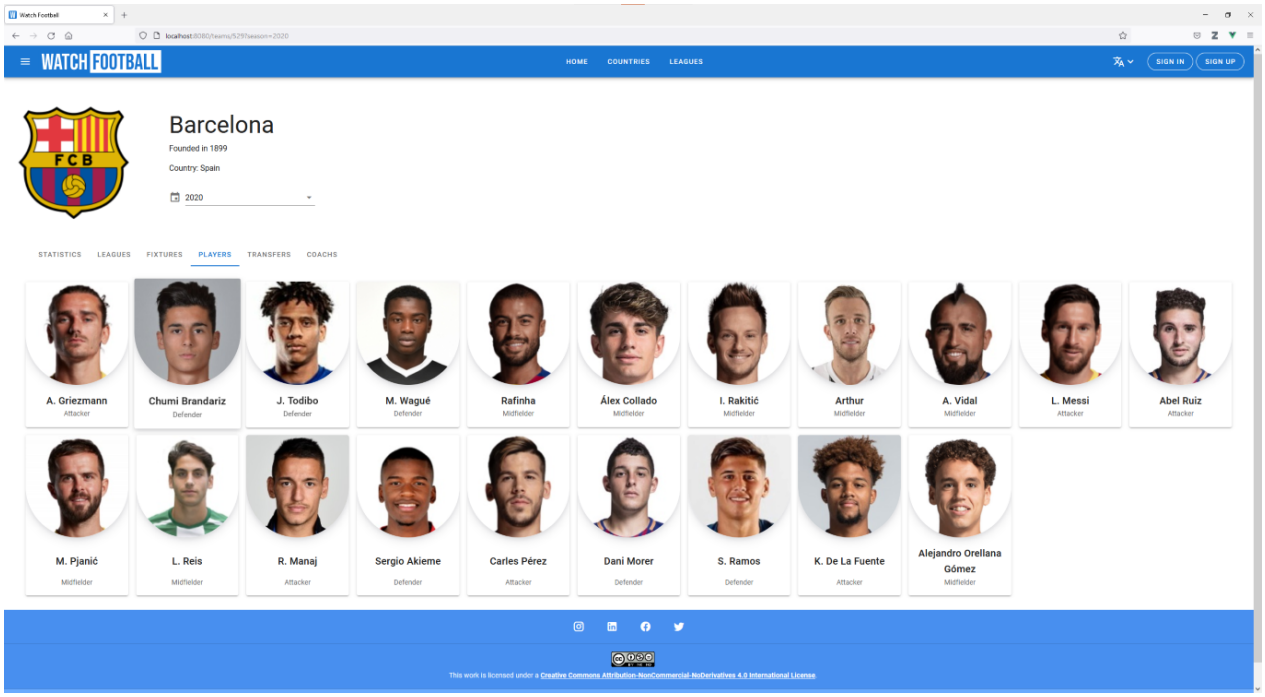


Figura A.23: Apariencia final de la pestaña Players en la vista Team.

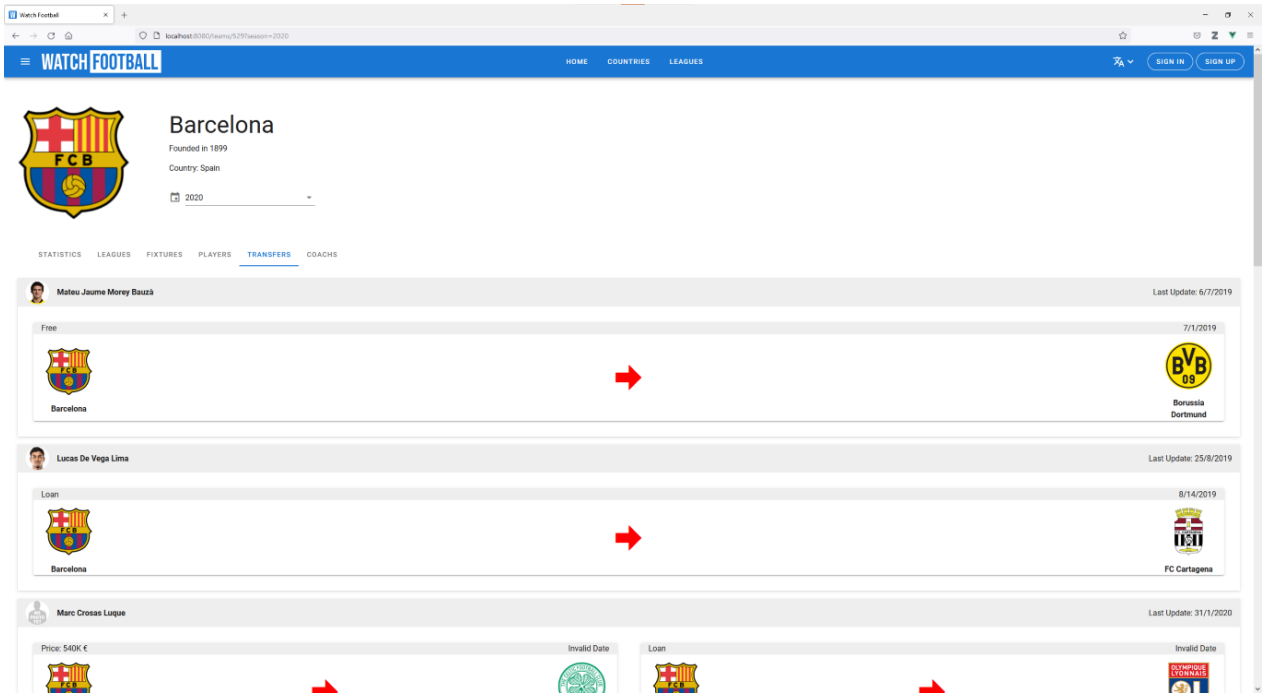


Figura A.24: Apariencia final de la pestaña Transfers en la vista Team.

puede verse en la figura A.34

A.11. SignIn

La página de SignIp es la página a través de la cual se puede acceder como usuario a la aplicación (Fig. A.35). La vista está formada por Por un formulario con dos campos de texto (email y contraseña) y tres botones (uno para

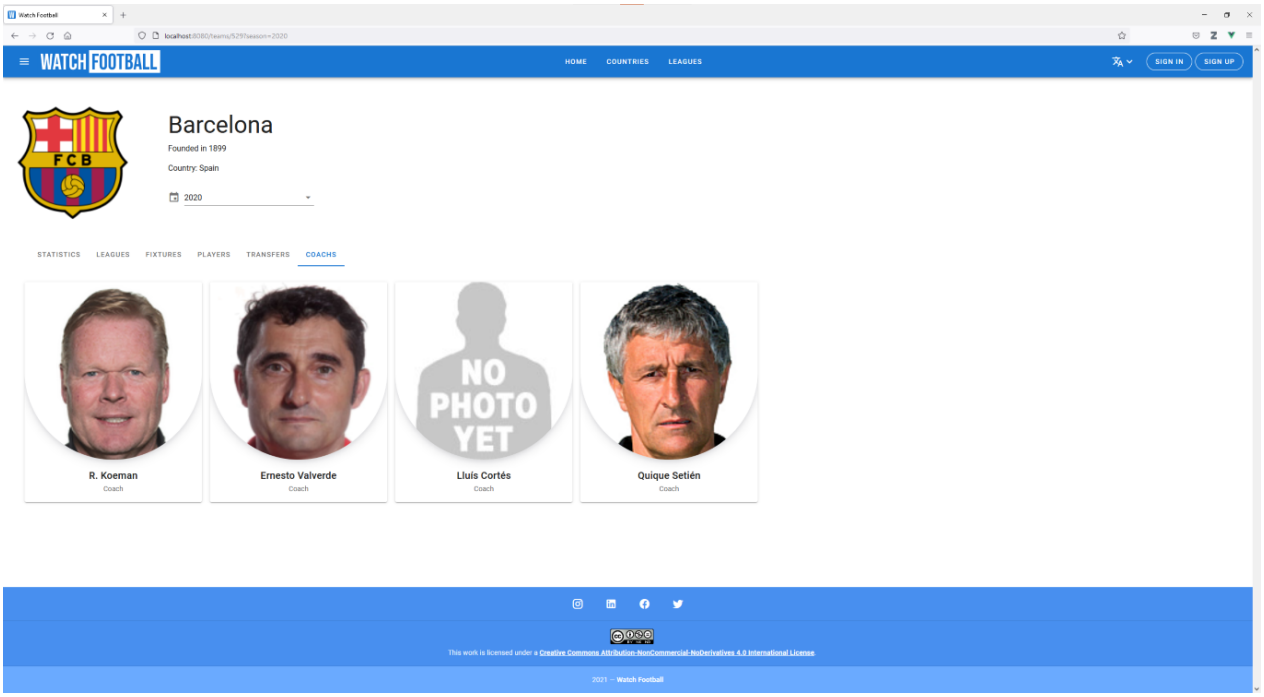


Figura A.25: Apariencia final de la pestaña Coachs en la vista Team.

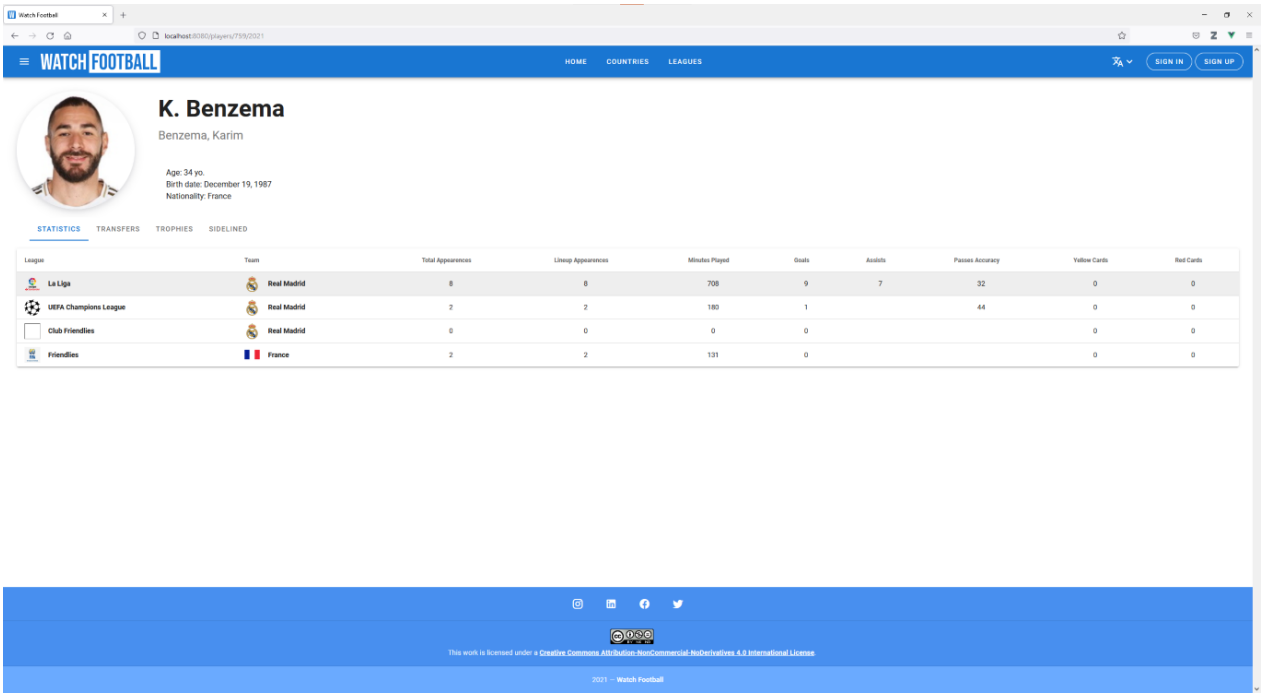


Figura A.26: Apariencia final de la pestaña Statistics en la vista Player.

enviar el formulario, otro para abrir el formulario de recuperación de contraseña y otro para redirigir al usuario a la página de registro).

Los campos de texto cuentan con reglas para verificar que los campos se han rellenado y que estos contienen información formateada de la manera correcta. Por ejemplo, el campo de email solo permite introducir cadenas de texto con el formato específico de un correo electrónico. De no cumplirse estas reglas se mostrará un error específico en el campo afectado (Fig. A.36).

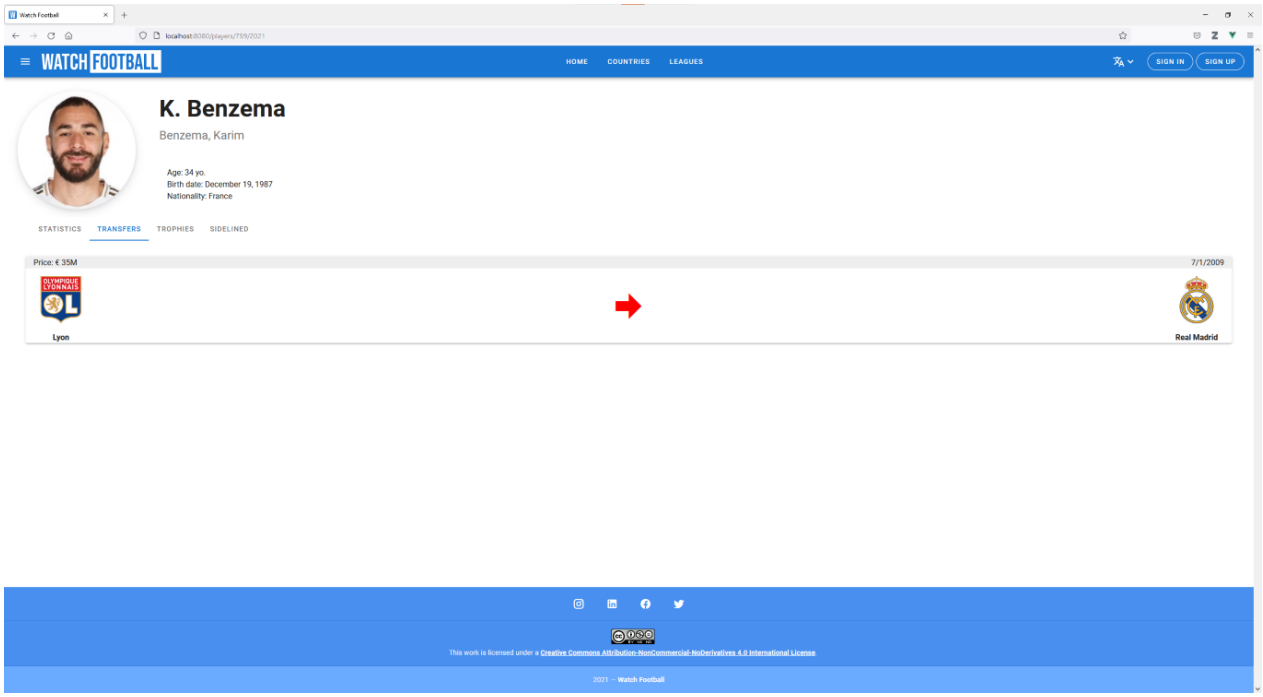


Figura A.27: Apariencia final de la pestaña Transfers en la vista Player.

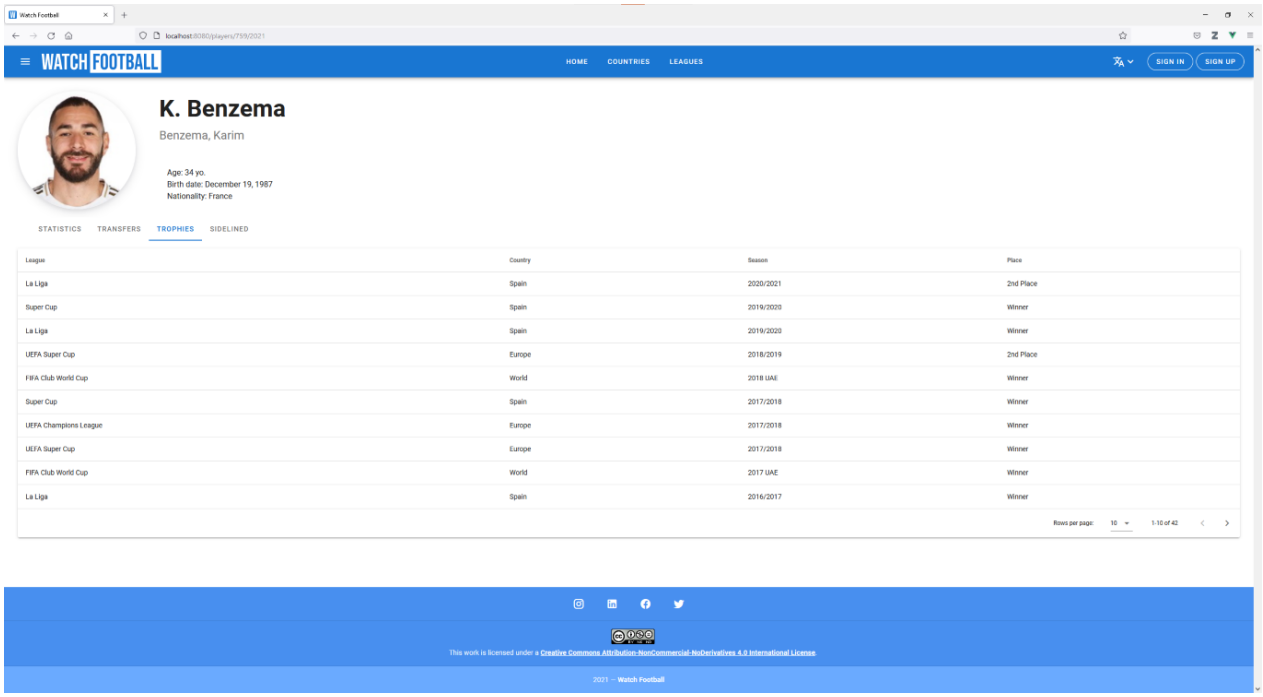


Figura A.28: Apariencia final de la pestaña Trophies en la vista Player.

El formulario de inicio de sesión también cuenta con una herramienta de recuperación (Fig. A.39) de contraseña que permite modificarla en caso de no poder acceder con tu cuenta a la aplicación.

Una vez rellenados los campos de manera correcta (Fig. A.37) es posible enviar el formulario para autentificar el usuario. Una vez enviado el formulario se muestra un mensaje en la parte superior de la página (Fig. A.38) informando sobre el resultado de la autenticación (un cartel rojo con un mensaje de error si se ha producido un error o un mensaje verde si todo ha funcionado correctamente).

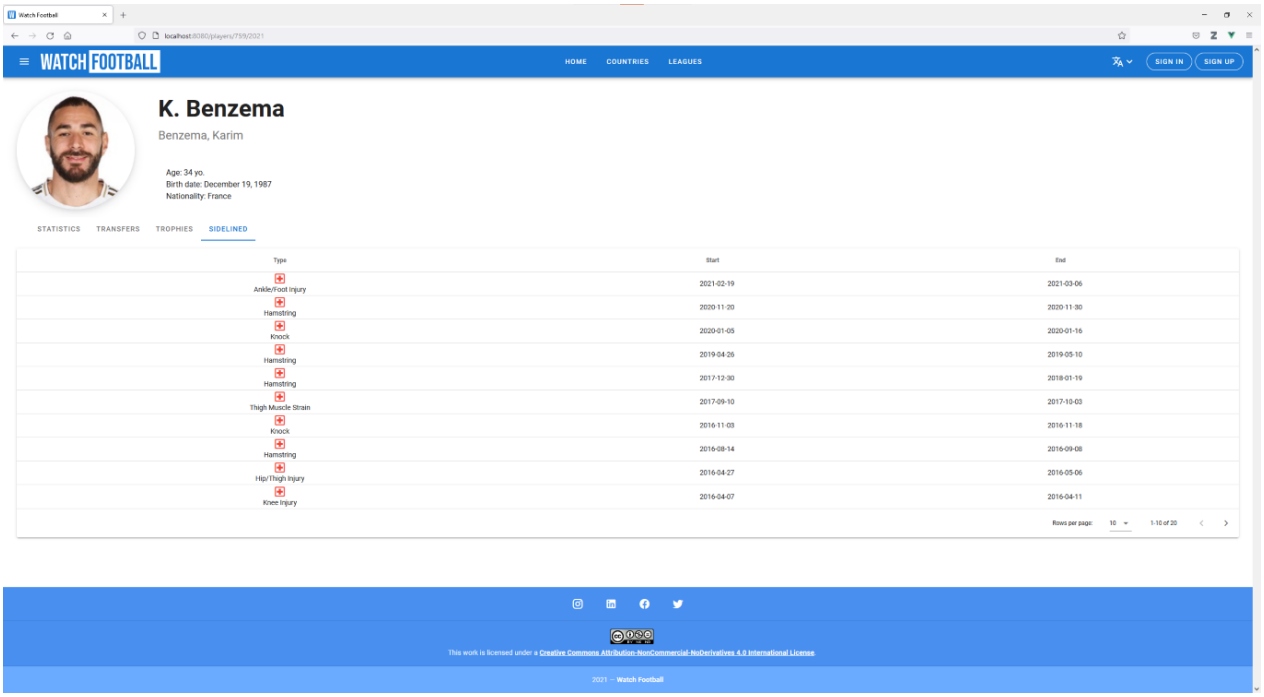


Figura A.29: Apariencia final de la pestaña Sidelined en la vista Player.

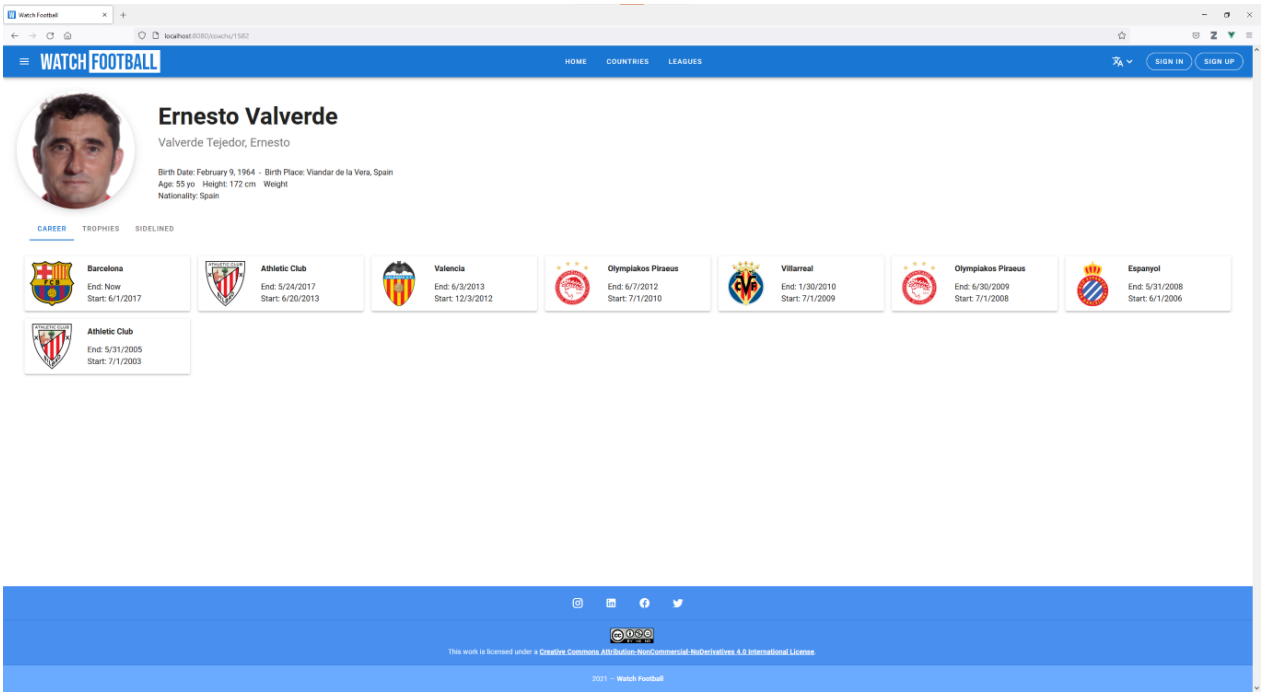


Figura A.30: Apariencia final de la pestaña Career en la vista Coach.

A.12. SignUp

La página de SignUp es la página que permite registrarse en la aplicación (Fig. A.40). Esta contiene un formulario con los siguientes campos:

- **Nombre**: Nombre completo del usuario.

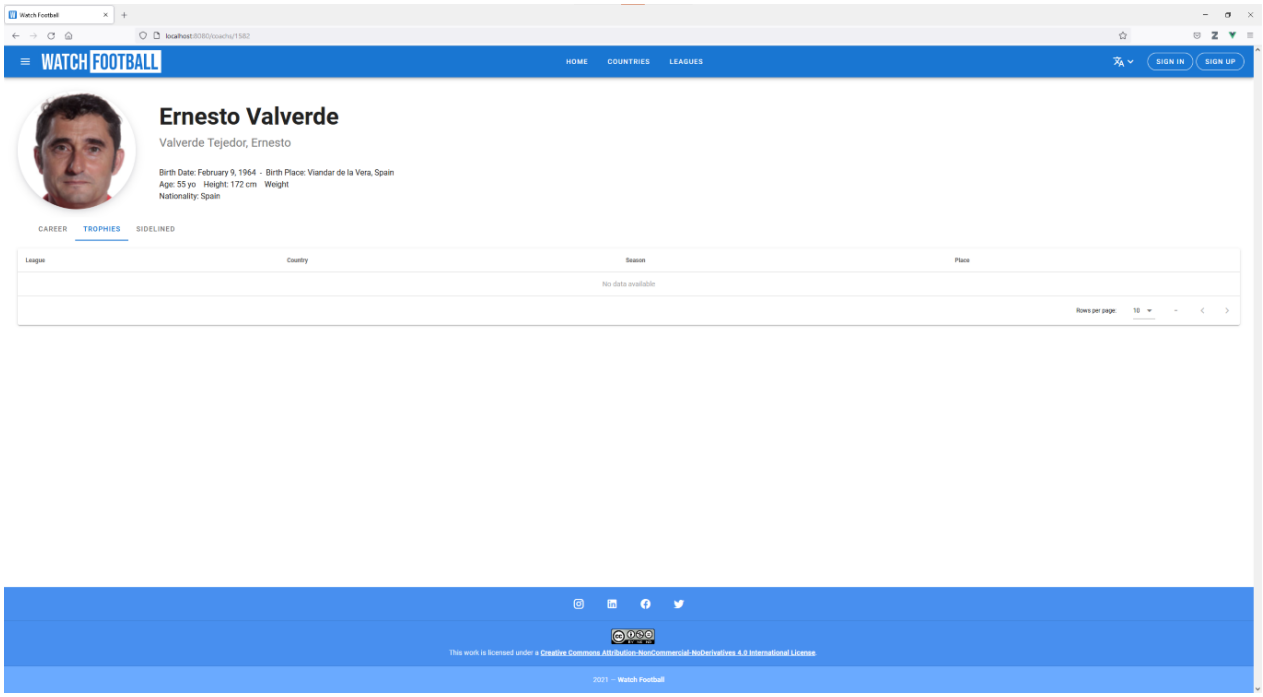


Figura A.31: Apariencia final de la pestaña Trophies en la vista Coach.

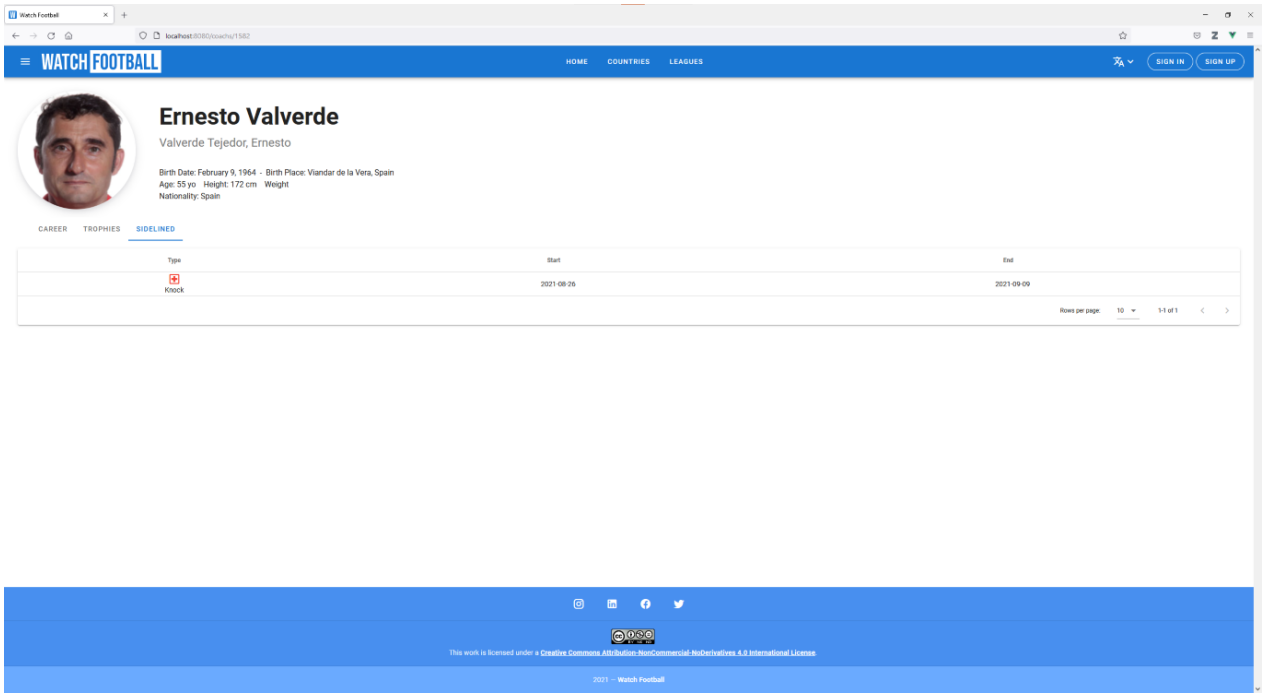


Figura A.32: Apariencia final de la pestaña Sidelined en la vista Coach.

- **Nombre de Usuario**: Nombre abreviado de usuario.
- **Email**: Email vinculado a la cuenta.
- **Contraseña**: Contraseña del usuario.
- **Repetir contraseña**: Campo para verificar que la contraseña introducida es correcta.
- **País**: País al que quieres vincular la cuenta. Este es el único campo del formulario que no es obligatorio.

Figura A.33: Visualización de las herramientas de cambio de idioma.

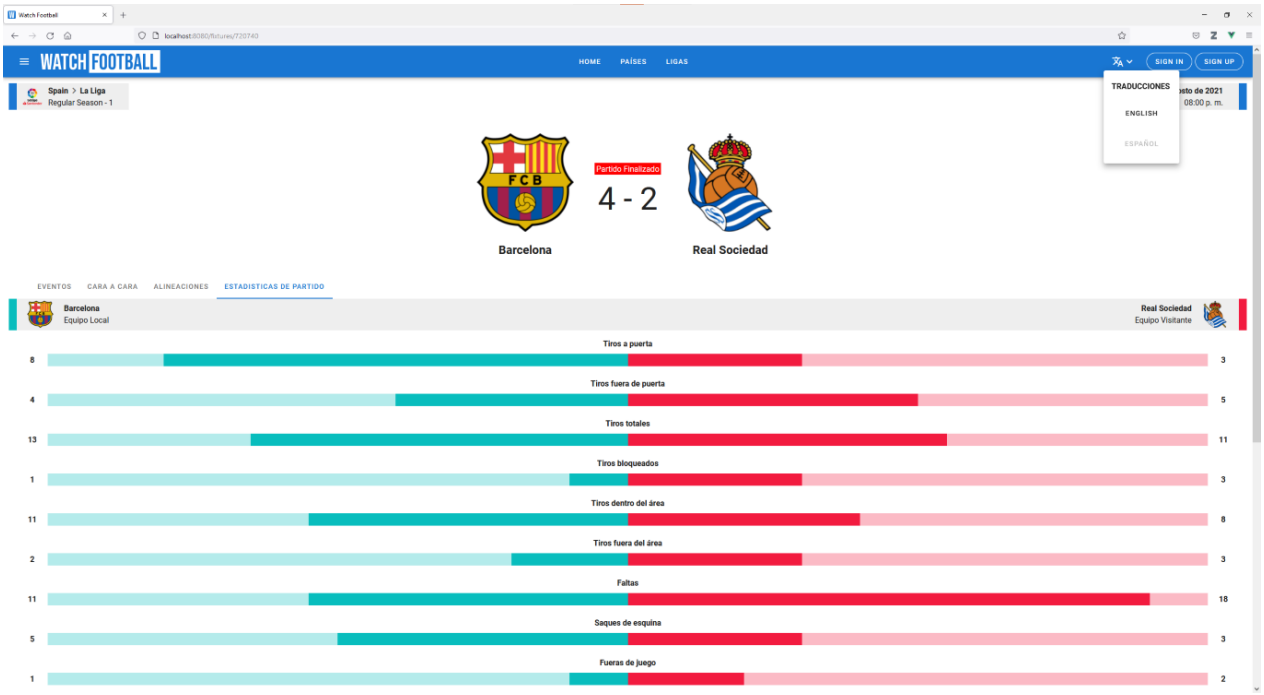


Figura A.34: Apariencia de la aplicación tras el cambio de idioma.

- **Términos y condiciones**: Los términos y condiciones de uso de la aplicación.

Todos los campos de texto del formulario cuentan con reglas personalizadas que deben verificarse antes de poder enviar el formulario, por ello en caso de que alguna de ellas no se cumpla se mostrará un error en el campo de texto afectado (Fig. A.41).

Para poder registrarse en la aplicación es necesario aceptar los términos y condiciones de uso de la misma. Estos pueden ser consultados a través del enlace que aparece en el formulario (Fig. A.42).

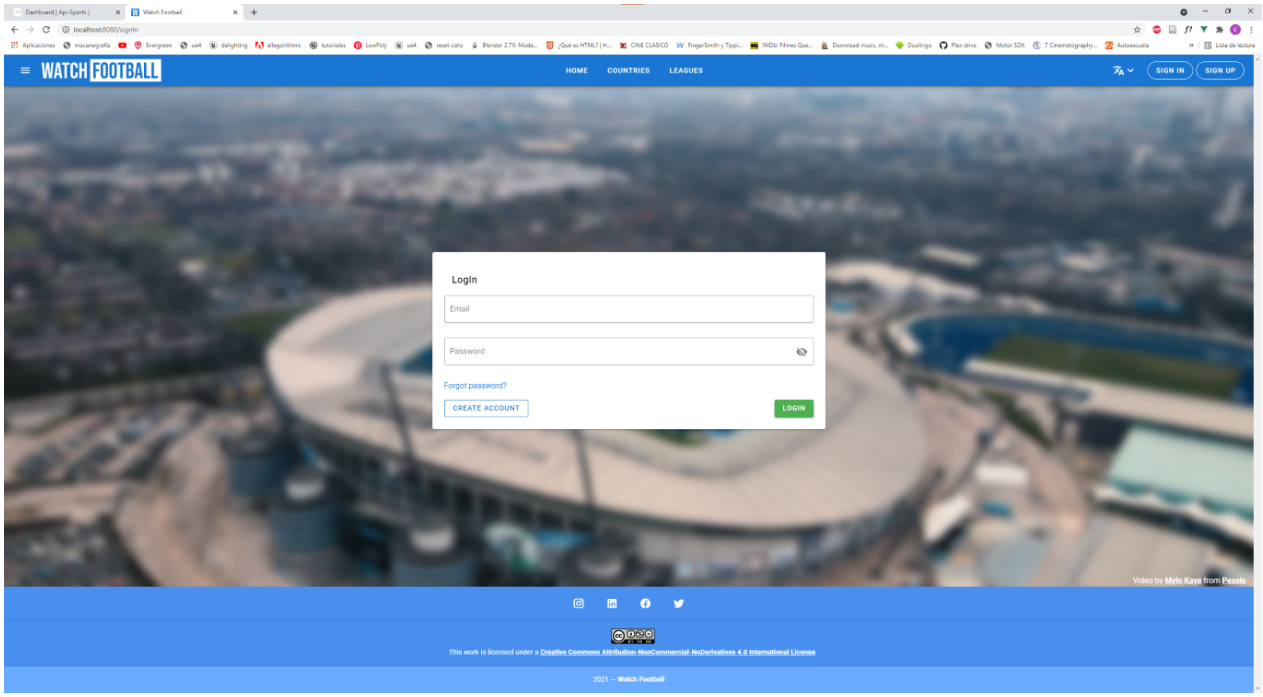


Figura A.35: Apariencia final de la página SignIn.

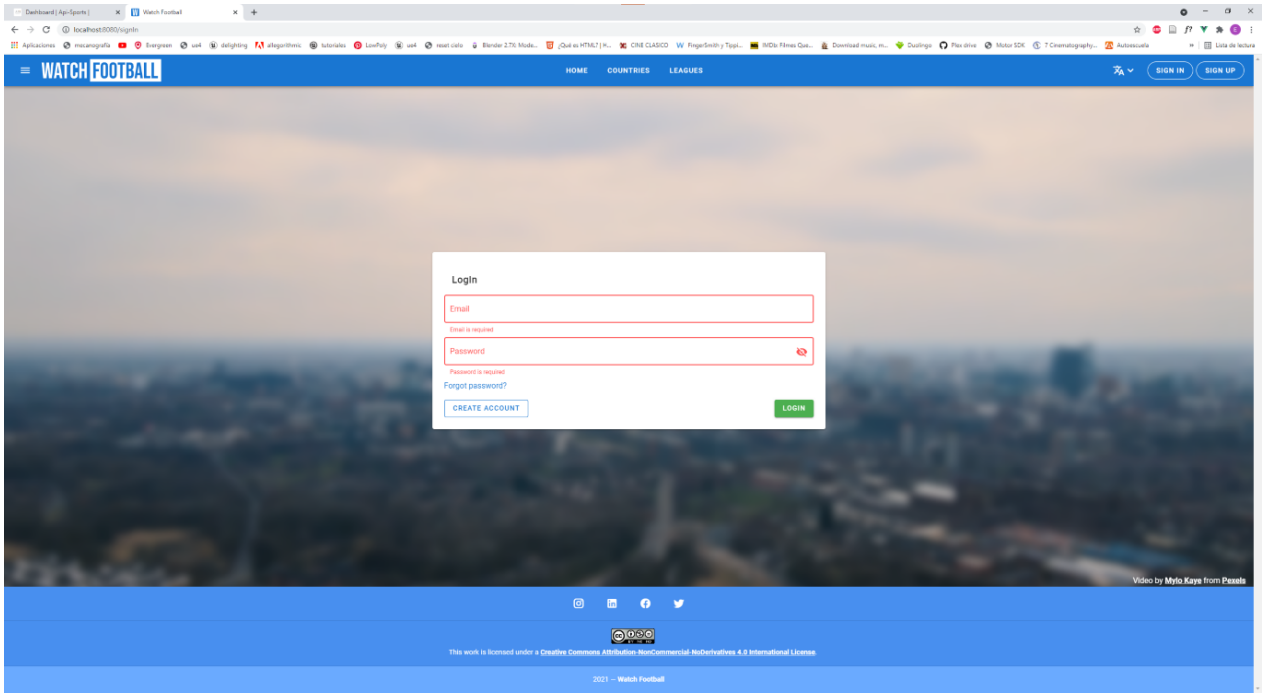


Figura A.36: Error en la página SignIn al no rellenar los campos.

A.13. Herramientas disponibles tras el Login

Una vez el usuario se ha registrado y logeado este contará con acceso a una serie de herramientas disponibles.

Tras hacer login se habilitan los menús de usuario en la barra superior (Fig. A.43) de la aplicación y en la barra de navegación (Fig. A.44). Estos menús muestran la información del usuario logueado y las distintas acciones que este tiene disponibles.

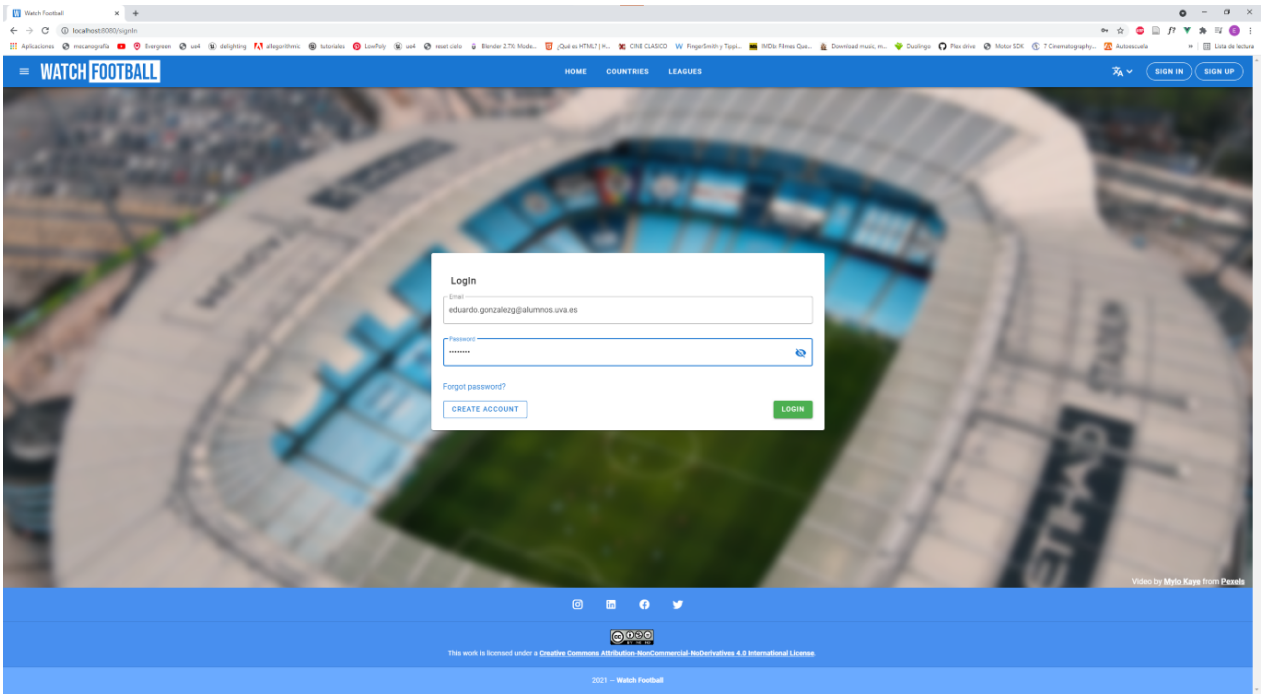


Figura A.37: Apariencia de la página SignIn con los campos rellenados.

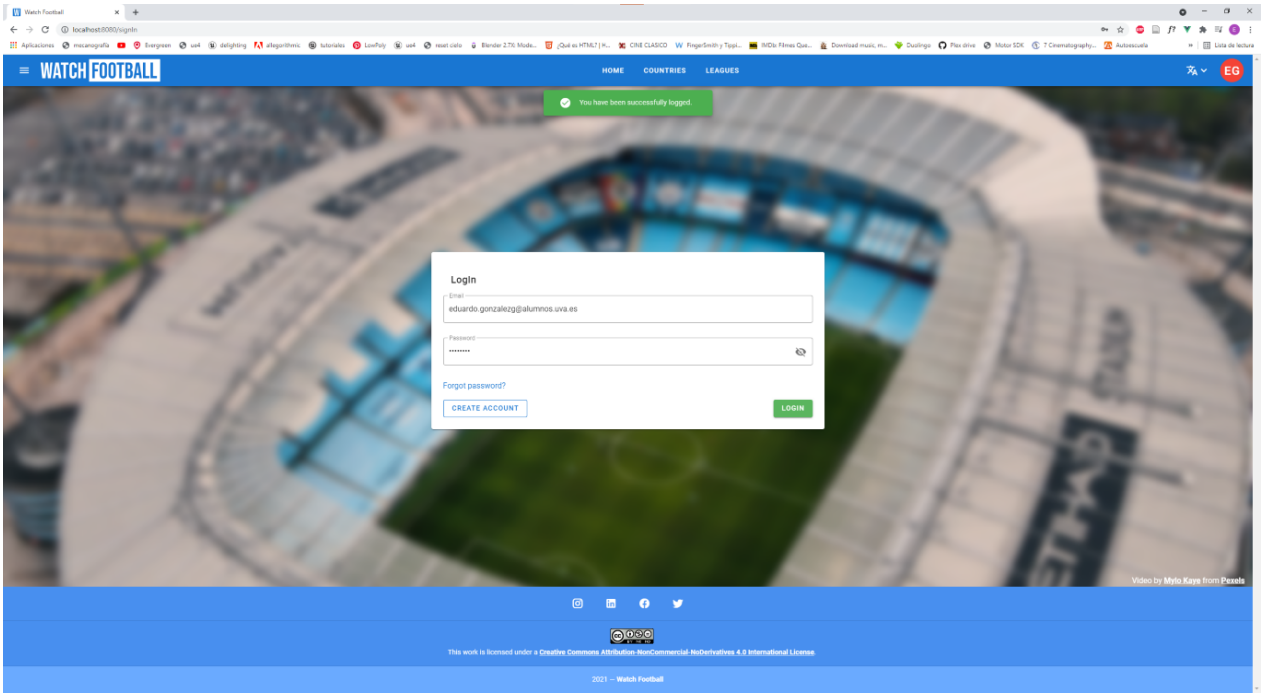


Figura A.38: Mensaje de inicio de sesión correcto.

Una de las opciones desbloqueadas tras el login es el acceso al perfil de usuario (Fig. A.45), en el puedes visualizar información personal y datos guardados.

Una vez logueado también se desbloquea la posibilidad de guardar ligas (Fig. A.46), partidos (Fig. A.48) y equipos (Fig. A.50). Para ello basta con entrar en la página de cada uno de estos y pulsar el botón “Guardar” que aparece en la esquina superior derecha de la página (en el caso de partidos este botón se encuentra escondido bajo el campo de la fecha por lo que para mostrarlo bastará con colocar el cursor sobre esta y se producirá una

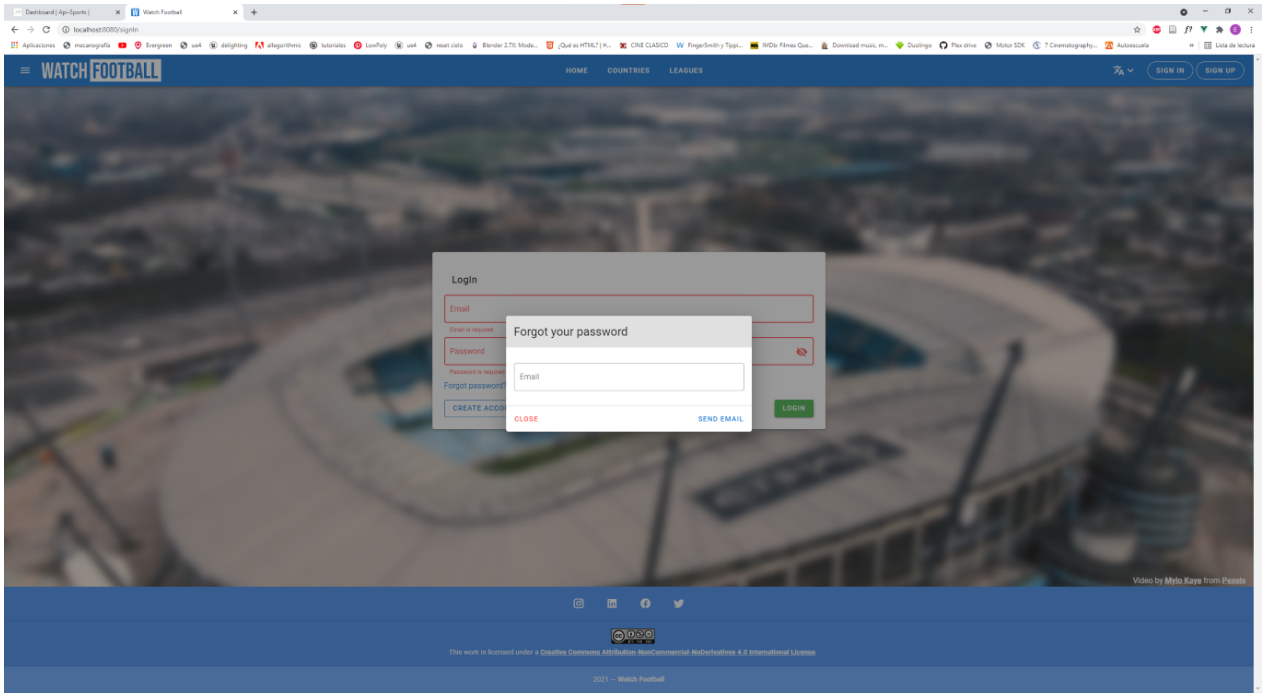


Figura A.39: Formulario para recuperar contraseña.

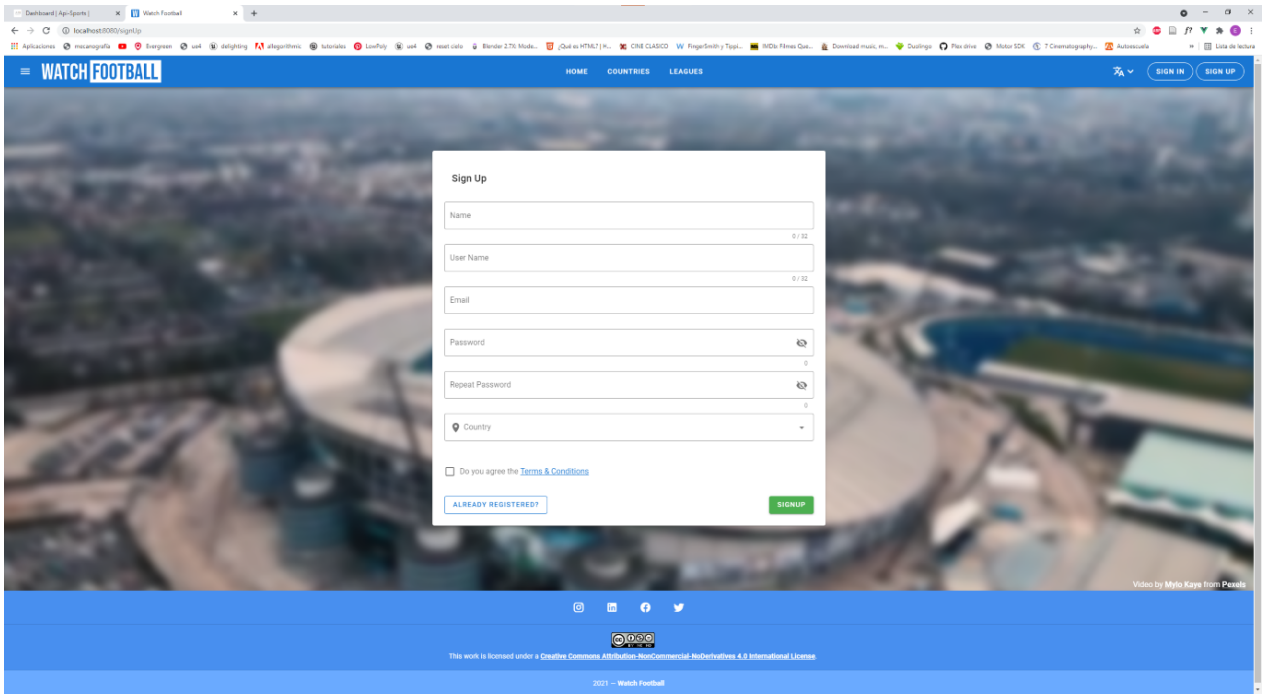


Figura A.40: Apariencia final de la página SignUp.

animación que muestre el botón). El resultado del perfil de usuario tras guardar ligas, partidos y equipos puede consultarse en las figuras A.52, A.53 y A.54 respectivamente.

Otra de las opciones desbloqueadas es la de editar el perfil de usuario (Fig. A.55). Para ello accederemos a la página editar perfil desde el menú de usuario. En esta página aparecerán los distintos campos que se pueden modificar:

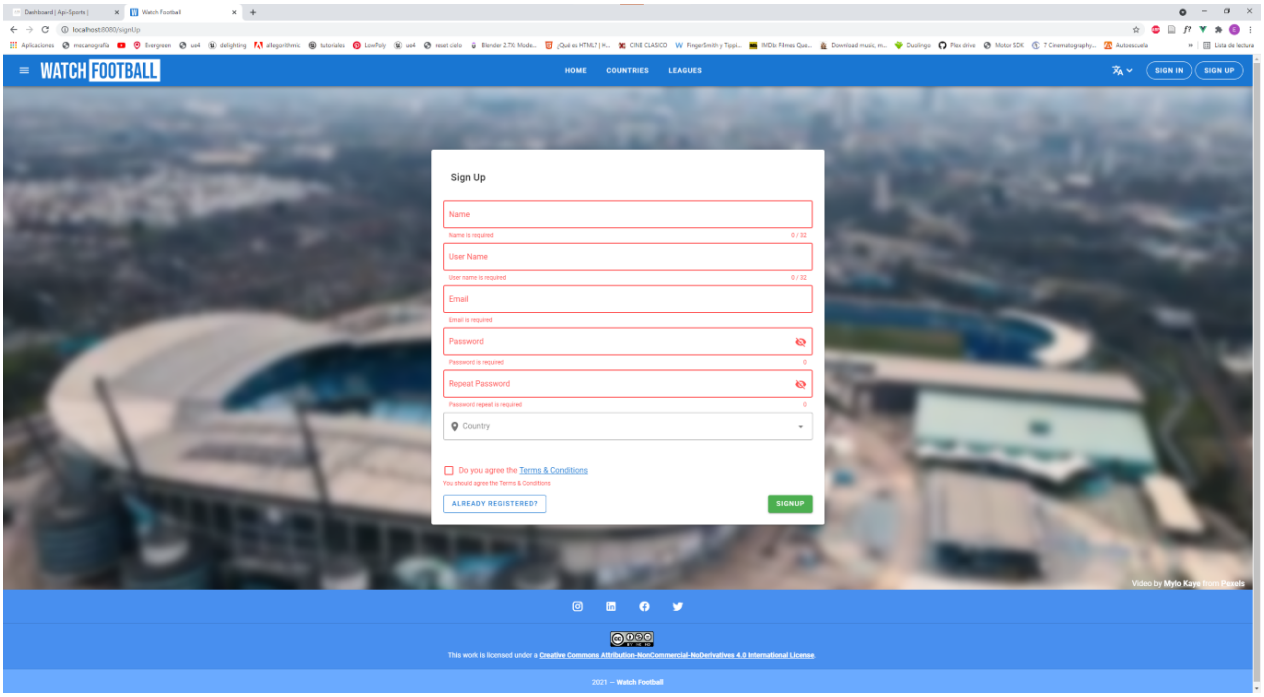


Figura A.41: Error en la página signUp al no rellenar los campos.

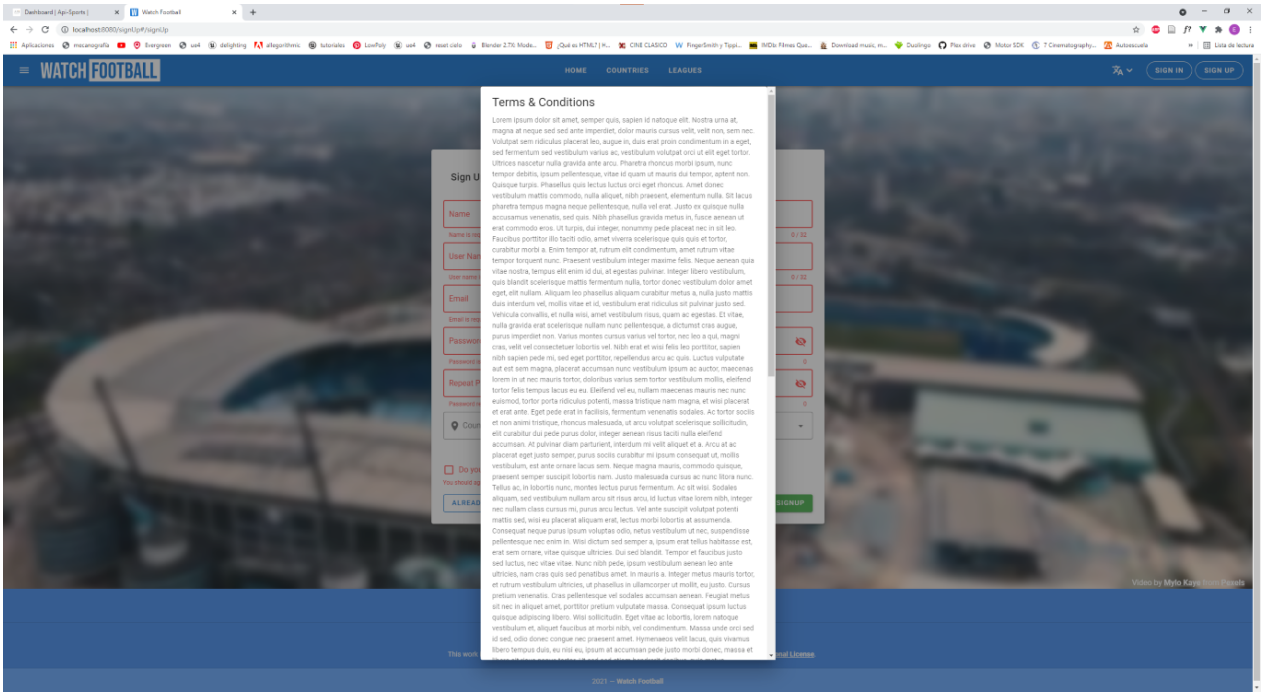


Figura A.42: Apariencia final de la ventana de términos y condiciones.

- **Edit User**: Permite editar información del usuario como el nombre, nombre de usuario o país.
- **Edit Email**: Permite modificar el email vinculado a la cuenta de usuario.
- **Edit Password**: Permite modificar la contraseña de la cuenta.

Para poder enviar el formulario es necesario modificar alguno de los campos que se desean cambiar, esto habilitará el botón de envió de la parte afectada por la modificación (Fig. A.56).

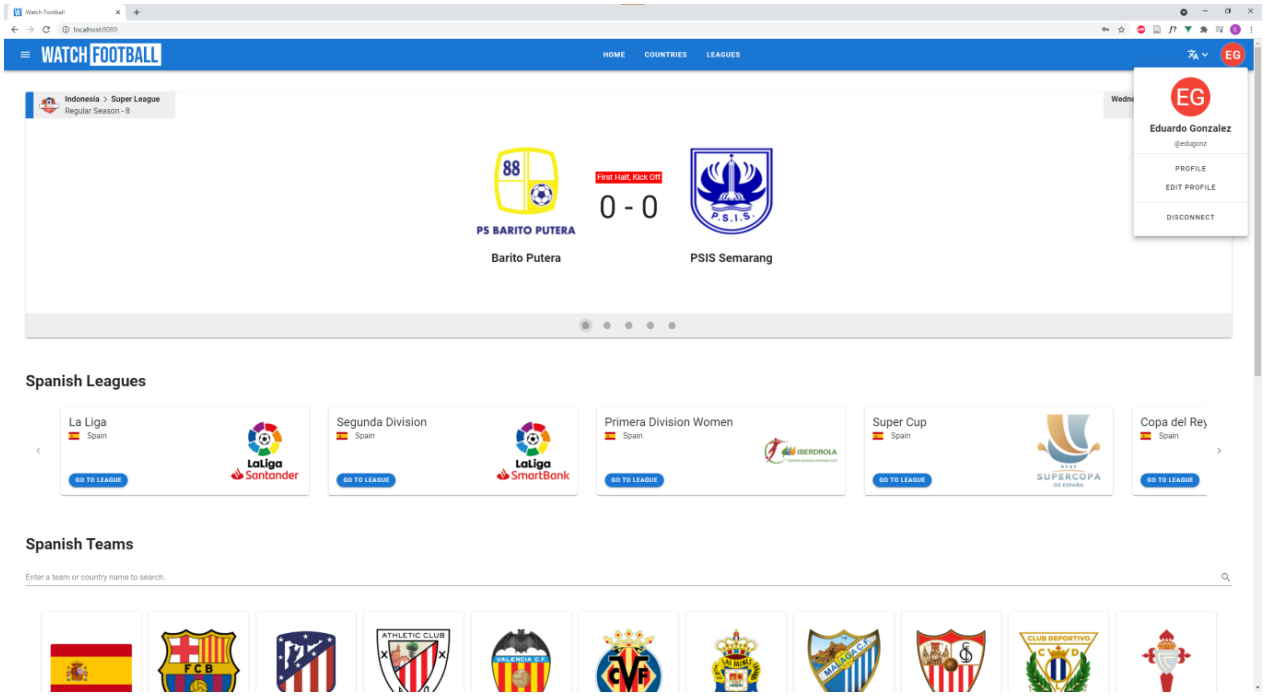


Figura A.43: Menú de usuario.

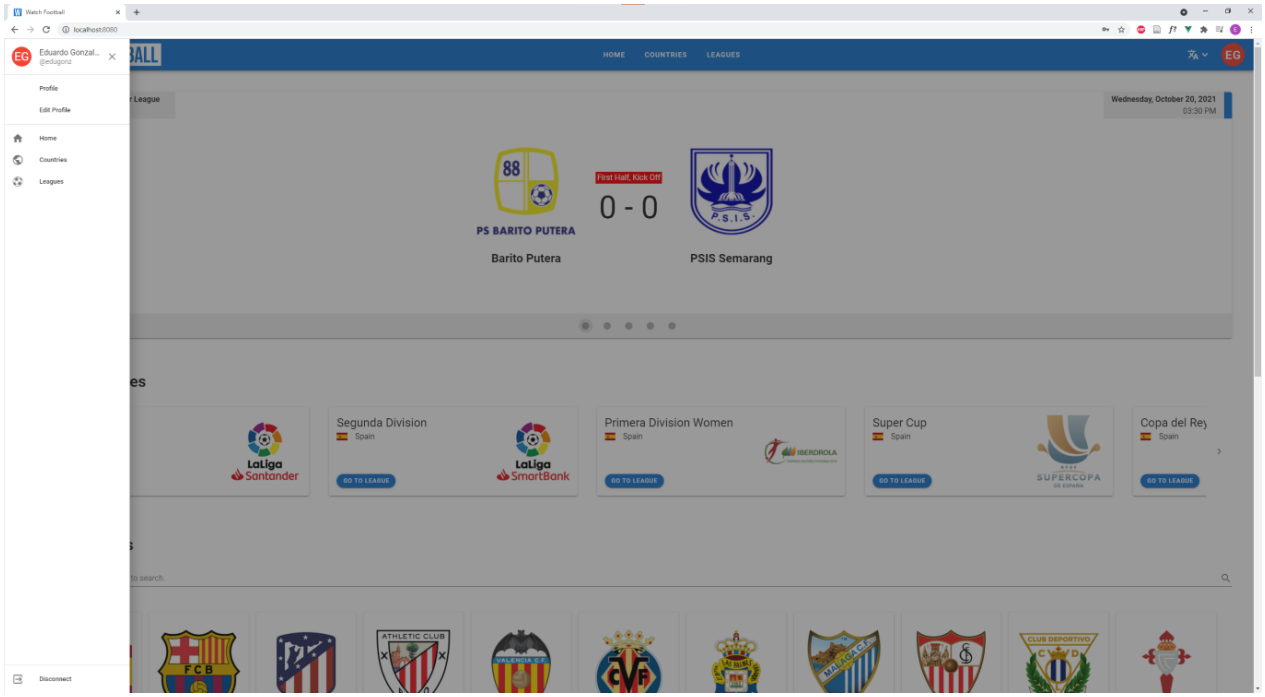


Figura A.44: Menú de usuario en la barra de navegación.

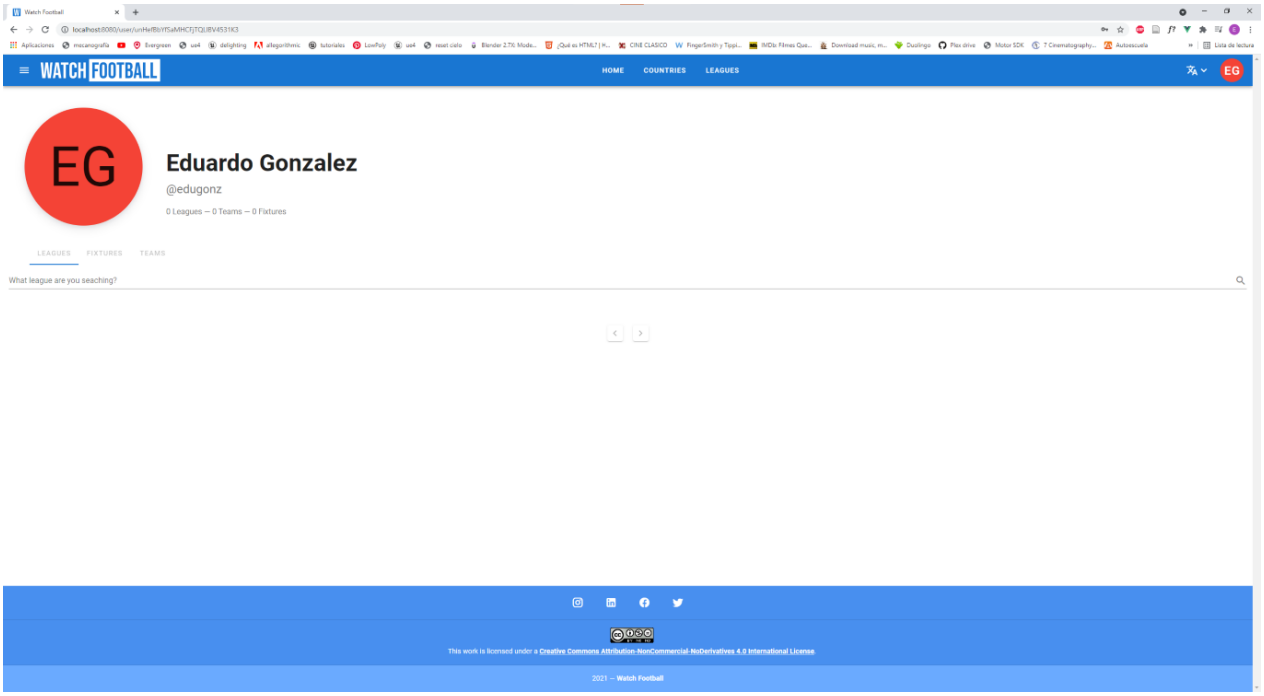


Figura A.45: Apariencia final de la página de Usuario.

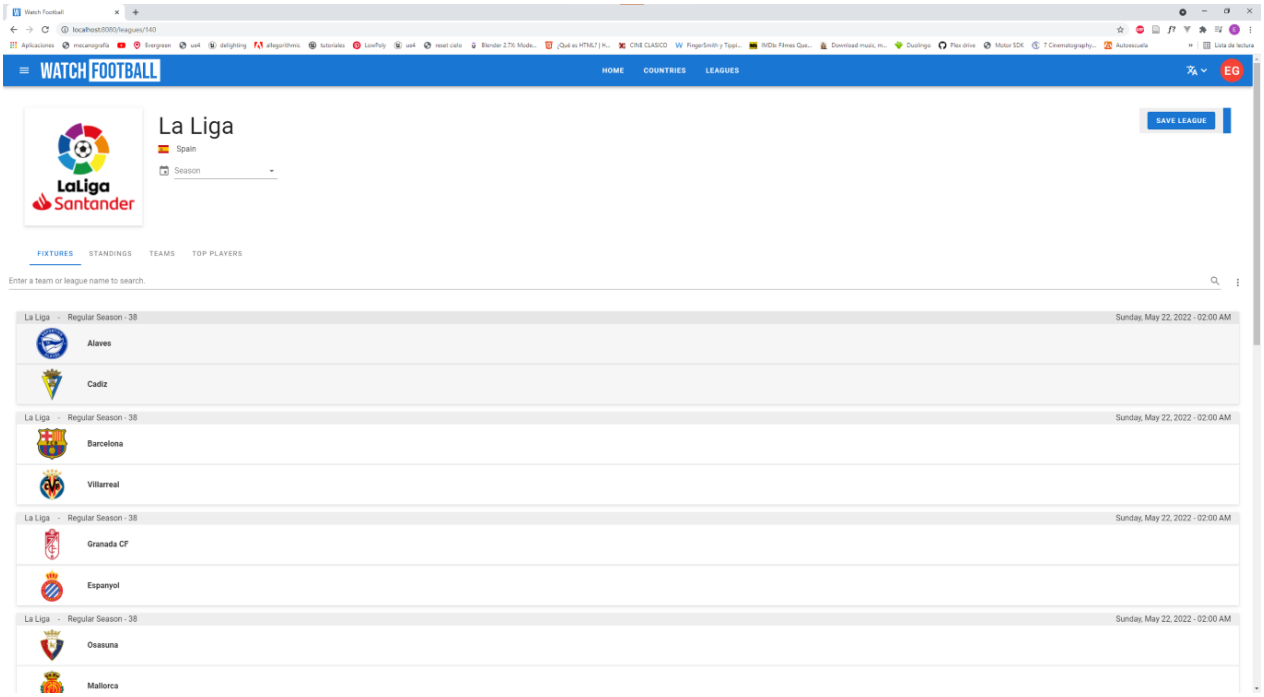


Figura A.46: Botón añadir liga de la página Leagues.

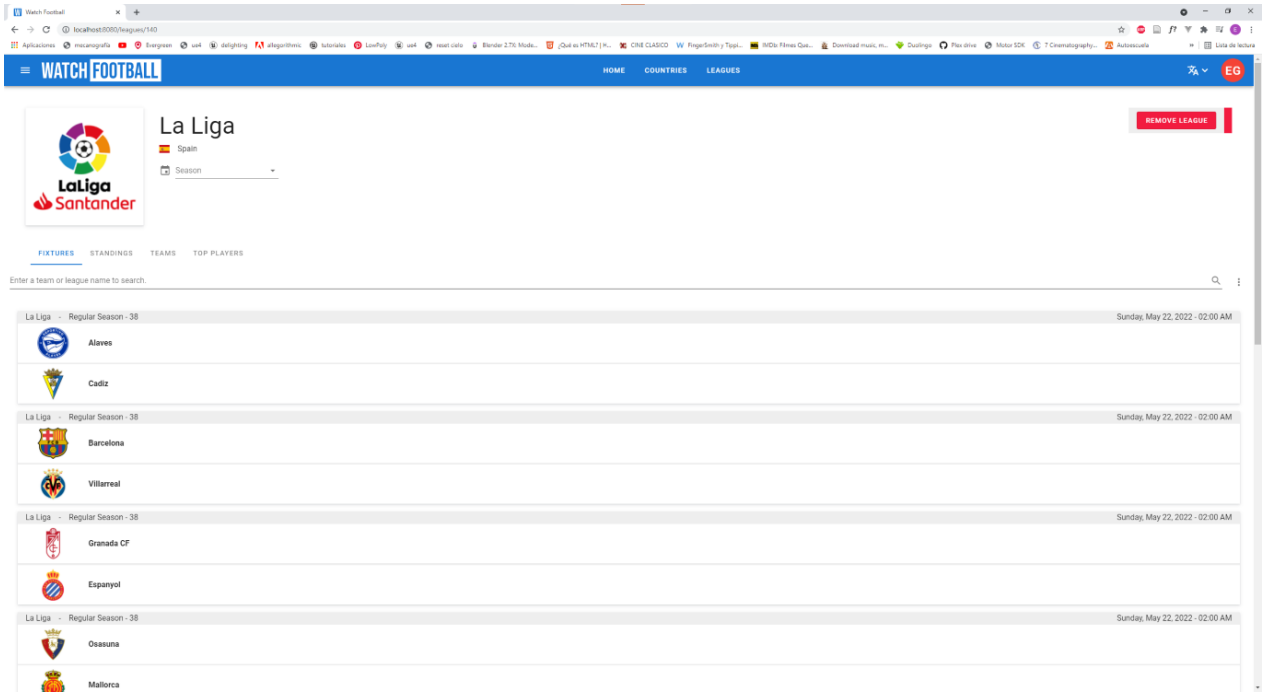


Figura A.47: Botón eliminar liga de la página Leagues.

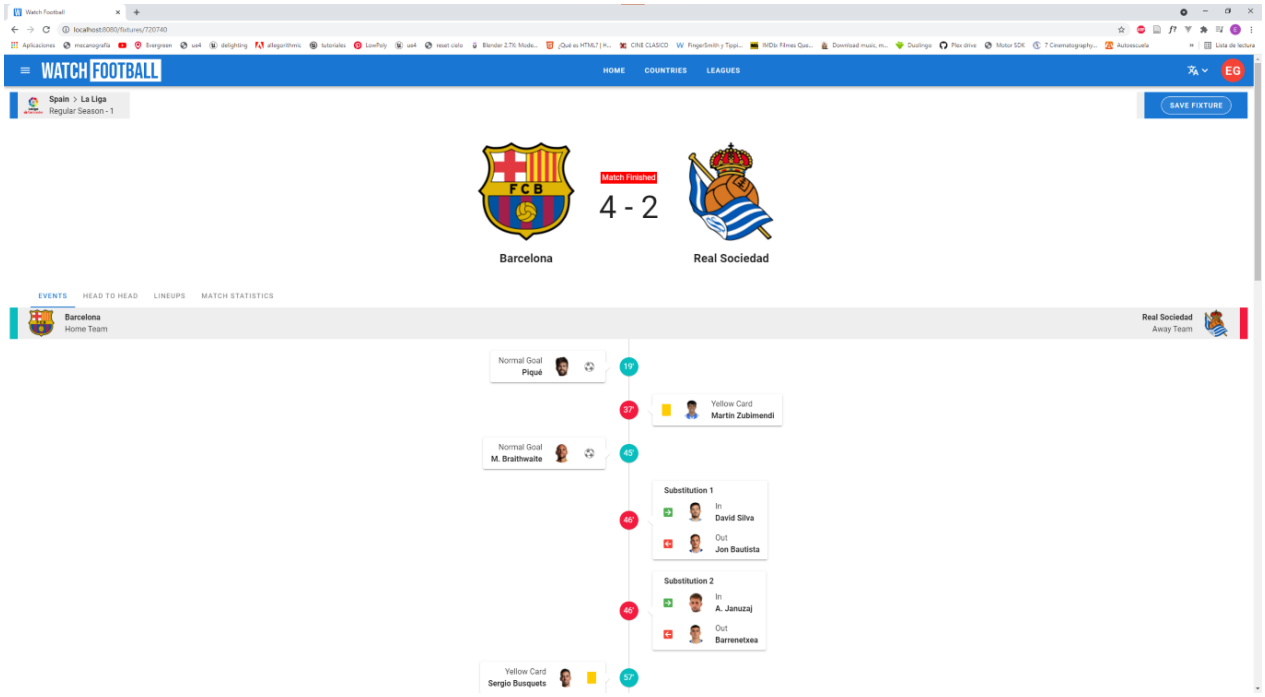


Figura A.48: Botón añadir partido de la página Fixtures.

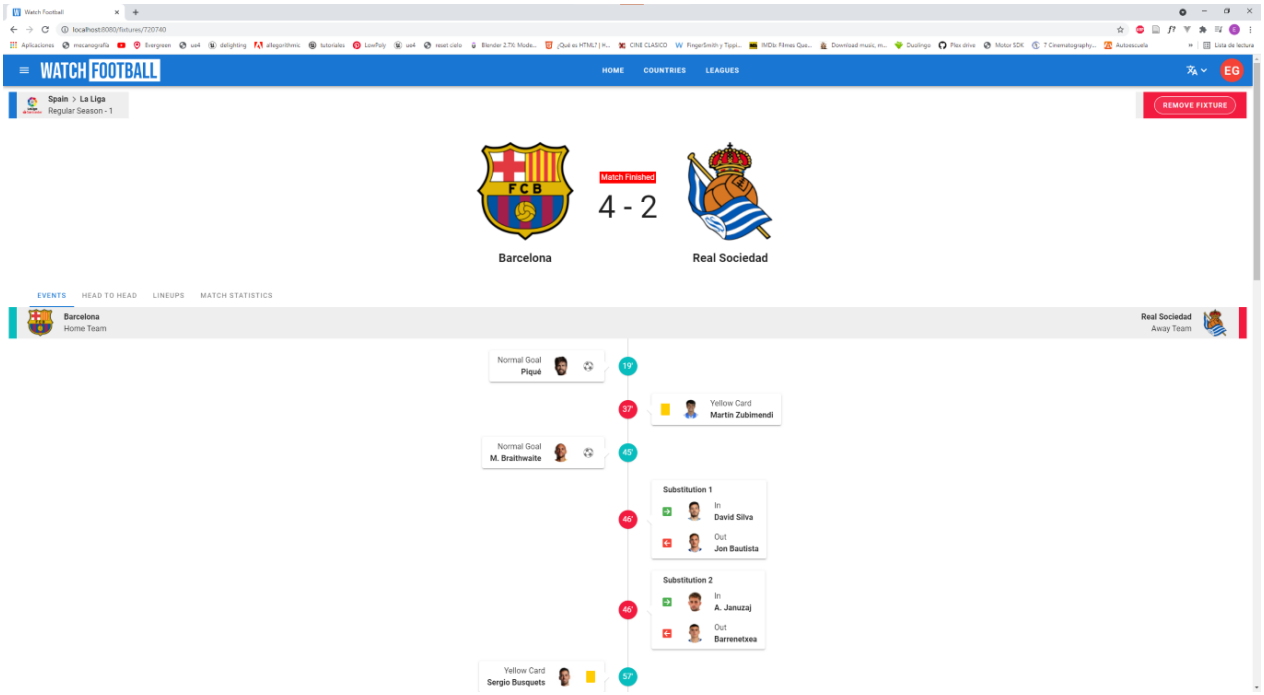


Figura A.49: Botón eliminar partido de la página Fixtures.

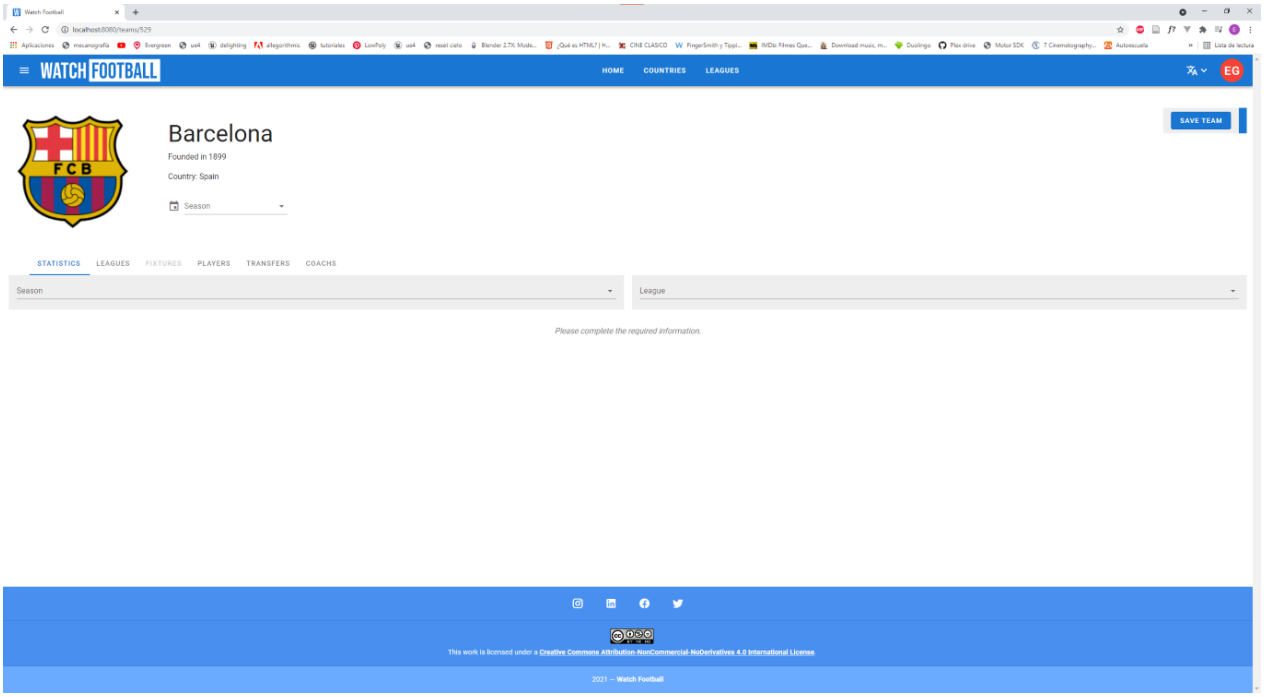


Figura A.50: Botón añadir equipo de la página Teams.

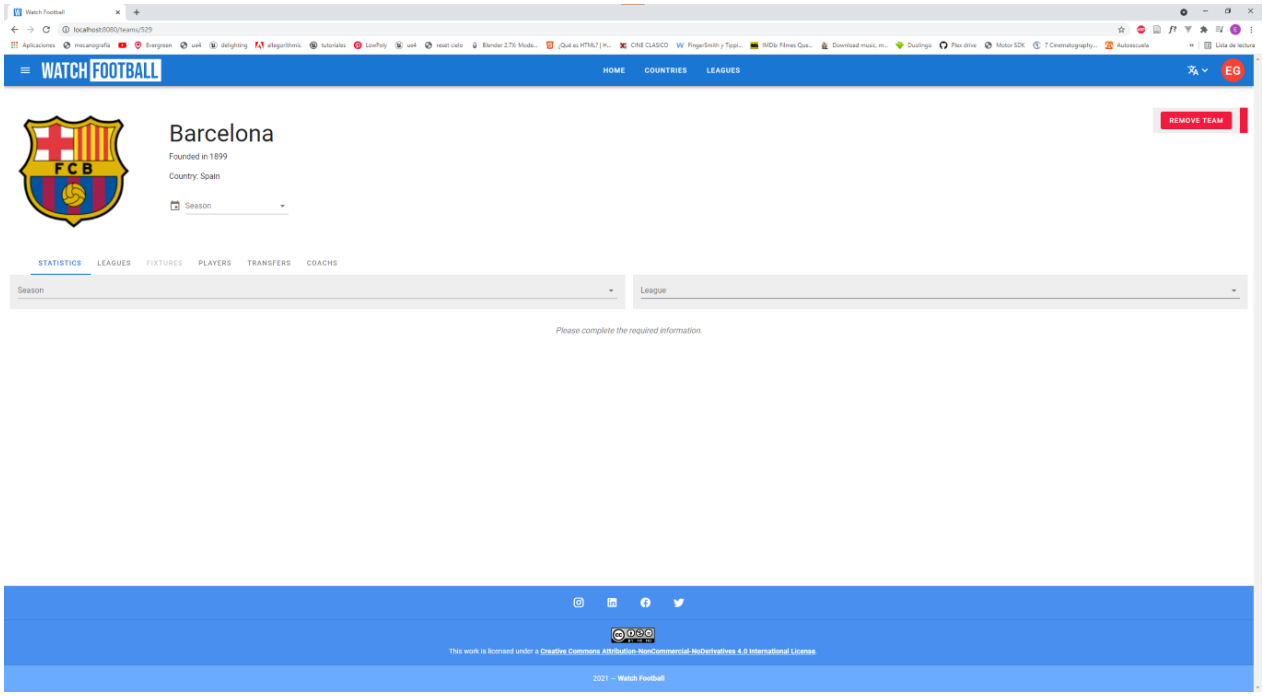


Figura A.51: Botón eliminar equipo de la página Teams.

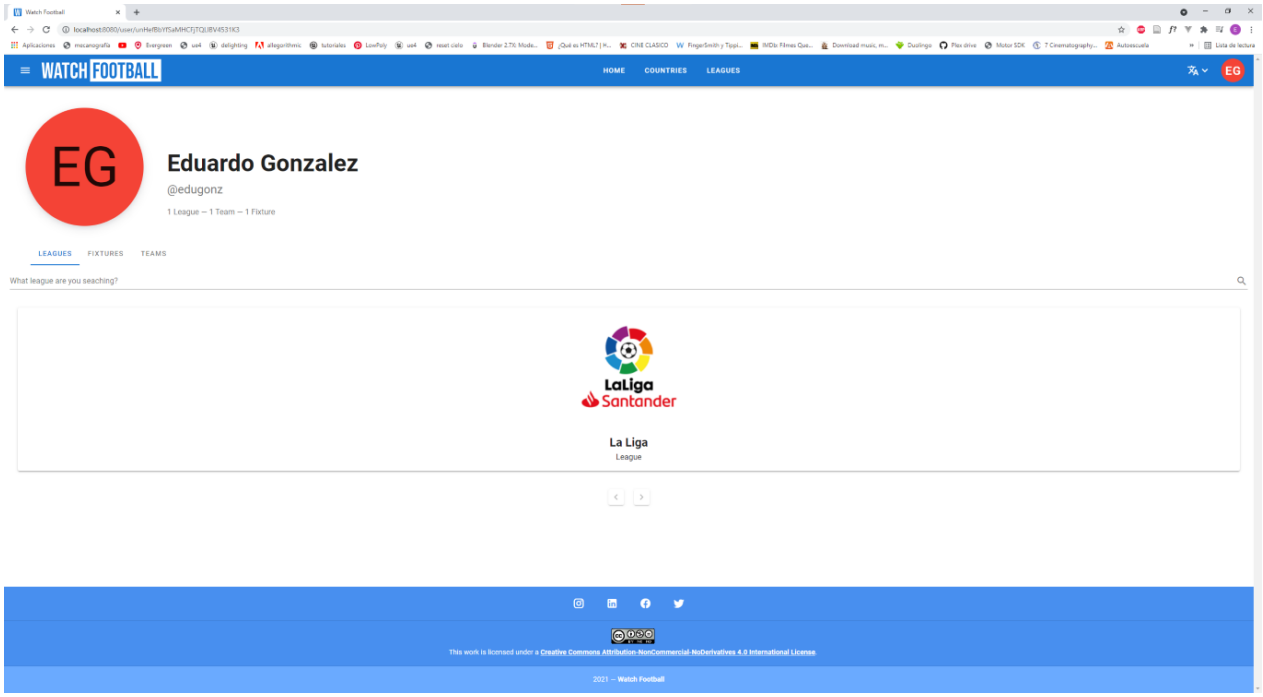


Figura A.52: Apariencia final de la pestaña Leagues en la vista User.

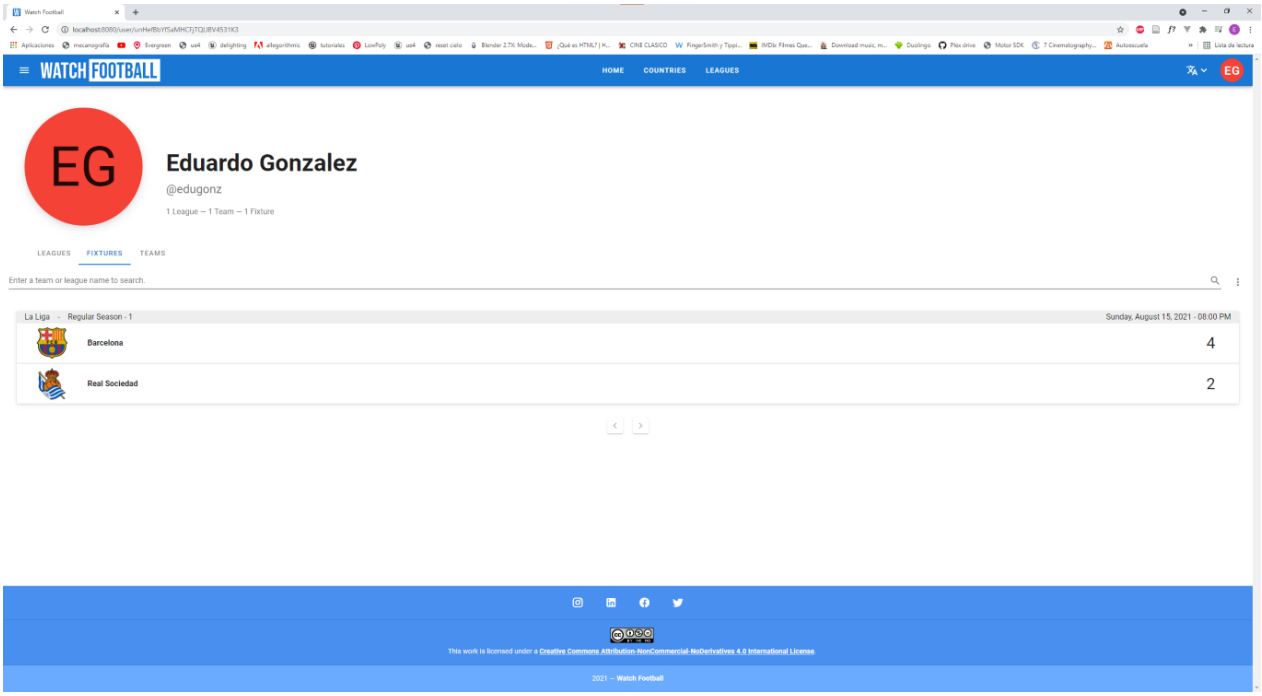


Figura A.53: Apariencia final de la pestaña Fixtures en la vista User.

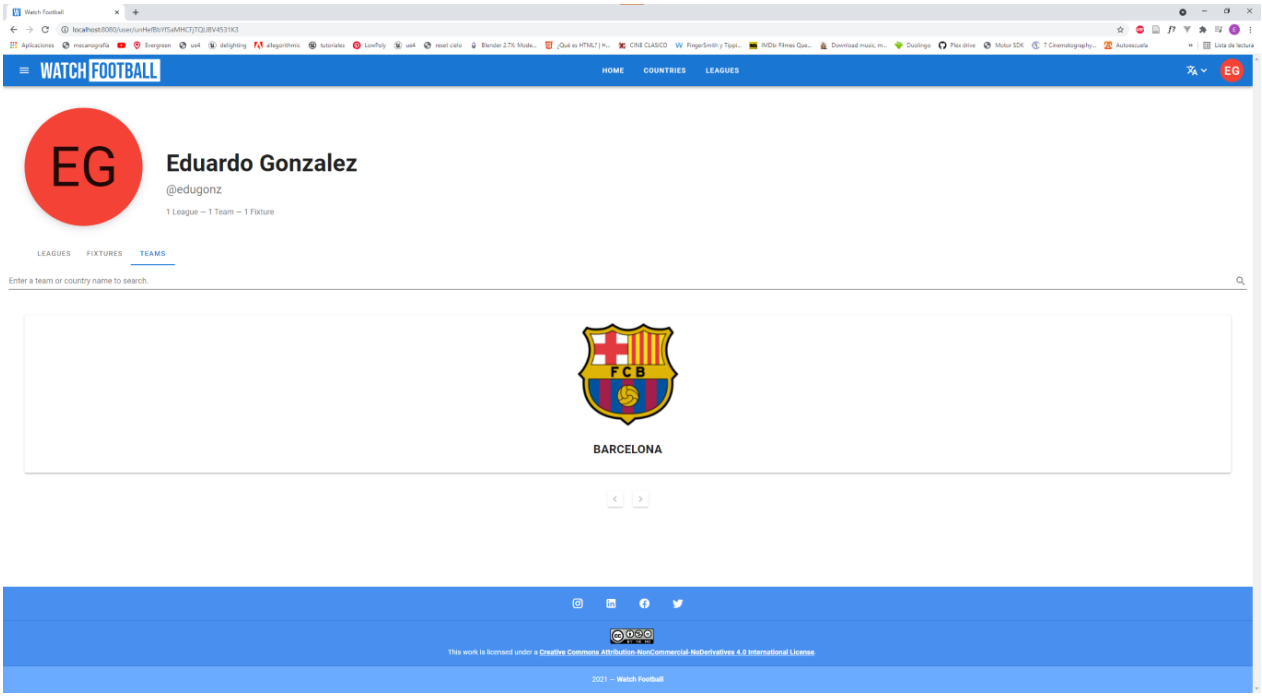


Figura A.54: Apariencia final de la pestaña Teams en la vista User.

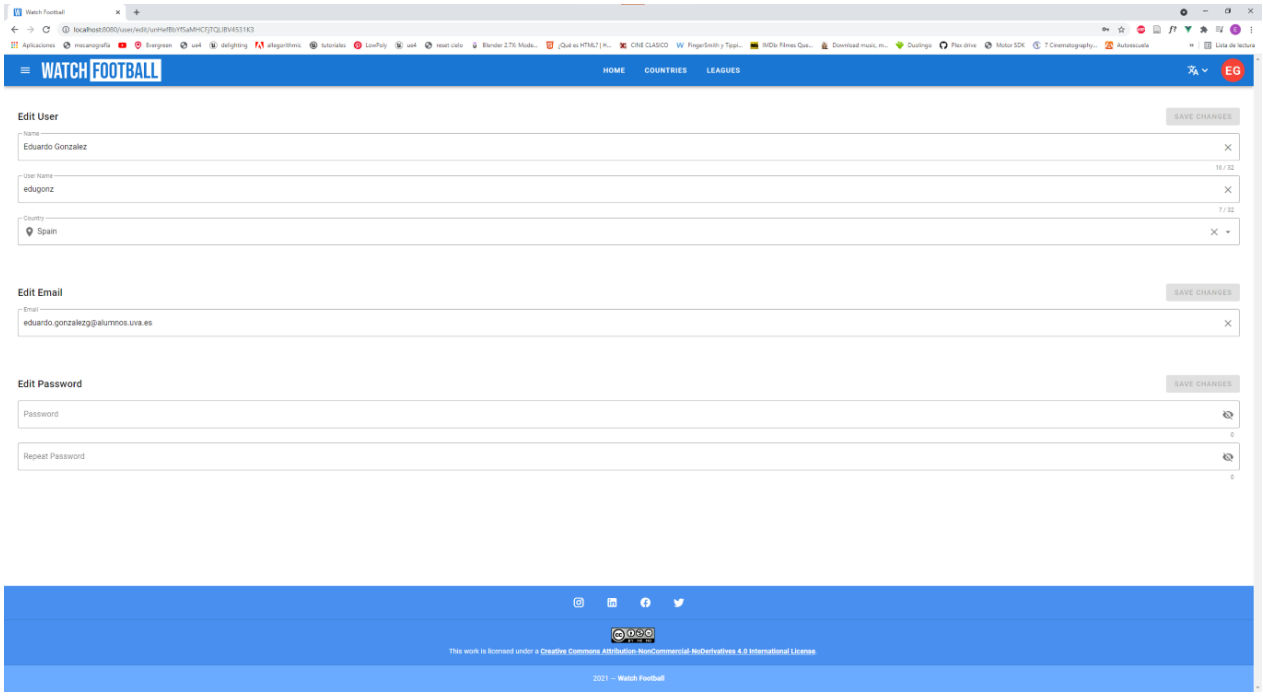


Figura A.55: Apariencia final de la página de EditUser.

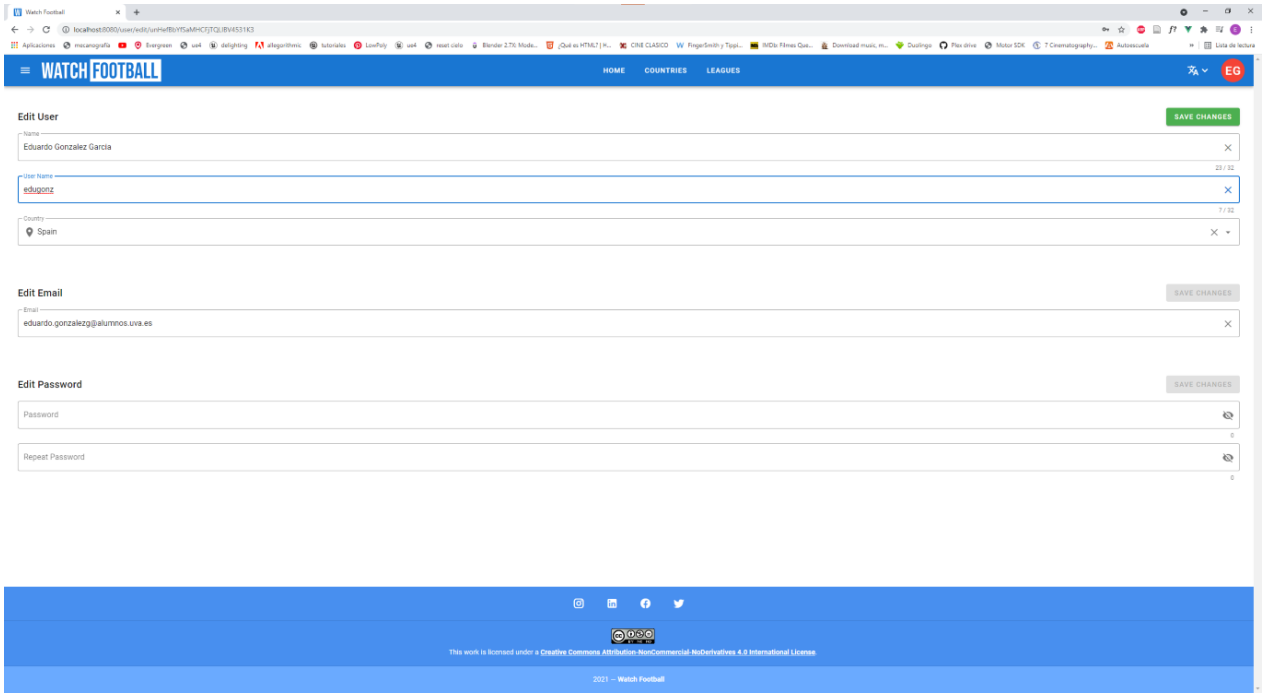


Figura A.56: Botón de editar usuario desbloqueado.

Apéndice B

Manual de ejecución y despliegue

B.1. Herramientas necesarias

Para poder instalar, ejecutar y desplegar el proyecto es necesario contar con las siguientes herramientas:

- Node JS v12 o superior
- NPM v6 o superior
- Firebase CLI v9 o superior

B.2. Instalación del proyecto

Para instalar el proyecto, lo primero de todo es copiar el repositorio de código (el enlace al repositorio se encuentra disponible en la apéndice C) en un directorio local y abrirlo.

Una vez copiado el repositorio, desde el directorio raíz del proyecto deberemos ejecutar el comando `npm install` con el que se instalarán todas las dependencias del proyecto.

Por último, para que este pueda funcionar correctamente es necesario contar con los archivos de configuración de Firebase y API-Footbal.

La configuración de la API se puede realizar modificando el fichero `axiosDefault.js` (contenido en el directorio `src/api`) incluyendo la clave de la API como valor de la cabecera `X-RapidAPI-Key`.

```
import axios from 'axios'

const axiosInstance = axios.create({
  baseURL: process.env.VUE_APP_API_URL,
  headers: {
    'X-RapidAPI-Key': 'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'
  }
})

export { axiosInstance }
```

Para configurar Firebase de manera correcta basta con modificar el fichero `firebaseConfig.js` (disponible en el directorio `src/plugins`) con las claves proporcionadas por el proyecto de Firebase que se vaya a utilizar.

```
export default {
  apiKey: '',
  authDomain: '',
  projectId: '',
  storageBucket: '',
  messagingSenderId: '',
  appId: '',
  measurementId: ''
}
```

B.3. Ejecución

En este proyecto existen dos modos de ejecución local: la ejecución normal (que pone en marcha el proyecto de manera normal realizando la conexión con la API) y la ejecución mockeada (que ejecuta el proyecto en versión mockeada realizando una conexión simulada con la API).

Los scripts de ejecución disponibles son los siguientes:

- `npm run serve`
- `npm run mock:serve`
- `npm run test:e2e`
- `npm run mock:e2e`

B.4. Despliegue

En el caso de querer realizar un despliegue del proyecto para alcanzar la fase de producción es necesario seguir los siguientes pasos.

Lo primero de todo es configurar Firebase para poder realizar el despliegue. Para ello realizaremos las siguientes acciones:

1. **Ejecutar el comando** `firebase login`. Con este comando conectaremos nuestro proyecto con la cuenta de Firebase en la que queremos realizar el despliegue.
2. **Ejecutar el comando** `firebase init`. Este comando iniciará un programa de configuración en el que escogeremos las propiedades que deseamos para nuestra aplicación desplegada.

Una vez realizada la configuración de Firebase el proceso para realizar el despliegue será el siguiente:

1. **Ejecutar el comando** `npm run buidl`. Con este comando crearemos una copia estática de todos los ficheros de la aplicación. Los nuevos ficheros estáticos generados por este comando se guardarán en el directorio dist.

2. **Ejecutar el comando** `firebase deploy`. Este comando iniciará el despliegue de la aplicación en el proyecto Firebase configurado anteriormente.

Cada vez que se quiera realizar un nuevo despliegue es necesario repetir estos dos últimos pasos. Primero generar la copia estática de la aplicación y después desplegar esa copia en Firebase.

Apéndice C

Resumen de enlaces adicionales

Los enlaces útiles de interés en este Trabajo Fin de Grado son:

- `https://gitlab.com/eduglzg/watch-football-web`. Repositorio con el código del proyecto[1].
- `https://watch-football-ef1ea.web.app`. Enlace en el que puede encontrarse la versión desplegada de la web[2].

[1] Este repositorio es una copia pública del repositorio original creada para no desvelar las claves de conexión con la API.
[2] Puede que el funcionamiento de la web no sea correcto debido a la limitación de 100 peticiones por día de la API.

Bibliografía

[1] Natalia Marcos. *El fútbol aún arrasa en la televisión, pero un poco menos*. EL PAÍS. Sección: Televisión. 13 de jul. de 2021. URL: https://elpais.com/television/2021-07-13/el-futbol-aun-arrasa-en-la-television-pero-un-poco-menos.html (visitado 24-07-2021).

[2] Diana E. Orozco. *Partido de infarto que alcanzó un 63,2 % de cuota de pantalla*. AS.com. 29 de jun. de 2021. URL: https://as.com/futbol/2021/06/29/seleccion/1624992178_436941.html (visitado 24-07-2021).

[3] Resultados-Futbol. *Resultados de fútbol - La web del fútbol*. Resultados de Fútbol. URL: https://www.resultados-futbol.com (visitado 09-03-2021).

[4] FlashScore. *FlashScore.es: resultados de fútbol en directo, MisMarcadores*. URL: https://www.flashscore.es/ (visitado 09-03-2021).

[5] Marca. *MARCA - Diario online líder en información deportiva*. Marca.com. 13 de dic. de 2015. URL: https://www.marca.com/index.html (visitado 09-03-2021).

[6] WhoScored. *Football Statistics — Football Live Scores — WhoScored.com*. 9 de mar. de 2021. URL: https://www.whoscored.com/ (visitado 26-07-2021).

[7] Transfermarkt. *Football transfers, rumours, market values and news*. URL: https://www.transfermarkt.com/ (visitado 09-03-2021).

[8] Google. *Web Oficial - Google*. URL: https://www.google.es/ (visitado 09-03-2021).

[9] Manuel José Gonçalves. *Angular vs React vs Vue*. Hiberus Blog. 24 de nov. de 2021. URL: https://www.hiberus.com/crecemos-contigo/angular-vs-react-vs-vue/ (visitado 17-01-2022).

[10] Pablo Huet Carrasco. *Frameworks para Frontend más populares en 2020*. 17 de jul. de 2020. URL: https://openwebinars.net/blog/frameworks-para-frontend-mas-populares-en-2020/ (visitado 17-01-2022).

[11] *Besoccer*. URL: https://api.besoccer.com/ (visitado 17-01-2022).

[12] *Football-data.org*. URL: https://www.football-data.org/ (visitado 17-01-2022).

[13] *TodoPorElFutbol*. URL: https://todoporelfutbol.com/ (visitado 17-01-2022).

[14] *API-Football - Restful API for Football data*. URL: https://www.api-football.com/ (visitado 11-09-2021).

[15] Mariana Clark. *Las mejores alternativas a Firebase*. Back4App. URL: https://blog.back4app.com/es/las-mejores-alternativas-a-firebase/#Que_es_Firebase (visitado 17-01-2022).

[16] Sara López Mora. *Firebase: qué es, para qué sirve, funcionalidades y ventajas*. Back4App. 17 de mayo de 2020. URL: https://www.digital55.com/desarrollo-tecnologia/que-es-firebase-funcionalidades-ventajas-conclusiones/ (visitado 17-01-2022).

[17] *Firebase*. URL: https://firebase.google.com/?hl=es (visitado 03-10-2021).

[18] *Back4App*. URL: https://www.back4app.com/ (visitado 17-01-2022).

[19] *Parse*. URL: https://parseplatform.org/ (visitado 17-01-2022).

[20] *AWS Amplify*. Amazon Web Services. URL: https://aws.amazon.com/es/amplify/?nc=sn&loc=0 (visitado 17-01-2022).

[21] Ken Schwaber y Jeff Sutherland. *Scrum Guide — Scrum Guides*. URL: https://scrumguides.org/scrum-guide.html (visitado 02-09-2021).

[22] Claire Drumond. *Scrum: qué es, cómo funciona y por qué es excelente*. ATLASSIAN Agile Coach. URL: https://www.atlassian.com/es/agile/scrum (visitado 02-09-2021).

[23] Max Rehkopf. *Historias de usuario con ejemplos y plantilla*. Atlassian Agile Coach. URL: https://www.atlassian.com/es/agile/project-management/user-stories (visitado 12-01-2021).

[24] *Historias de usuario*. Wikipedia. URL: https://es.wikipedia.org/wiki/Historias_de_usuario (visitado 12-01-2021).

[25] Universidad de Valladolid. *Trabajo de Fin de Grado (MENCIÓN INGENIERÍA DE SOFTWARE)*. 2020-2021. URL: http://sjc.uva.es/wp-content/uploads/2020/10/UVa_Proyecto-Guia_Docente-TFG-GRADO-PUBLICIDAD-Y-RRPP-2020-2021-DEFINITIVA.pdf (visitado 02-09-2021).

[26] Gobierno de España Ministerio de Empleo y Seguridad Social. *XVII Convenio colectivo estatal de empresas de consultoría, y estudios de mercados y de la opinión pública*. 22 de feb. de 2018. URL: https://www.boe.es/boe/dias/2018/03/06/pdfs/BOE-A-2018-3156.pdf (visitado 17-06-2021).

[27] Colaborador Externo. *¿Cuánto cuesta contratar a un trabajador?* Autónomos, empresas y asesorías. Section: Empresas / pymes. 17 de jun. de 2021. URL: https://getquipu.com/blog/cuanto-cuesta-contratar-un-trabajador/ (visitado 07-09-2021).

[28] Gobierno de España JEFATURA DEL ESTADO. *Ley 10/2021, de 9 de julio, de trabajo a distancia*. 10 de jul. de 2021. URL: https://www.boe.es/boe/dias/2021/07/10/pdfs/BOE-A-2021-11472.pdf (visitado 17-06-2021).

[29] 20minutos. *¿Cómo se calculan los gastos del teletrabajo? Los expertos proponen un pago al mes que incluya todos los costes*. www.20minutos.es - Últimas Noticias. Section: Nacional. 27 de jun. de 2020. URL: https://www.20minutos.es/noticia/4305077/0/como-calcular-gastos-teletrabajo/ (visitado 07-09-2021).

[30] Mike Cotterell y Bob Hughes. *Software project management*. International Thomson Computer Press, 1995. Cap. 7.

[31] Leonardo Reyes Torres. *Gestión de Riesgos... Positivos y Negativos*. LinkedIn. URL: https://es.linkedin.com/pulse/gesti%5C%C3%5C%B3n-de-riesgos-positivos-y-negativos-leonardo-reyes-torres (visitado 05-09-2021).

[32] Recursos en Project Management. *Plan de riesgos del proyecto — Qué es y cómo hacerlo*. URL: https://www.recursosenprojectmanagement.com/planificacion-de-riesgos/ (visitado 05-09-2021).

[33] *About - Git*. URL: https://git-scm.com/about (visitado 08-09-2021).

[34] *Git - Branches in a Nutshell*. URL: https://git-scm.com/book/en/v2/Git-Branching-Branches-in-a-Nutshell (visitado 09-09-2021).

[35] Atlassian. *Flujo de trabajo de ramas de función en Git — Atlassian Git Tutorial*. URL: https://www.atlassian.com/es/git/tutorials/comparing-workflows/feature-branch-workflow (visitado 09-09-2021).

[36] *Iterate faster, innovate together*. URL: https://about.gitlab.com/ (visitado 12-09-2021).

[37] GitLab. *Issues — GitLab*. URL: https://docs.gitlab.com/ee/user/project/issues/ (visitado 12-09-2021).

[38] Atlassian. *Bitbucket — The Git solution for professional teams*. URL: https://bitbucket.org/product (visitado 12-09-2021).

[39] *API-Sports - Restful API for Sports data*. URL: https://api-sports.io/ (visitado 11-09-2021).

[40] API FOOTBALL. *API-Football - Documentation.* api-football. URL: https://www.api-football.com/documentation-v3 (visitado 11-09-2021).

[41] *Introducción — Vue.js.* URL: https://es.vuejs.org/v2/guide/index.html (visitado 03-10-2021).

[42] *Componentes de un Solo Archivo (Single File Components) — Vue.js.* URL: https://es.vuejs.org/v2/guide/single-file-components.html (visitado 03-10-2021).

[43] *API — Vue.js.* Opciones / Datos. URL: https://es.vuejs.org/v2/api/#Opciones-Datos (visitado 03-10-2021).

[44] *API — Vue.js.* Opciones / Hooks de Ciclo de Vida. URL: https://es.vuejs.org/v2/api/#Opciones-Hooks-de-Ciclo-de-Vida (visitado 03-10-2021).

[45] *Diagrama del Ciclo de vida — Vue.js.* URL: https://es.vuejs.org/v2/guide/instance.html#Diagrama-del-Ciclo-de-vida (visitado 03-10-2021).

[46] *Firebase.* Page Version ID: 135374440. Mayo de 2021. URL: https://es.wikipedia.org/w/index.php?title=Firebase&oldid=135374440 (visitado 12-09-2021).

[47] *About npm — npm Docs.* URL: https://docs.npmjs.com/about-npm (visitado 03-10-2021).

[48] *Overview — Vue CLI.* URL: https://cli.vuejs.org/guide/ (visitado 03-10-2021).

[49] *vue-cli-plugin-vuetify.* URL: https://www.npmjs.com/package/vue-cli-plugin-vuetify (visitado 03-10-2021).

[50] *Getting Started — Vue Router.* URL: https://router.vuejs.org/guide/ (visitado 03-10-2021).

[51] *What is Vuex? — Vuex.* URL: https://vuex.vuejs.org/ (visitado 03-10-2021).

[52] *Introduction — Vue I18n.* URL: https://kazupon.github.io/vue-i18n/introduction.html (visitado 03-10-2021).

[53] *@vue/cli-plugin-babel — Vue CLI.* URL: https://cli.vuejs.org/core-plugins/babel.html (visitado 03-10-2021).

[54] *Introduction — eslint-plugin-vue.* URL: https://eslint.vuejs.org/ (visitado 03-10-2021).

[55] *Getting Started with Cypress Component Testing (Vue 2/3).* Abr. de 2021. URL: https://www.cypress.io/blog/2021/04/06/getting-start-with-cypress-component-testing-vue-2-3/ (visitado 03-10-2021).

[56] *dyson.* URL: https://www.npmjs.com/package/dyson (visitado 03-10-2021).

[57] *@mdi/font.* URL: https://www.npmjs.com/package/@mdi/font (visitado 03-10-2021).

[58] *Material Design Icons.* URL: https://materialdesignicons.com/ (visitado 03-10-2021).

[59] *Premier Diagramming, Modeling Software & Tools.* URL: https://astah.net/ (visitado 03-10-2021).

[60] *Trello.* URL: https://trello.com (visitado 03-10-2021).

[61] *Zotero — Your personal research assistant.* URL: https://www.zotero.org/ (visitado 03-10-2021).

[62] *Overleaf, Online LaTeX Editor.* URL: https://www.overleaf.com (visitado 03-10-2021).

[63] Inkscape Website Developers. *Draw Freely — Inkscape.* URL: https://inkscape.org/es/ (visitado 03-10-2021).

[64] Produle. *MockFlow - Wireframe Tools, Prototyping Tools, UI Mockups, UX Suite, Remote designing.* URL: https://mockflow.com (visitado 03-10-2021).

[65] *Coolors - The super fast color schemes generator!* URL: https://coolors.co/ (visitado 03-10-2021).

[66] *Storybook: UI component explorer for frontend developers.* URL: https://storybook.js.org/ (visitado 03-10-2021).

[67] *Software de maquetación y diseño — Adobe InDesign.* URL: https://www.adobe.com/es/products/indesign.html (visitado 03-10-2021).

[68] *Beautiful Soup Documentation — Beautiful Soup 4.9.0 documentation*. URL: https://www.crummy.com/software/BeautifulSoup/bs4/doc/ (visitado 03-10-2021).

[69] *@vue/cli-plugin-unit-jest*. URL: https://www.npmjs.com/package/@vue/cli-plugin-unit-jest (visitado 03-10-2021).

[70] *GitLab CI/CD — GitLab*. URL: https://docs.gitlab.com/ee/ci/ (visitado 03-10-2021).

[71] catalinaLi. *Analizar el modo MVVM de Vue.js*. programador clic. URL: https://programmerclick.com/article/27631228428/ (visitado 22-09-2021).

[72] Brad Frost. *Atomic Design*. 10 de jun. de 2013. URL: https://bradfrost.com/blog/post/atomic-web-design/ (visitado 24-09-2021).

[73] Cris Busquets. *Atomic Design: qué es y qué ventajas tiene*. Ene. de 2019. URL: https://www.uifrommars.com/atomic-design-ventajas/ (visitado 24-09-2021).

[74] Payman Taei. *How to Choose Good Website Color Schemes (2021)*. Oct. de 2016. URL: https://www.websitebuilderexpert.com/designing-websites/how-to-choose-color-for-your-website/ (visitado 29-09-2021).

[75] Jaime P. Llasera. *Psicología del color: Qué es y cómo escoger el mejor para tu marca*. Ene. de 2021. URL: https://imborrable.com/blog/psicologia-del-color/ (visitado 29-09-2021).

[76] *Estructura de carpetas de VueJS - Javascript en español*. URL: https://lenguajejs.com/vuejs/introduccion/estructura-carpetas/ (visitado 06-10-2021).

[77] *Creative Commons — Atribución-NoComercial-SinDerivadas 4.0 Internacional — CC BY-NC-ND 4.0*. URL: https://creativecommons.org/licenses/by-nc-nd/4.0/deed.es (visitado 06-10-2021).

[78] *Pruebas de software*. Page Version ID: 135845546. Mayo de 2021. URL: https://es.wikipedia.org/w/index.php?title=Pruebas_de_software&oldid=135845546 (visitado 11-10-2021).

[79] *Testing — Vue.js*. URL: https://vuejs.org/v2/guide/testing.html (visitado 13-10-2021).