

# KING OF GOAL

MEMORIA

ALEJANDRO ACUAVIVA PLAZUELO

2º DAW

## Índice

Introducción .....	- 2 -
Descripción .....	- 4 -
Organización .....	- 4 -
Funcionalidades .....	- 6 -
Diseño .....	- 7 -
Arquitectura .....	- 7 -
Entidad Relación .....	- 8 -
Mapa web .....	- 9 -
Desarrollo .....	- 9 -
Preparación .....	- 9 -
V 0.1 .....	- 10 -
V 0.2 .....	- 11 -
V 0.3 .....	- 13 -
V 0.4 .....	- 14 -
V 0.5 .....	- 16 -
V 0.6 .....	- 18 -
V 0.7 .....	- 20 -
Pruebas .....	- 21 -
Instalación y despliegue .....	- 23 -
Conclusiones .....	- 25 -
Índice de imágenes .....	- 26 -
Bibliografía .....	- 27 -

## Introducción


La idea principal de **King Of Goal** está relacionada con el entretenimiento dentro del mundo del fútbol. Consiste en un juego sobre cartas de futbolistas donde el usuario crea un club en el cual debe reunir dichas cartas para montar las mejores plantillas posibles, ganando monedas en un minijuego e invirtiéndolas en abrir sobres para conseguir dichas cartas. El **objetivo** es que la aplicación a desarrollar logre todas estas funcionalidades descritas.

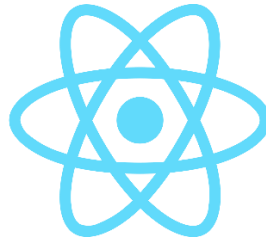


Ilustración 1: Introducción

Como **antecedente** principal, se encuentra un modo de juego del videojuego para PS4 FIFA. También existen varias aplicaciones acerca de este juego, aunque tan solo como aplicaciones móviles, y en algunas las interfaces de las plantillas son incómodas para el usuario.

En cuanto a las **tecnologías** para desarrollar la aplicación, se emplearán las siguientes:

-  **React.js:** Es un **framework** de **Javascript** de código abierto diseñado para crear interfaces de usuario con el objetivo de facilitar el desarrollo de aplicaciones en la arquitectura SPA (Single Page Application). Se ha hecho uso de esta tecnología por la gran organización del proyecto por vistas y componentes pudiendo así reutilizar mucho código en cuanto al desarrollo; y por la arquitectura SPA mencionada anteriormente ya que da una experiencia más fluida al usuario, cargando de una sola vez todo el código HTML, CSS y JavaScript sin tener que volver a cargarse al cambiar de ruta.

*Ilustración 2: React.js*

✚ **Lumen:** Es un **microframework** escrito en **PHP** basado en **Laravel** especializado en hacer **API REST FULL**. Se ha hecho uso de este microframework para complementar al lado cliente desarrollado en React.js utilizándolo de API REST.

*Ilustración 3: Lumen*

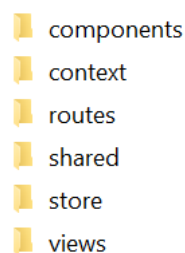
✚ **MySQL:** Es un sistema gestor de base de datos para bases de datos relacionales. Utiliza múltiples tablas para almacenar y organizar la información. Es escrito en C y C++ y destaca por su gran adaptación a diferentes entornos de desarrollo, permitiendo su interacción con los lenguajes de programación más utilizados como PHP, Perl y Java y su integración en distintos sistemas operativos. Se ha utilizado este gestor ya que es de los más adaptables a cualquier entorno.

*Ilustración 4: MySQL*

## Descripción

### Organización

Como se ha mencionado anteriormente, el lado **cliente** de la aplicación ha sido desarrollado con el framework **React.js**. El código de este queda organizado de la siguiente forma:

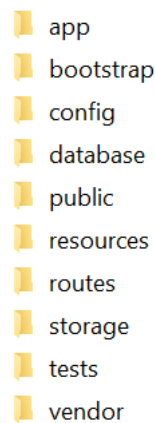


*Ilustración 5: Cliente*






- ✚ **Components:** Dentro están todos los componentes de la aplicación. Se entiende como componente un fragmento de código que se va a repetir en varias vistas, como por ejemplo un botón con un estilo predeterminado o una ventana modal con una funcionalidad implementada que tendrá el mismo funcionamiento en distintas vistas.
- ✚ **Views:** Aquí se encuentran todas las vistas de la aplicación. Cada una contiene dos archivos: el que contiene la estructura y las funcionalidades; y el otro que contiene los estilos.
- ✚ **Routes:** Donde se reúnen todos los archivos relacionados con las rutas de la aplicación, tanto objetos para crear las rutas como para configurar los menús.
- ✚ **Shared:** Aquí se encuentran estilos, variables y funciones que se van a utilizar a lo largo de todas las vistas de la aplicación.
- ✚ **Context y store:** En estos directorios se monta una estructura para almacenar las peticiones que se van a realizar al servidor, los datos que se reciben; y tener estos datos organizados en secciones.

Una vez desarrollado todo el código cliente, se compila se generará un fichero el cual se deberá añadir al servidor.

Luego, la organización del servidor es la siguiente:



*Ilustración 6: Servidor*

-  **App:** Dentro están los controladores con los métodos pertenecientes a cada ruta.
-  **Database:** Aquí se almacenan todas las migraciones que se ejecutarán para crear las tablas de la base de datos.
-  **Resources:** Contiene el fichero generado una vez compilado el cliente que muestra la vista de la aplicación.
-  **Routes:** Contiene un fichero con todas las rutas a las cuáles se harán las peticiones.
-  **Storage:** Dentro se almacenan todas las imágenes empleadas en la aplicación, tanto estáticas como las que se subirán al servidor.

Para la obtención de los datos en el lado cliente, desde este se envían las **peticiones http** (con parámetros o no) a las **url** programadas en el servidor. En él se realiza la conexión con la base de datos mediante **MySQL** para realizar las consultas y buscar los

datos ahí; y una vez obtenidos estos, el servidor los envía al cliente en un **JSON**, funcionando así como una **API REST**.

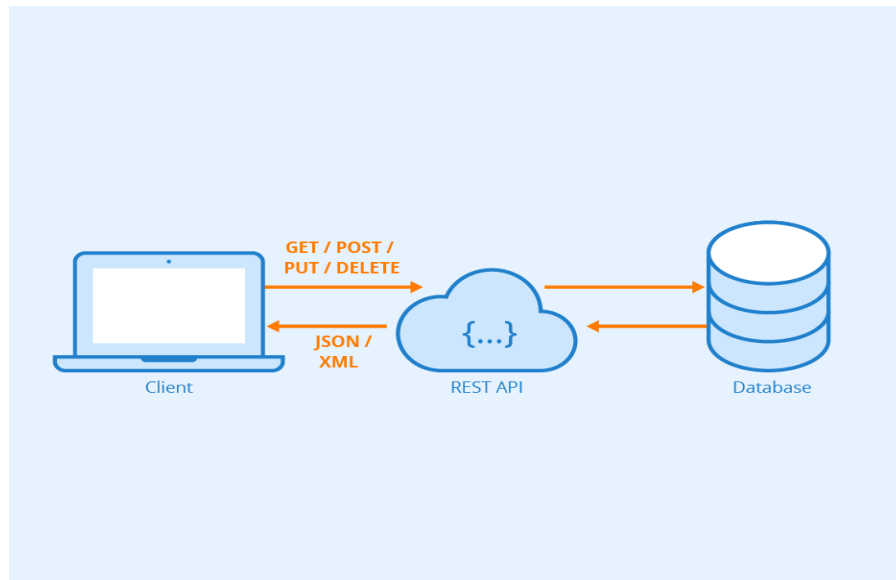


Ilustración 7: API REST

## Funcionalidades

En la aplicación han sido implementadas las siguientes funcionalidades:

- ✓ Validación de formularios.
- ✓ Pantalla de carga mientras se realizan las peticiones.
- ✓ Registro de usuarios almacenando la contraseña hasheada.
- ✓ Login de usuarios con un sistema de token identificando al usuario logueado.
- ✓ Sistema de roles y permisos.
- ✓ Control de las vistas para usuarios logueados o administradores.
- ✓ Creación del club del usuario.
- ✓ Implementación de un mensaje de notificación para informar al usuario de errores o de acciones con éxito.
- ✓ Middlewares para controlar si el usuario tiene los permisos necesarios para ejecutar una acción.
- ✓ Menú lateral desplegable.
- ✓ Migas de pan para volver a secciones anteriores con más facilidad.
- ✓ Botones para volver a la sección anterior.
- ✓ Paginación.
- ✓ Cambiar el nombre y el escudo del club.
- ✓ Borrar el club.

- ✓ Insertar, eliminar y modificar permisos.
- ✓ Insertar, eliminar y modificar jugadores.
- ✓ Insertar, eliminar y modificar países.
- ✓ Insertar, eliminar y modificar ligas.
- ✓ Insertar, eliminar y modificar equipos.
- ✓ Insertar, eliminar y modificar tipos de cartas.
- ✓ Insertar, eliminar y modificar cartas.
- ✓ Insertar, eliminar y modificar sobres.
- ✓ Filtrar jugadores.
- ✓ Vender un jugador a cambio de monedas.
- ✓ Abrir sobres en los cuales salen una cantidad de jugadores, programado con un algoritmo el cual, a mayor media, mayor será la probabilidad que salga. El jugador tendrá que pagar una cantidad de monedas para abrirlo, y los jugadores que salgan los podrá guardar en su club.
- ✓ Crear plantillas con los jugadores del club.
- ✓ Modificar el nombre de una plantilla.
- ✓ Borrar una plantilla.
- ✓ Colocar jugadores en la plantilla.
- ✓ Quitar jugadores de la plantilla.
- ✓ Intercambiar jugadores de una posición a otra.
- ✓ Jugar al draft para ganar monedas.

## Diseño

### Arquitectura

Este proyecto hace uso de las arquitecturas **SPA** y **API REST** como se ha mencionado en la introducción.

Para implementar la arquitectura **SPA**, **React.js** hace uso de esta arquitectura en los proyectos creados por esta tecnología, dividiendo todas las secciones en distintas vistas y conectándolas entre si por el sistema de enrutamiento que dicho framework emplea.

En cuanto a la **API REST**, por cada entidad de la base de datos se realizan una serie de peticiones agrupadas en un archivo que se encuentra en el cliente. Cada petición apunta a una ruta distinta que tiene asociado un método que pertenece a un controlador; y en caso de que la petición reciba datos por parámetros estos serán pasados en un objeto **JSON** a los controladores que esperarán recibir dicho objeto y estos procesarán la información y devolvería los datos que se quieren obtener en



formato JSON al cliente. En el repositorio se encuentra disponible la documentación de la API.

## Entidad Relación

Como resultado en cuanto a la estructura de la base de datos se han dado las siguientes tablas, entre las cuales la más amplia hace referencia a las **Cartas de los jugadores** ya que es la que contiene más información sobre otras entidades que ninguna otra.

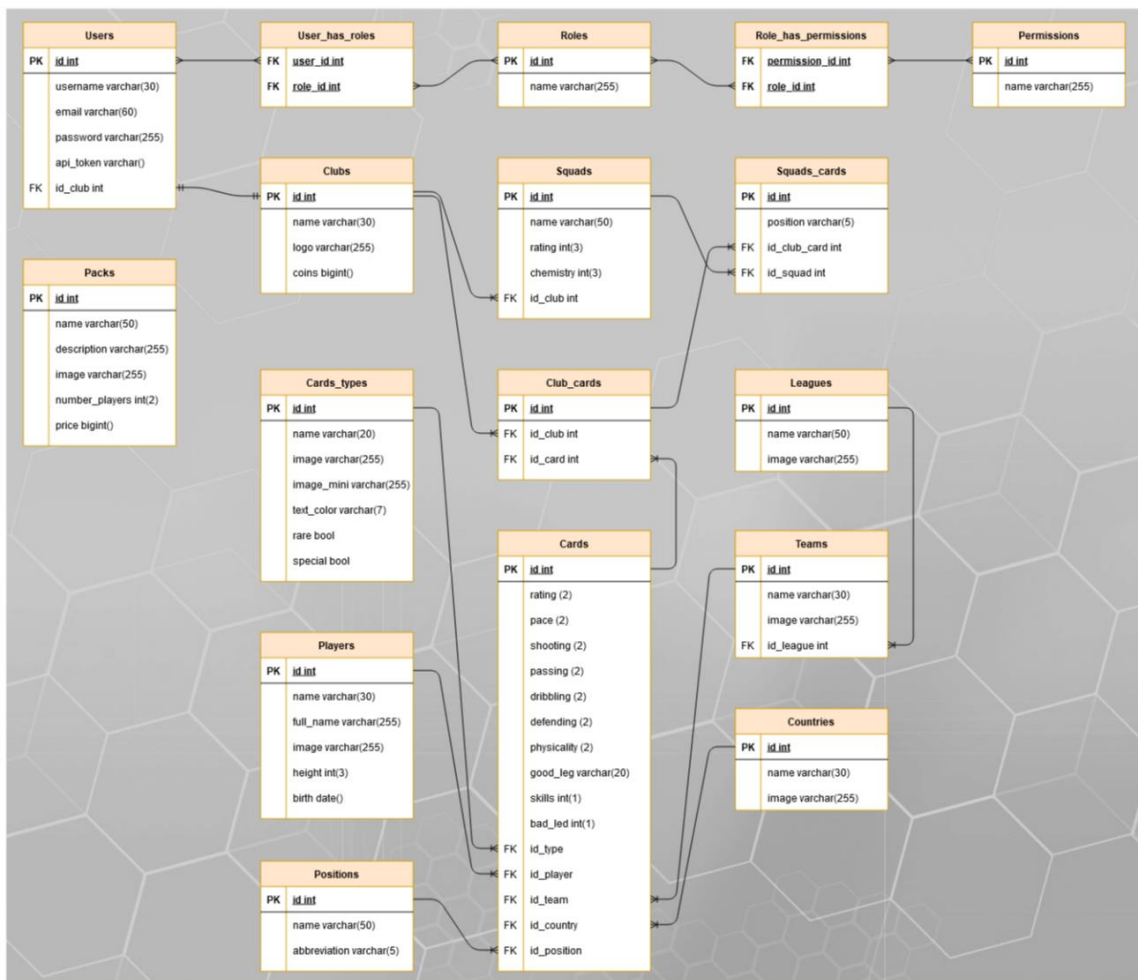


Ilustración 8: Entidad Relación

## Mapa web

La estructura web de la aplicación queda reflejada a modo de árbol de la siguiente forma:

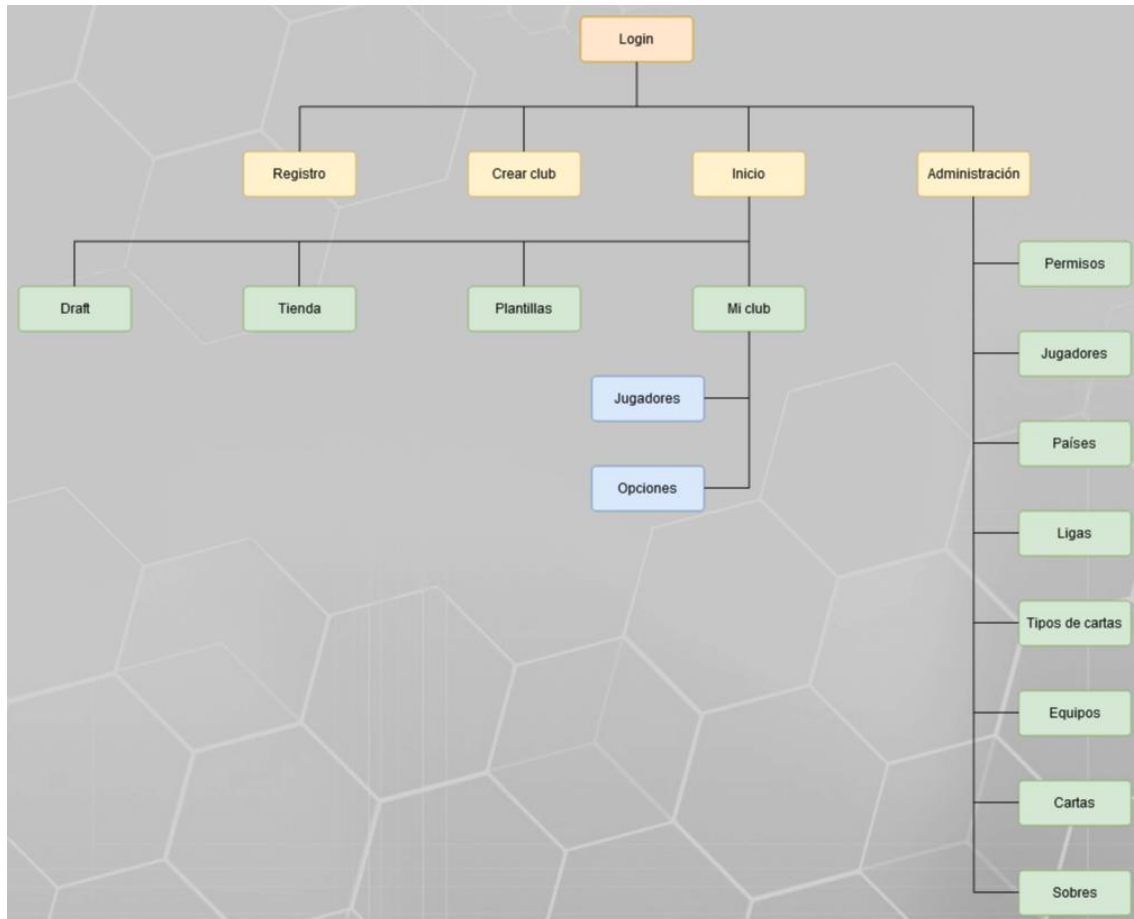


Ilustración 9: Mapa web

## Desarrollo

### Preparación

En primer lugar, se ha diseñado la **base de datos** con sus tablas, campos y relaciones. Luego, se ha creado el proyecto de React.js para el **cliente**, y el proyecto de Lumen para el **servidor**.

Se crea la base de datos, y a medida que se vaya llegando a una sección de la aplicación: se crea una migración en el servidor con las tablas pertenecientes a esa

sección y se ejecuta para que se cree la tabla en la base de datos, las rutas del servidor junto a los controladores los cuales se necesiten para dicha sección.

La aplicación se ha ido desarrollando por **versiones** y se explicará lo que se ha ido implementando en cada una.

## V 0.1

- ✓ Se han realizado las **conexiones** del cliente con el servidor, y del servidor con la base de datos.
- ✓ Se implementa el sistema de **rutas** en el cliente para poder desplazarse entre vistas.
- ✓ En el cliente se crea el **store**, donde se irán almacenando los datos llegados del servidor.
- ✓ Se crea la tabla de **usuarios**, junto con el **login** y el **registro** de los mismos. Al crear un nuevo usuario se hashearé su contraseña y se guardará así en la base de datos. Cuando el usuario inicie sesión, se creará un token y este se almacenará también en la base de datos para identificar al usuario.

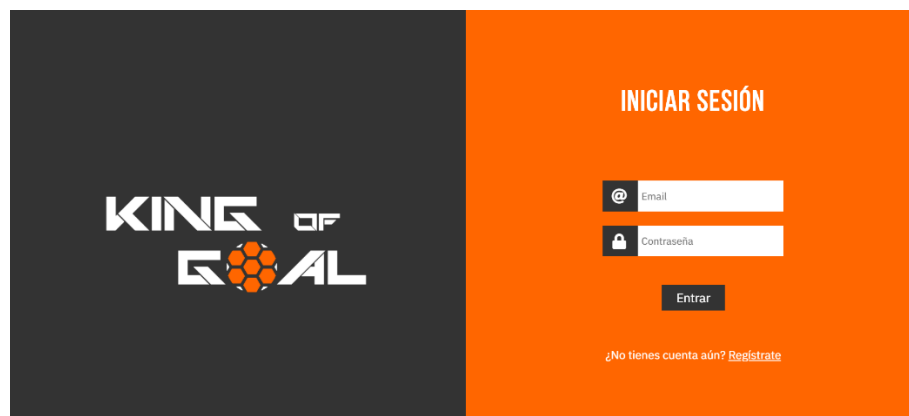


Ilustración 10: Login

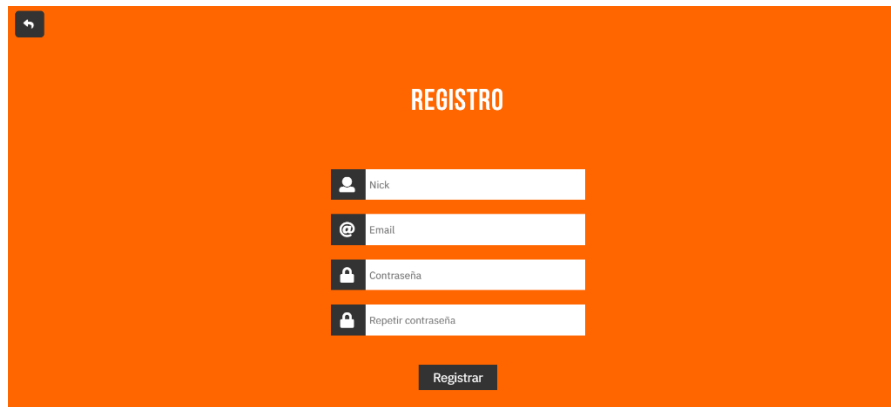
A screenshot of a registration form titled "REGISTRO" on an orange background. The form contains four input fields: "Nick" with a person icon, "Email" with an @ icon, "Contraseña" with a lock icon, and "Repetir contraseña" with a lock icon. Below the fields is a dark button labeled "Registrar". A small back arrow icon is in the top left corner of the form area.

Ilustración 11: Registro

- ✓ Se **validan** los formularios, tanto en el cliente como en el servidor.
- ✓ Se añade un **cuadro de notificación** los errores ocurridos o las acciones realizadas con éxito.

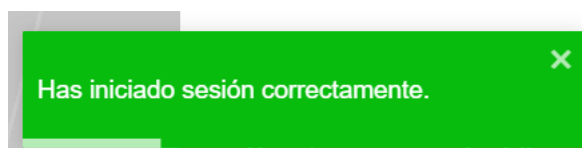


Ilustración 12: Cuadro de notificación

- ✓ **Logout** para cerrar sesión. Cuando el usuario cierra sesión, el token se elimina de la base de datos.

## V 0.2

- ✓ **Menú principal.**



Ilustración 13: Menú principal

✓ Menú lateral.



Ilustración 14: Menú lateral

- ✓ Al sistema de rutas se le añaden las **subrutas**.
- ✓ Sección de la **administración**.



Ilustración 15: Administración

- ✓ Se han implementado **middlewares** en el servidor.
- ✓ Sistema de **roles** y **permisos** mediante un paquete de Laravel que crea las tablas de roles y permisos en la base de datos. En el servidor se crea un **seeder** que crea el usuario de administrador junto con el rol *super-admin*, que se le asignará a este usuario.
- ✓ Manejo de **permisos** en la administración. Cada permiso que se cree, se le asignará al rol *super-admin*.

### V 0.3

Esta versión se centra en la administración, incluyendo las secciones para el manejo de los siguientes datos, creando así sus respectivas tablas y relaciones en la base de datos:

- Jugadores.
- Países.
- Ligas.
- Equipos.
- Tipos de cartas.
- Cartas.

A esta versión también se le añade lo siguiente:

- ✓ Creación de la tabla de **posiciones**.
- ✓ Cartas dinámicas de los jugadores.



Ilustración 16: Carta

## V 0.4

- ✓ Creación de la tabla de los **clubes** de los usuarios.
- ✓ Vista de la creación del **club** del usuario.

**CREAR CLUB**

**H** Introduce el nombre del club

Selecciona el escudo

Selecciona una liga

Confirmar

Ilustración 17: Crear club

- ✓ Una vez el usuario inicia sesión, se realiza la petición al servidor de una **primera carga** para recibir todos los datos de la aplicación.
- ✓ Se añade la sección **Mi club** junto a sus subsecciones **Jugadores** y **Opciones**.

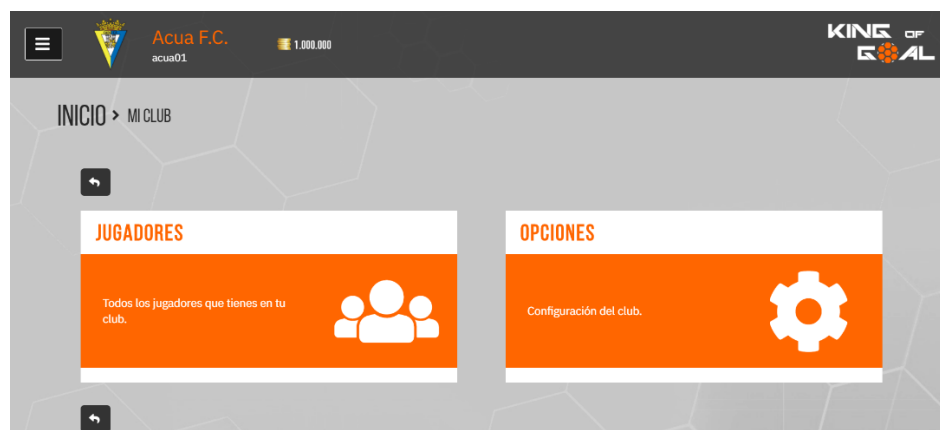


Ilustración 18: Mi club

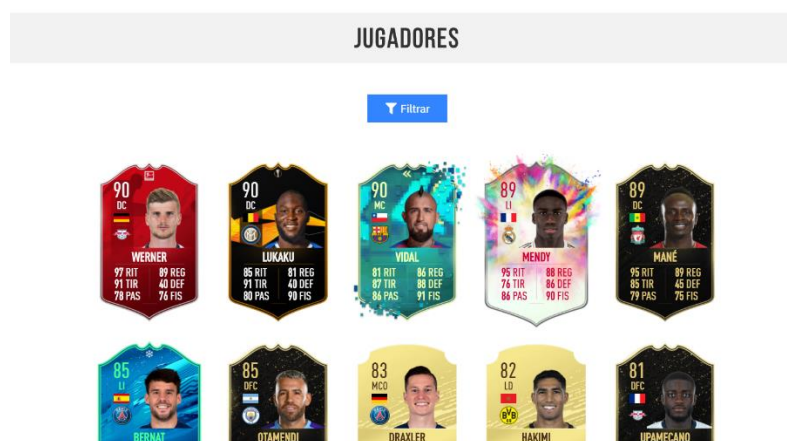


Ilustración 19: Jugadores

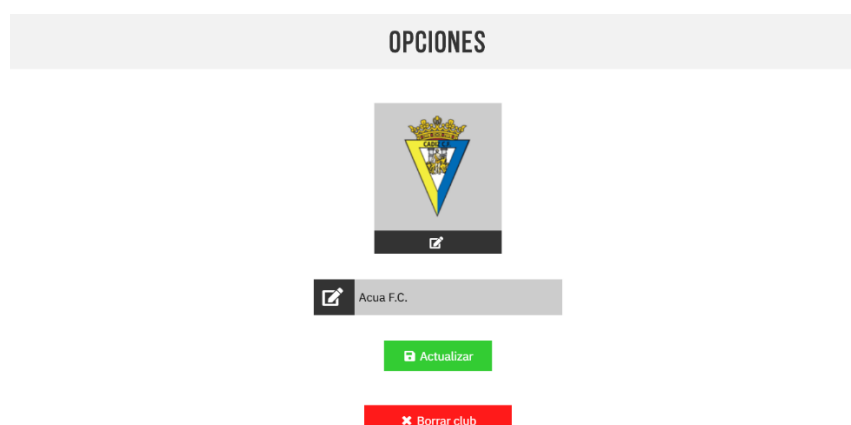


Ilustración 20: Opciones



- ✓ Se crea el **filtro de jugadores**.



Ilustración 21: Filtrar



Ilustración 22: Ordenar

- ✓ Se añade la funcionalidad de **vender** cartas para ganar monedas.

## V 0.5

- ✓ Se añaden las **migas de pan** y el **botón de volver atrás** para dar más facilidad a la navegación.



Ilustración 23: Migas de pan

- ✓ Se crea la sección de **Sobres** en la administración, al igual que su respectiva tabla en la base de datos.
- ✓ Se añade la sección de **Tienda** en el menú principal con el listado de los sobres disponibles.



Ilustración 24: Tienda

- ✓ Se implementa la **animación** una vez se vaya a abrir el sobre con los jugadores que salgan aleatoriamente.



Ilustración 25: Animación

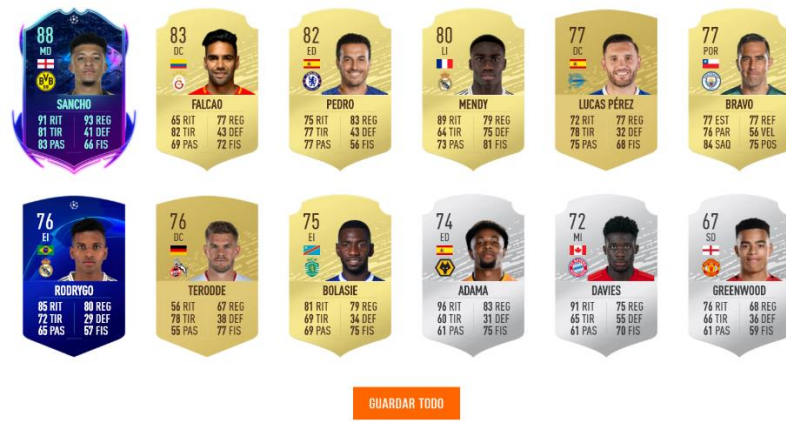


Ilustración 26: Jugadores

V 0.6

- ✓ Se crea la sección **Plantillas** en el menú principal, con el listado de las plantillas y sus funcionalidades de **crear**, **renombrar** y **borrar**.

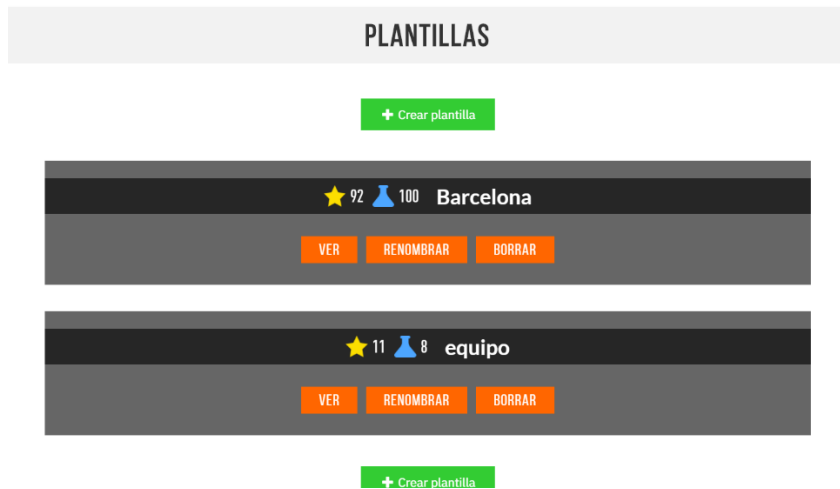


Ilustración 27: Plantillas

- ✓ Se implementa la **vista de la plantilla** con sus cartas con sus funcionalidades de incluir cartas en ella, quitarlas o intercambiarlas con otras del club o entre posiciones.



Ilustración 28: Vista de la plantilla, puntuaciones

Para montar el diseño de la plantilla, se ha utilizado **canvas** para pintar las líneas de colores y luego las cartas se van colocando encima de ellas mediante coordenadas.

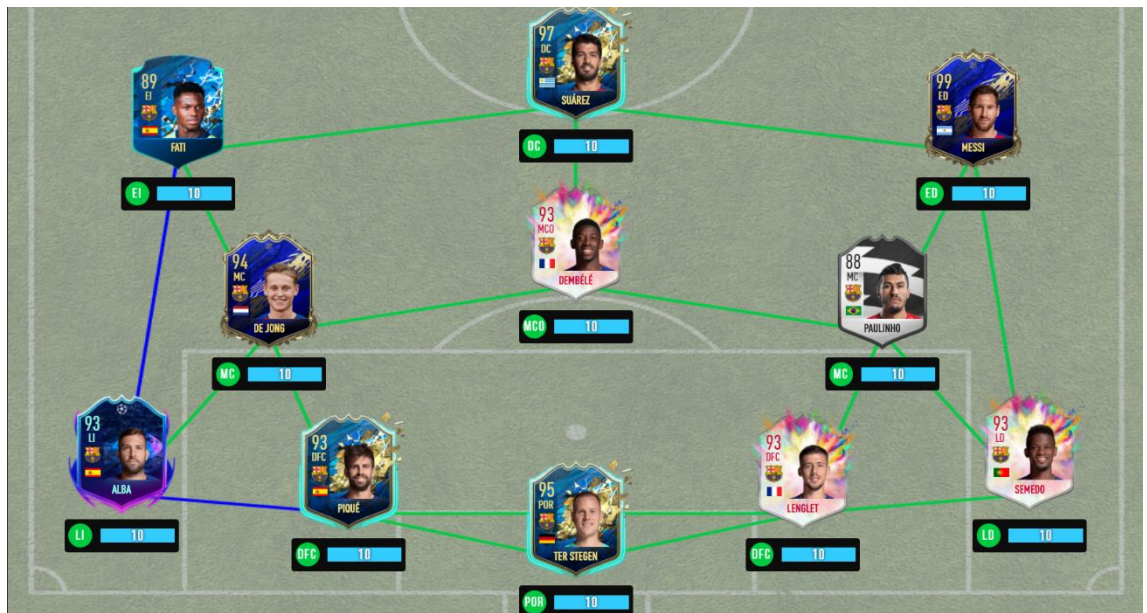


Ilustración 29: Vista de la plantilla, titulares

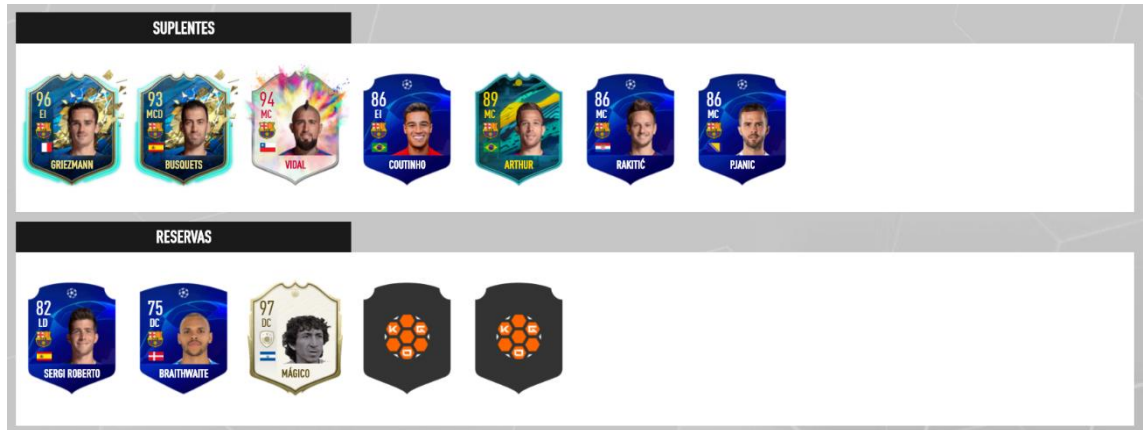


Ilustración 30: Vista de la plantilla, banquillo

## V 0.7

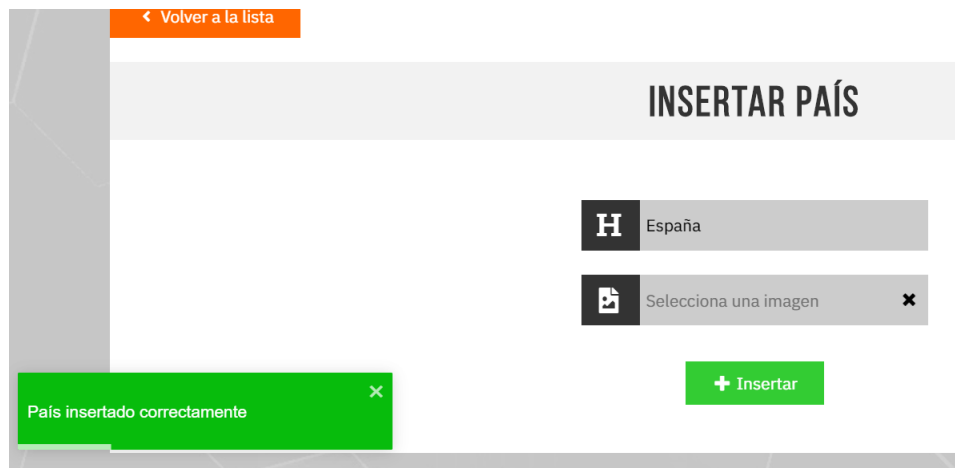
- ✓ Se crea la sección **Draft** con el minijuego para ganar monedas. Para ello se ha reutilizado el diseño de la plantilla anterior, modificando que al pinchar en una posición en vez de que salgan las cartas de nuestro club, salgan 5 cartas aleatorias; y luego que al hacer click en *finalizar* nos lleve a la vista de recoger la recompensa.

RESULTADO	
★	93
🏆	100
TOTAL	193
💰	9.000
RECoger RECOMPENSA	

Ilustración 31: Recompensa

## Pruebas

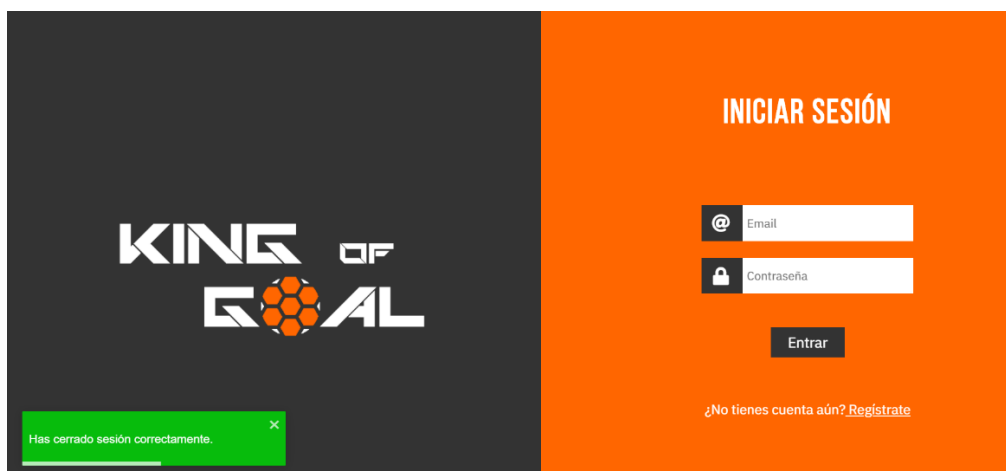
En cuanto a la realización de **pruebas**, se ha comprobado que todas las **validaciones** implementadas en los formularios validan correctamente los datos, insertando en ellos múltiples tipos de valores y patrones.



The screenshot shows a web form titled "INSERTAR PAÍS". At the top left, there is a link "< Volver a la lista". The form contains two input fields: the first is labeled "H" and contains the text "España"; the second is labeled with an image icon and contains the placeholder text "Selecciona una imagen" with a close button "x". Below the first field, a green notification box displays "País insertado correctamente" with a close button "x". At the bottom right, there is a green button labeled "+ Insertar".

Ilustración 32: Probando validaciones

También se ha comprobado vista por vista que todas las **funcionalidades** programadas en la aplicación funcionan correctamente.



The screenshot shows a login page with a dark grey left sidebar and an orange right section. The sidebar features the "KING OF GOAL" logo and a green notification box at the bottom that says "Has cerrado sesión correctamente." with a close button "x". The orange section is titled "INICIAR SESIÓN" and contains two input fields: "Email" (with an @ icon) and "Contraseña" (with a lock icon). Below these fields is a dark grey button labeled "Entrar". At the bottom of the orange section, there is a link: "¿No tienes cuenta aún? [Regístrate](#)".

Ilustración 33: Probando funcionalidades

Para comprobar que las **peticiones** al servidor se realizan con éxito, se ha hecho uso de la herramienta **Postman**, que nos permite crear peticiones y probar los resultados.



Ilustración 34: Postman

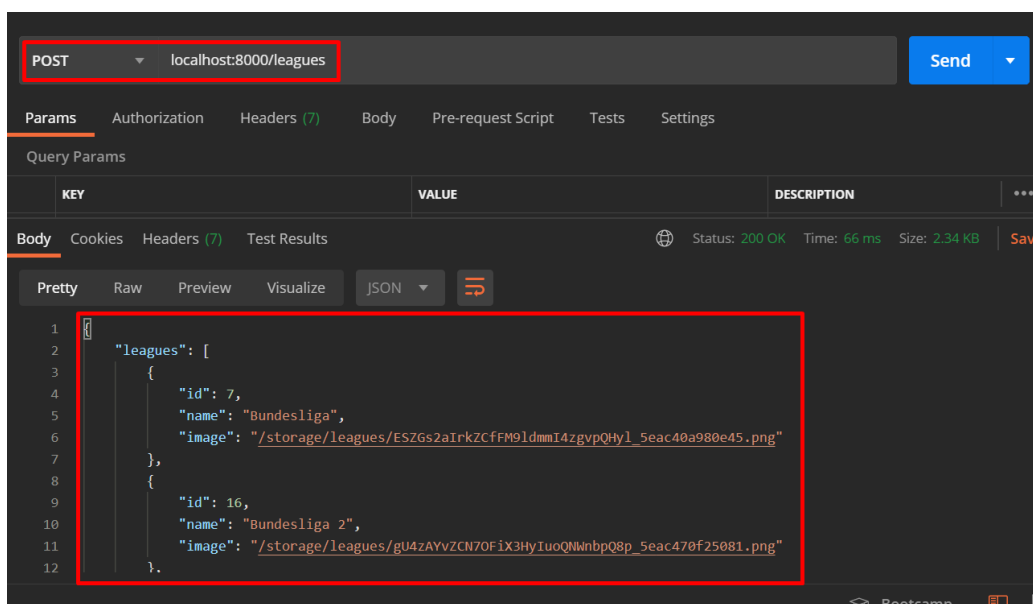


Ilustración 35: Probando peticiones con Postman

## Instalación y despliegue

En cuanto al despliegue de la aplicación se ha utilizado **Docker**, que es una tecnología que nos permite automatizar la implementación de aplicaciones como contenedores portátiles y autosuficientes que se pueden ejecutar tanto en la nube como localmente.

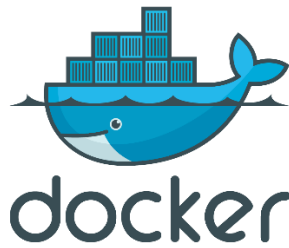


Ilustración 36: Docker

Para desplegar nuestra aplicación se ha hecho lo siguiente:

1. Una vez terminado el desarrollo del cliente y del servidor, se compila el cliente con el comando '***npm run build***', y nos generará una carpeta llamada **build** que contendrá todo el código del cliente compilado y los recursos guardados en la carpeta **public** de nuestro proyecto cliente.

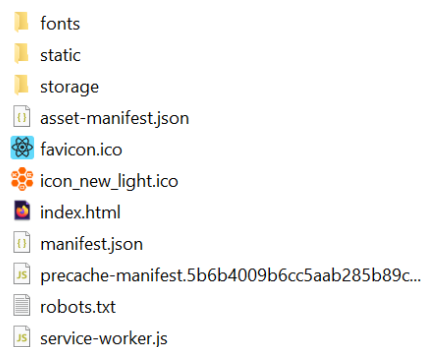
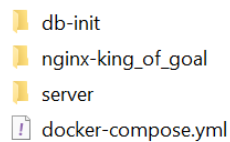


Ilustración 37: Estructura de la carpeta build

2. Copiamos el código compilado generado en el archivo **index.html**, nos dirigimos al directorio **server** y lo pegamos en **resources > views > index.blade.php**.



3. En **server**, dentro de **public** pegamos los demás ficheros y directorios generados por la compilación realizada anteriormente.
4. Ahora si ya tendríamos listo el servidor para realizar el despliegue. A continuación, en mismo nivel del directorio **server** ambos ficheros añadimos los siguientes archivos que estarán disponibles en **despliegue.zip** del repositorio.

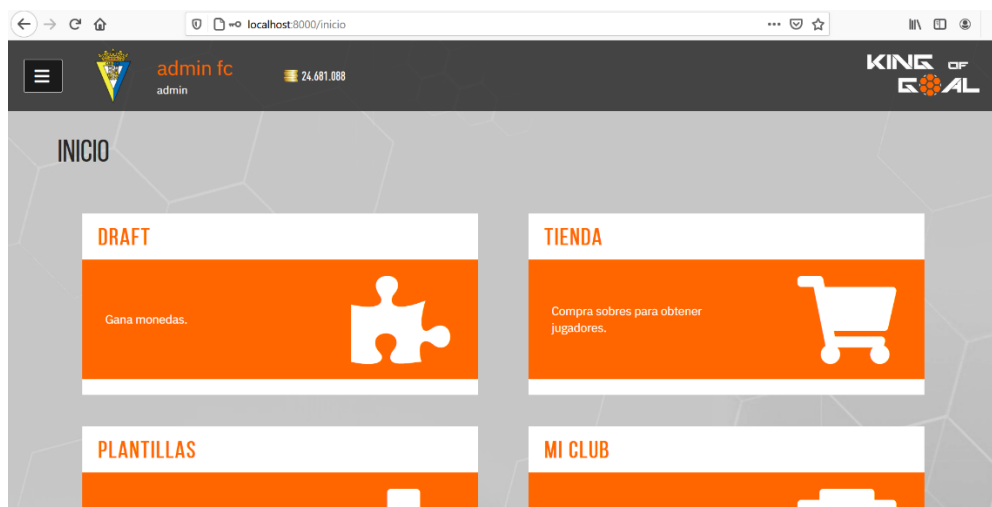
*Ilustración 38: Estructura de despliegue.zip*

5. Iniciamos **Docker** y dentro del directorio donde tenemos la estructura de ficheros anterior, ejecutamos el comando '**docker-compose up**' para así descargar las imágenes que se emplearán para construir los contenedores en los cuales desplegaremos nuestra aplicación. Tendremos dos contenedores, uno para la base de datos y otro para el servidor.

```
PS C:\Users\aleap\development> docker-compose up
Starting development_server_1 ...
Starting development_server_1 ... done
```

*Ilustración 39: Ejecutando docker-compose up*

6. Finalmente nos dirigimos al navegador y comprobamos en '**localhost:8000**' que nuestra aplicación se ha desplegado con éxito.

*Ilustración 40: Resultado del despliegue*

## Conclusiones

Una vez visto todo y llegado al final del proyecto, se han cumplido todos los objetivos pretendidos inicialmente. El desarrollo de la aplicación ha sido extenso e incluso a veces muy complejo debido a las funcionalidades más difíciles de implementar como las secciones **Plantillas** y **Draft**.

Mediante el uso de las tecnologías empleadas **React.js** y **Lumen** he aprendido a como un framework estructura un proyecto y todas las facilidades que pueden llegar a emplearnos; y como un cliente se conecta a un servidor por separados ambos para realizar peticiones y obtener datos, así como el concepto de **API REST**.

Como posibles futuras mejoras a la aplicación, podrían añadirse nuevas funcionalidades como las siguientes:

- ✚ El usuario al crear una nueva plantilla pueda elegir y cambiar la formación.
- ✚ Cuando se termine de completar un equipo haciendo un Draft que con el propio equipo montado se pueda simular partidos y dependiendo de las valoraciones de la plantilla tenga más posibilidad de ganarlos o no, y conseguir así más monedas.
- ✚ Como recompensas de un Draft poder obtener sobres.

En resumen, cumpliendo todos los objetivos citados anteriormente se ha logrado una aplicación bastante compleja y completa.

## Índice de imágenes

Ilustración 1: Introducción .....	- 2 -
Ilustración 2: React.js .....	- 3 -
Ilustración 3: Lumen.....	- 3 -
Ilustración 4: MySQL .....	- 3 -
Ilustración 5: Cliente .....	- 4 -
Ilustración 6: Servidor .....	- 5 -
Ilustración 7: API REST.....	- 6 -
Ilustración 8: Entidad Relación .....	- 8 -
Ilustración 9: Mapa web.....	- 9 -
Ilustración 10: Login .....	- 10 -
Ilustración 11: Registro .....	- 11 -
Ilustración 12: Cuadro de notificación .....	- 11 -
Ilustración 13: Menú principal .....	- 12 -
Ilustración 14: Menú lateral .....	- 12 -
Ilustración 15: Administración .....	- 13 -
Ilustración 16: Carta .....	- 14 -
Ilustración 17: Crear club .....	- 14 -
Ilustración 18: Mi club.....	- 15 -
Ilustración 19: Jugadores.....	- 15 -
Ilustración 20: Opciones.....	- 15 -
Ilustración 21: Filtrar .....	- 16 -
Ilustración 22: Ordenar .....	- 16 -
Ilustración 23: Migas de pan .....	- 16 -
Ilustración 24: Tienda.....	- 17 -
Ilustración 25: Animación.....	- 17 -
Ilustración 26: Jugadores.....	- 18 -
Ilustración 27: Plantillas .....	- 18 -
Ilustración 28: Vista de la plantilla, puntuaciones .....	- 19 -
Ilustración 29: Vista de la plantilla, titulares .....	- 19 -
Ilustración 30: Vista de la plantilla, banquillo .....	- 20 -
Ilustración 31: Recompensa .....	- 20 -
Ilustración 32: Probando validaciones .....	- 21 -
Ilustración 33: Probando funcionalidades .....	- 21 -
Ilustración 34: Postman.....	- 22 -
Ilustración 35: Probando peticiones con Postman .....	- 22 -
Ilustración 36: Docker .....	- 23 -
Ilustración 37: Estructura de la carpeta build .....	- 23 -
Ilustración 38: Estructura de despliegue.zip .....	- 24 -
Ilustración 39: Ejecutando docker-compose up.....	- 24 -
Ilustración 40: Resultado del despliegue .....	- 24 -

## Bibliografía

**React.js:** <https://es.wikipedia.org/wiki/React>

**Lumen:** [https://en.wikipedia.org/wiki/Lumen\\_\(software\)](https://en.wikipedia.org/wiki/Lumen_(software))

**MySQL:** <https://www.e-sepestudio.com/noticias/que-es-mysql>

**Documentación React.js:** <https://es.reactjs.org/>

**Documentación Laravel:** <https://laravel.com/docs/8.x>

**Documentación Lumen:** <https://lumen.laravel.com/docs/8.x>