UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

Graduado en Ingeniería del Software

Herramienta para cuantificar el rendimiento de futbolistas

A tool to quantify the performance of football players

Realizado por
José Luis Díaz Martín

Tutorizado por
Gabriel Jesús Luque Polo
Francisco Javier Ferrer Urbano

Departamento
Lenguaje y Ciencias de la Computación

MÁLAGA, junio de 2021

UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
Graduado en Ingeniería del Software

Herramienta para cuantificar el rendimiento de futbolistas

A tool to quantify the performance of football players

Realizado por
José Luis Díaz Martín

Tutorizado por
Gabriel Jesús Luque Polo
Francisco Javier Ferrer Urbano

Departamento
Lenguaje y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA
MÁLAGA, junio de 2021

Fecha defensa: julio de 2021

Antes de comenzar quería dedicar unas palabras de agradecimiento a todas aquellas personas que me han ayudado y apoyado durante todo el camino para llegar hasta aquí.

En primer lugar a mis padres, que desde siempre han luchado con mucho esfuerzo para que yo pudiera realizar este camino, brindándome una gran educación junto con mucho cariño.

A mi hermana, que siempre ha creído en mí desde pequeño y me inculcó que podía lograr lo que me propusiera.

A mi abuela, que se ha preocupado siempre por mí, rezando a todo santo que se le pudiera realizar una oración y alegrándose por cada pequeño logro obtenido durante el camino.

A mi pareja, que ha sido un pilar fundamental desde que llegó a mi vida y me ha dado alas para poder alcanzar todas las metas que me proponga.

A mis dos tutores, que me han ayudado a realizar este proyecto de la mejor forma posible.

A mis profesores de instituto, que creyeron en mí y decidieron concederme la matrícula de honor, la cual me ha permitido poder emprender esta carrera universitaria.

A mis compañeros y amigos, por toda la ayuda, apoyo y buenos momentos que hemos vivido a lo largo de estos años.

Gracias a todos, lo hemos conseguido.

Resumen

Tradicionalmente el rendimiento de un jugador se ha medido por parámetros muy básicos como son los goles marcados, en el caso de los delanteros, o de paradas realizadas en el caso de porteros. Estos indicadores tan básicos no muestran de forma adecuada el rendimiento real de los jugadores, ya que están influidos por el rendimiento del resto del equipo que no se están midiendo y combinando adecuadamente. Por ejemplo, un portero mediocre puede sobresalir si tiene una excelente defensa, o un delantero puede destacar si tiene a grandes pasadores en el equipo.

En la actualidad se recopila gran cantidad de información sobre las prestaciones de un futbolista durante los partidos, entrenamientos y otras actividades que realiza que pueden servir para ofrecer una mejor visión sobre su rendimiento.

En este trabajo de fin de grado se propone usar técnicas que estudien el rendimiento con un análisis avanzado que tenga en cuenta la interrelación de los datos de rendimiento y se crearán métricas avanzadas para poder cuantificarlo.

Palabras clave: **rendimiento, datos, jugador, análisis**

Abstract

Traditionally, a player's performance has been measured by very basic parameters such as goals scored, in the case of forwards players, or "saves" made by goalkeepers. These very basic indicators do not adequately show the actual performance of the players, as they are influenced by the execution of the game by the rest of the team. These measurements aren't combined properly. For example, a mediocre goalkeeper can excel or highlight if he has an excellent defending player; or maybe, a fordward player can excel if he has a great passer in the field.

Currently, a large amount of information has been collected about the performance of a footballer during matches, training sessions and other activities that he performs that can serve to offer a better vision of his standing out.

In this final degree project, it is proposed to use techniques that study performance with an advanced analysis that takes into account the interrelation of performance data; also, advanced metrics will be created to quantify the quality of the game and the players.

Keywords: perfomance, data, player, analysis

Índice

1 Introducción

En este primer capítulo se planteará el ámbito del trabajo de fin de grado, los objetivos que se plantean cubrir y una breve descripción de la organización y contenido de este documento.

1.1. Motivación

El fútbol es considerado uno de los deportes más populares en la actualidad, no solo a nivel deportivo, como un simple juego, sino también a nivel social, ya que mueve a grandes grupos sociales afines a un club o incluso a una nación.

Hoy en día, con los avances tecnológicos que se han dado existen multitud de recogida de datos durante la realización de un partido o incluso durante la práctica de un entrenamiento. Lo que abre un abanico inmenso de posibilidades para realizar un análisis tanto de los propios jugadores como del juego colectivo del equipo.

Así pues, los clubes siempre buscan mejorar en todos los aspectos posibles para conseguir mejores resultados, ya que en la actualidad el nivel económico que se mueve en la élite del fútbol es muy grande. Para el cuerpo técnico de un club es muy interesante conocer si los últimos partidos de un jugador han sido buenos o malos en comparación a la media de sus actuaciones a lo largo de esa misma temporada, de una diferente o de toda su carrera.

1.2. Objetivos

El objetivo principal de este trabajo fin de grado será el poder elaborar un informe detallado y cuantitativo de un futbolista concreto en función de toda la información recogida de

los datos de este en los partidos. Con el fin de evaluar su rendimiento en un partido específico con la media generada por sus datos históricos o los datos de una temporada concreta, a modo de una comparación contra sí mismo, mostrando la información de manera detallada y visual.

Este objetivo principal no es nada sencillo y para conseguirlo planteamos dos grandes actividades:

- Por un lado, la del tratamiento de datos lo implica que examinar las fuentes de datos, como por ejemplo OptaSports [1] que existen sobre el rendimiento de los jugadores, examinar qué técnicas estadísticas y de otros dominios se pueden utilizar para obtener conocimiento de esos datos y la definición de métricas cuantificables que nos puedan dar una visión precisa del jugador.
- Por otro lado, necesitamos crear un sistema de información para almacenar de forma eficiente toda esa información y crear un sistema web con una interfaz intuitiva y visual que nos permita interactuar con nuestro sistema.

1.3. Metodología de trabajo

La metodología de trabajo que se ha llevado a cabo durante todo el proyecto ha sido Scrum, la cual consiste en realizar springs de trabajo de una semana en este caso, donde siempre se adapta el trabajo a las necesidades del cliente, el proyecto en este desarrollo. El esquema de la reuniones fue el siguiente:

- Exponer las tareas propuestas en la reunión anterior para analizar como ha sido el trabajo.
- Revisar los problemas e incidencias que hayan podido surgir durante el desarrollo de las tareas.
- Planificar y plantear las nuevas tareas para la próxima reunión.

Con esta filosofía de trabajo se comenzó el trabajo, tras haber discutido y diseñado una planificación de fases de trabajo a realizar durante estos meses:

1. Recopilación de una fuente de datos

2. Tratamiento de datos de dicha fuente

3. Desarrollo del sistema de información

4. Desarrollo de la interfaz web

1.4. Estructura del documento

La memoria está dividida en 6 capítulos, dos anexos y las referencias bibliográficas utilizadas a lo largo de este documento:

1. **Introducción**: se explica la motivación y los objetivos que persiguen el trabajo, así como también la estructura del documento.

2. **Tecnologías**: se detalla todas las tecnologías utilizadas durante el desarrollo del proyecto.

3. **Tratamiento de datos**: se expone como se ha llevado a cabo el tratamiento de la fuente de datos.

4. **Desarrollo del sistema de información**: se relata como se ha creado la base de datos.

5. **Desarrollo de la interfaz web**: se cuenta como se realizó el diseño y las funcionalidades implementadas dentro de la interfaz.

6. **Conclusiones y líneas futuras**: se expone las conclusiones y posibles líneas futuras que tienen este proyecto.

2

Tecnologías

Tras haber explicado la motivación y objetivos de este trabajo de fin de grado, en este capítulo se van a tratar las diferentes tecnologías software empleadas para llevar a cabo el desarrollo del proyecto.

2.1. Tecnologías de backend

2.1.1. MySQL

MySQL[2] es un sistema de gestión de bases de datos relacionales de código abierto, con un modelo cliente-servidor. Utilizado para crear y administrar bases de datos basadas en un modelo relacional.

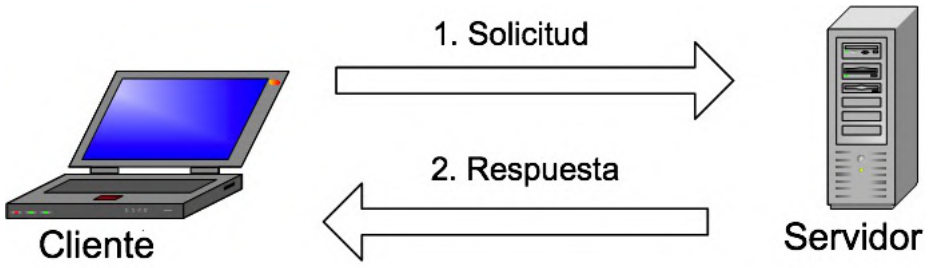

Figura 1: Funcionamiento de MySQL

La imagen explica la estructura básica cliente-servidor, uno o más dispositivos(clientes) se conectan a un servidor a través de una red específica. Cada cliente puede realizar una petición desde la interfaz de usuario y el servidor producirá la respuesta deseada, siempre que las instrucciones escritas sean correctas.

Se ha decidido utilizar MySQL frente a otras opciones ya que es flexible y fácil de usar a través del programa MySQL Workbench. Ofreciendo un alto rendimiento, un estándar dentro de la industria con numerosos recursos disponibles realizados por desarrolladores cualificados y un sistema seguro gracias a los privilegios de acceso y la administración de cuentas de usuario.

2.1.2. Python

Python[3] es un lenguaje de programación interpretado, multiparadigma y multiplataforma usado, principalmente, en Big Data, AI (Inteligencia Artificial), Data Science, frameworks de pruebas y desarrollo web. Esto lo convierte en un lenguaje de propósito general de gran nivel debido a su extensa biblioteca, cuya colección ofrece una amplia gama de instalaciones.

Gracias a esa gran biblioteca de frameworks y librerías decidimos utilizarlo, ya que se iba a basar toda la lógica del backend con Flask y el manejo de la base de datos con SQLAlchemy que explicaremos más adelante.

2.1.3. Flask

Es un framework para aplicaciones web en Python que permite desarrollar el *back-end* sin una arquitectura (MVC) concreta. Es ligero, por lo que carece de abstracción de base de datos y ciertos elementos habituales, aunque permite añadirlos. Admite el uso de cookies y hace las de servidor web para desarrollo y depuración.

Podemos añadir nuevas funcionalidades según se vaya requiriendo a lo largo del desarrollo, lo cual es muy interesante para solo instalar las cosas necesarias. En este caso haremos uso de SQLAlchemy[5] para conectar el servidor con la base de datos.

Además vamos a hacer uso de llamadas HTTP del tipo *GET* y del tipo *POST*, donde las llamadas del primer tipo las utilizaremos para obtener recursos de la interfaz web mientras que por el contrario el segundo tipo se usará para crear nuevos datos y enviar dichos recursos a la interfaz web.

2.1.4. SQLAlchemy

SQLAlchemy es un mapeador de objetos relacionales más conocido como ORM, es decir, una librería que los desarrolladores utilizan para crear bases de datos y manipular sus datos sin la necesidad de usar SQL. Esto ayuda mucho a requerir de un menor esfuerzo en muchos casos y simplificar la manipulación de datos.

2.1.5. Matplotlib

Una de las bibliotecas utilizadas de Python ha sido Matplotlib [9], para la generación de gráficos a partir de datos contenidos en listas o arrays en el lenguaje de programación Python, disponiendo de un sitio web con una gran guía de usuario, la cual explica todas las funcionalidades que ofrece.

Podemos destacar los mapas de calor, estos se pueden representar como una cuadrícula de valores dada una matriz de valores que defina el número de filas y columnas. Esto resulta muy interesante para poder mostrar datos como donde se localizan los eventos del jugador a lo largo de todo el campo o los disparos a portería.

2.1.6. Apache

Como servidor HTTP usaré Apache[6], el cual es un servidor web de código abierto, multiplataforma, modular, extensible y muy popular, lo que significa que será sencillo encontrar ayuda en internet.

Otra de las grandes ventajas es la posibilidad de previsualizar y probar código mientras se está desarrollando, pudiendo compartir archivos que tenemos en local hacia Internet. Al trabajar en local podemos controlar mucho mejor la seguridad de nuestro servidor y arreglar posibles fallos.

2.1.7. XAMPP

Para gestionar toda la base de datos MySQL y el servidor Apache, he decidido utilizar XAMPP[7], una herramienta ligera para instalar y utilizar. La cual permite configurar los puertos del ordenador y gestionar la base de datos a un solo click.

XAMPP proporciona una buena capa de seguridad para tener protegido todo nuestro desarrollo y poder testearlo sin preocupaciones de fallas de seguridad. Es multiplataforma y ofrece una interfaz web sencilla para trabajar.

2.2. Tecnologías de frontend

2.2.1. HTML

Para implementar el contenido de las páginas web se ha utilizado el lenguaje de HTML[10]. Básicamente se trata de un conjunto de etiquetas que sirve para definir el texto y otros elementos que compondrán una página web, como imágenes o listas.

Se desde respetar en todo momento las especificaciones propias del lenguaje, lo cual puede presentar en algunas situaciones problemas para realizar un tipo de diseño específico en la página.

2.2.2. Bootstrap

Para dar estilo a todo el contenido HTML se ha utilizado Bootstrap[11], que es una biblioteca multiplataforma o un kit de herramientas de código abierto para desarrollos web responsive con HTML, CSS[12] y Javascript[13].

Incluye diferentes componentes desde ventanas modales, menús, cuadros, botones, formularios... Este no solo nos permite darle forma al sitio web, sino que también es compatible con todo tipo de navegadores y dispositivos móviles.

2.2.3. JavaScript

Para conseguir algunas animaciones dentro de la página web se ha elegido JavaScript, que es un lenguaje de programación o de secuencias de comandos que te permite implementar funciones complejas en páginas web. Muestra oportunas actualizaciones de contenido, mapas interactivos, animación de Gráficos 2D/3D, entre otras funciones.

Esta tecnología está presente hoy en día en todos los navegadores modernos que interpretan directamente el código JavaScript, utilizando una sintaxis similar a la programación de objetos.

2.3. Tecnologías de desarrollo y auxiliares

2.3.1. Visual Studio Code

Como entorno de desarrollo se eligió Visual Studio Code[14], un editor de código fuente disponible en numerosos sistemas operativos. Incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código.

También es personalizable, por lo que los usuarios pueden cambiar el tema del editor, los atajos de teclado y las preferencias. Es gratuito y de código abierto, con el cual es posible desarrollar código en multitud de tipo de lenguajes, ya que cuenta con una gran variedad de extensiones disponibles.

2.3.2. Git

Esta aplicación requería de algún sistema de control, que nos permitiera tener almacenada a la misma, en caso de tener algún problema con los equipos o bien una manera de restituir la aplicación a un punto anterior si fuera necesario.

Para ello, se eligió por Git[15], una herramienta con la cual ya había trabajado con anterioridad al proyecto y que permite guardar una copia del proyecto, restaurar una versión

anterior si fuera necesario o desarrollar funcionalidades en paralelo sin que haya conflictos, entre otras.
Concretamente se ha utilizado la plataforma Github[16], siendo este el repositorio:
https://github.com/Joslu12/StatTFG

2.3.3. Microsoft Teams

La herramienta de Microsoft Teams[17] se ha utilizado para organizar las reuniones semanales del proyecto y llevar un seguimiento de las mismas. Permite crear grupos donde se puede llevar un registro de todas las conversaciones realizadas, así como de las llamadas y el propio chat.

Una función que ha sido muy beneficiosa fue la posibilidad de poder compartir pantalla, para la resolución de los problemas e incidencias que han aparecido a lo largo de todo el desarrollo del proyecto.

2.3.4. LaTeX

Para la realización de la memoria del proyecto se ha decidido utilizar LaTeX[18], que es un sistema de composición de textos, orientado especialmente a la creación de libros, documentos científicos y técnicos que contengan fórmulas matemáticas. Se ha desarrollado a través del programa Overleaf[18], que ofrece la comodidad de ser online y poder acceder desde diferentes equipos.

La razón de su elección se basa en las facilidades que ofrece a través de las instrucciones para añadir imágenes, fórmulas, tablas u otro tipos de contenido, ajustando los detalles de formato de manera independiente al contenido.

2.3.5. Quickmockup

Para realizar los bocetos de la interfaz web se ha utilizado la herramienta online llamada Quickmockup[21], permitiendo dar forma a los bocetos sin la necesidad de descargar ningún software. Ha sido muy sencilla e intuitiva de utilizar, además de ser gratuita.

Permitiendo descargar los diseños realizados y poder volverlos a cargar en cualquier otro momento para editarlos, siendo esto un punto positivo además de cómodo y disponer de elementos por defecto que se pueden arrastrar y modificar posteriormente.

2.3.6. Lucidchart

Para realizar el esquema de navegación de la interfaz web y crear los diferentes flujos de navegación que se producen, he decidido utilizar Lucidchart[22], que es una aplicación de diagramación inteligente.

Siendo un entorno de trabajo online que permite realizar multitud de tipo de diagramas, además de ofrecer recursos de formación en formato PDF o video para aprender a utilizar su herramienta, la cual es totalmente gratuita.

3

Tratamiento de Datos

3.1. Fuente de datos

La fuente de datos es suministrada por StatBomb[20], una empresa proveedora de datos avanzados de fútbol, líder en el sector y que colabora con multitud de clubes, competiciones, medios de comunicación y portales de apuesta. Dicha fuente se trata de una demo gratuita que incluye una cantidad suficiente de datos para cubrir el propósito de este proyecto.

Figura 2: Logo de StatsBomb

El objeto de estudio principal para analizar sus datos será todos los partidos del Futbol Club Barcelona en la Liga Española desde la temporada 2003/2004 hasta la temporada 2019/2020, ya que en dicha muestra de datos solo tenemos los partidos que juega el equipo blaugrana. Se ha decidido no tener en cuenta las estadísticas del resto de equipos, al disponer únicamente de sus enfrentamientos contra el Barcelona a lo largo de los años, lo que no ofrece unas estadísticas interesantes para analizar.

3.2. Tratamiento de los datos

El formato de los datos son listas de JSON, por lo que tuve que leer cada archivo y almacenarlo en una lista, que posteriormente sería iterada para insertarse dentro de la base de datos. Dividiremos todos los datos en:

- **Alineaciones**
- **Partidos**
- **Eventos**

Pero para poder realizar inserciones dentro de la base de datos, he tenido que seguir un orden, ya que muchas tablas tienen dependencias de otras como la tabla Country que aparece en la mayoría de tablas.

3.3. Base de datos

Para realizar el modelo de la base de datos, lo primero fue estudiar los diferentes tipos de datos que disponía la fuente y establecer las posibles relaciones entre los diferentes datos para definir las tablas.

Al disponer únicamente de ficheros de datos de alineaciones, eventos y partidos, de los cuales las alineaciones y los eventos están asociados a un partido concreto, estudie en la documentación de la fuente de datos todos los atributos que disponía cada uno de los tres tipos de ficheros que tenía, elaboré el diagrama de modelo de base de datos que podemos ver más adelante en la figura 3. En esta se siguió una nomenclatura de escribir para cada columna el nombre de la tabla añadiendo ya sea un id(identificador), nombres o cualquier tipo de información. Para las claves foráneas se ha seguido la nomenclatura de *"tablename_foreigntablename_id"*.

En un primer momento se planteó un modelo inicial para la base de datos, pero más tarde según iba avanzando el desarrollo se amplió el modelo para incluir todos los tipos de eventos que componen la base de datos, ya que incluirlo todos en una sola tabla iba a generar filas con muchas columnas y ralentizar las consultas a la tabla *Event*, que es la tabla con más registros

en la base de datos.

3.4. Modelo inicial

Este es el modelo inicial, con el que se empezó a trabajar, donde explicaré cada tabla que lo componen y los campos más importantes de cada una:

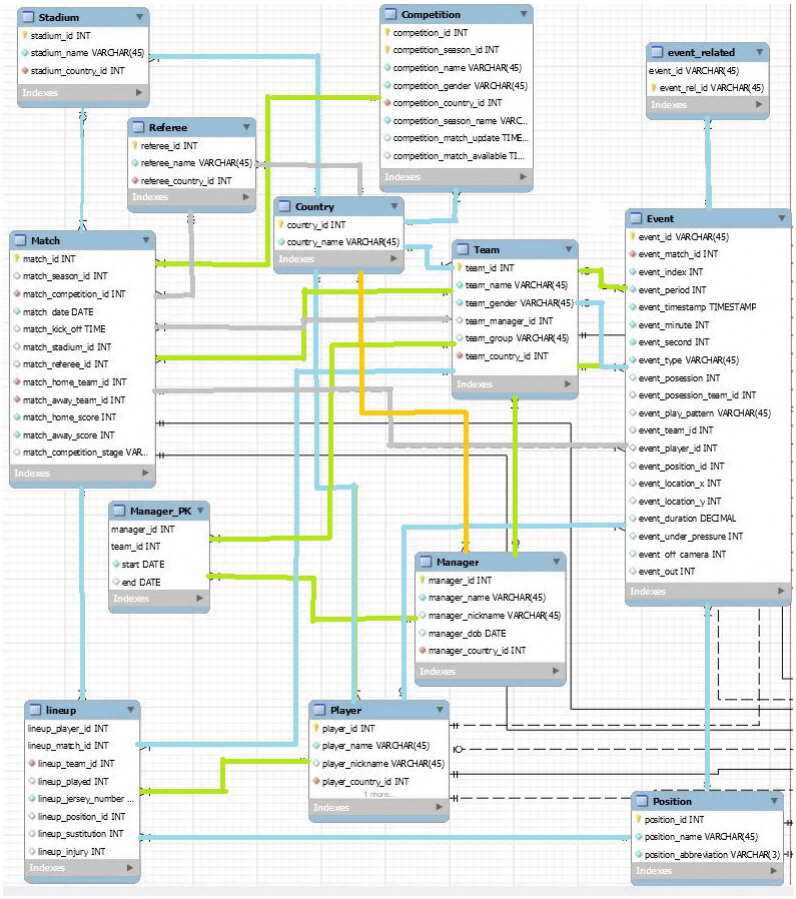

Figura 3: Modelo inicial de Base de Datos

3.4.1. Match

En primer lugar hablaré de la tabla *Match*, ya que en los ficheros iniciales todos guardan relación directamente con los partidos.

Podemos destacar de la tabla los siguientes campos:

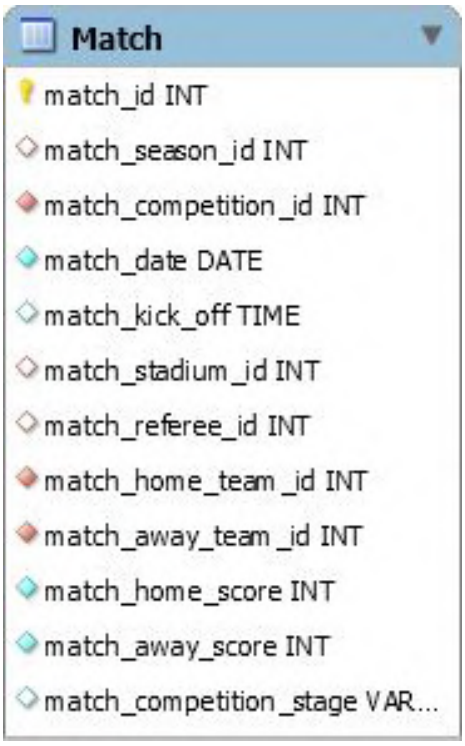

Figura 4: Tabla Match

- **match_id**; clave primaria para identificar cualquier partido dentro de la base de datos.
- **match_season_id**: identificador de la temporada en la que se disputó el partido.
- **match_competition_id**: identificador de la competición a la que pertenece.
- **match_home_team, match_away_team**:los dos equipos que jugaron el partido, distinguiendo el local del visitante.
- **match_home_score, match_away_score**: resultado del partido.

3.4.2. Player

Esta tabla contiene toda la información referente a cada jugador registrado en la base de datos.

- **player_id**: identificador del jugador.
- **player_name, player_nickname**: nombre y apodo del jugador.
- **player_country_id**: identificador del país al que pertenece el jugador.

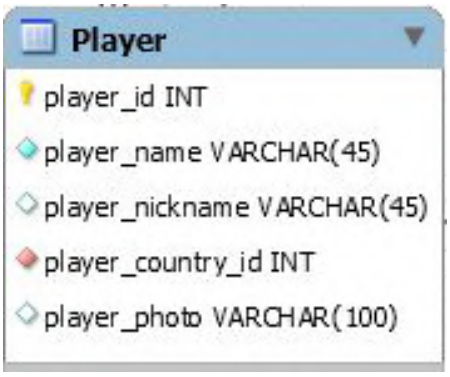

Figura 5: Tabla Player

- **player_photo**: fotografía del jugador.

3.4.3. Team

Esta tabla se compone de toda la información referente a cada equipo registrado en la base de datos.

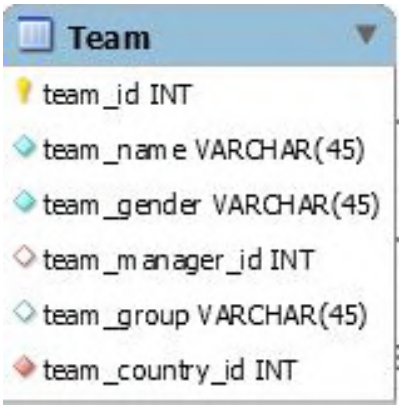

Figura 6: Tabla Team

- **team_id**: identificador del equipo.
- **team_name, team_gender**: nombre y categoría del equipo.
- **team_manager_id**: identificador del manager del equipo.
- **team_country_id**: identificador del país al que pertenece el equipo.

3.4.4. Lineup

Esta tabla contiene de toda la información referente a la alineación de un jugador en un partido concreto, donde cada alineación se compone de aquellos jugadores que disputaron ese partido.

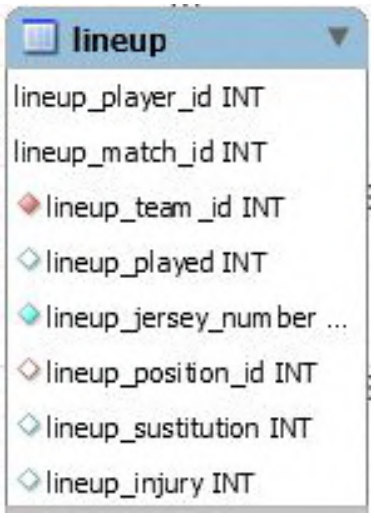

Figura 7: Tabla Lineup

- **lineup_player_id, lineup_match_id**: identificador del jugador y partido como par de clave primaria .
- **lineup_team_id**: identificador del equipo.
- **lineup_position_id**: identificador de la posición jugada por el jugador en el partido.
- **lineup_sustitution_id**: variable que guarda si el jugador fue sustituido.
- **lineup_injury** variable que indica si el jugador sufrió una lesión.

3.4.5. Competition

Esta tabla ofrece toda la información de las competiciones almacenadas en la base de datos.

- **competition_id, competition_season_id**: par de claves primarias identificadores de la competición y temporada en la que se disputó .

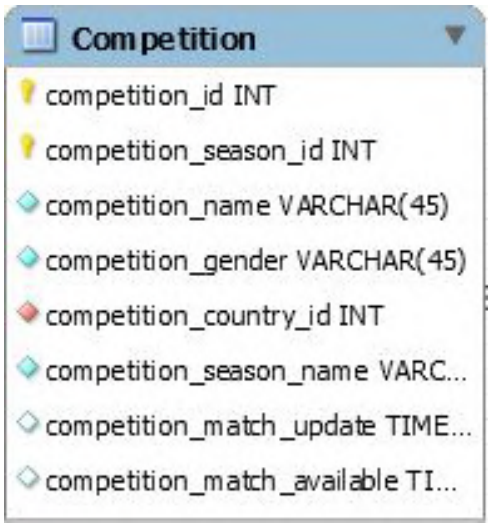

Figura 8: Tabla Competition

- **competition_name, competition_season_name**: nombre de la competición y de la temporada.
- **competition_country_id**: identificador del país donde se organiza la competición.

3.4.6. Country

En esta tabla se observa la información de los países guardados a los que pueden pertenecer las diferentes entidades.

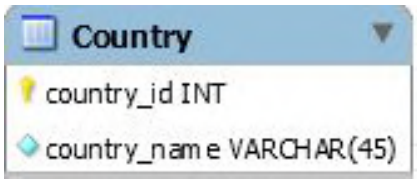

Figura 9: Tabla Country

- **country_id**: identificador del país como clave primaria.
- **country_name**: nombre del país.

3.4.7. Referee

En esta tabla se presenta la información de los árbitros guardados que han arbitrado algún partido.

Figura 10: Tabla Referee

- **referee_id**: identificador del arbitro como clave primaria.
- **referee_name**: nombre del arbitro.
- **referee_country_id**: identificador del país al que pertenece el arbitro.

3.4.8. Stadium

En esta tabla se ofrece la información de los estadios almacenados en donde se han disputado los encuentros.

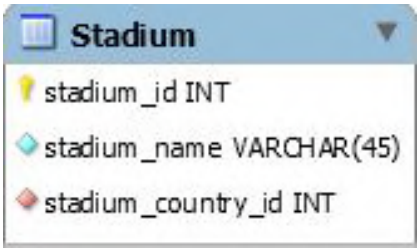

Figura 11: Tabla Stadium

- **stadium_id**: identificador del estadio como clave primaria.
- **stadium_name**: nombre del estadio.
- **stadium_country_id**: identificador del país en el que se localiza el estadio.

3.4.9. Position

Esta tabla ofrece todas las posiciones que puede ocupar un jugador en el terreno de juego.

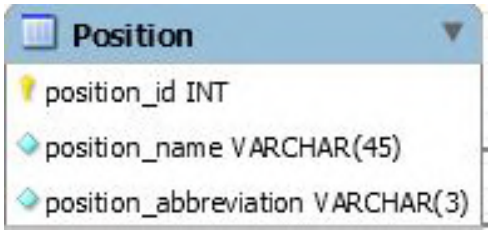

Figura 12: Tabla Position

- **position_id**: identificador de la posición como clave primaria.
- **position_name, position_abbreviation**: nombre y abreviación de la posición.

3.4.10. Manager

Esta tabla se compone de la información de los entrenadores que han dirigido a algún equipo durante un partido concreto.

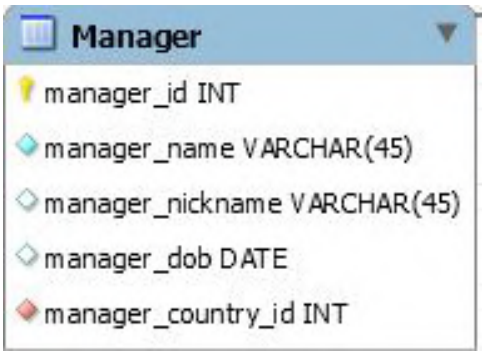

Figura 13: Tabla Manager

- **manager_id**: identificador del entrenador como clave primaria.
- **manager_name, manager_nickname**: nombre y apodo del manager.
- **manager_country_id**: identificador del país al que pertenece el manager.

3.4.11. ManagerPK

Esta tabla almacena las diferentes relaciones que se han dado entre un manager con los diferentes equipos donde ha entrenado.

Figura 14: Tabla ManagerPK

- **manager_id, team id**: identificadores del entrenador y el equipo donde entreno.
- **start, end**: fecha de inicio y de fin del entrenador en el club.

3.4.12. Event

A continuación tratamos con la tabla más grande en cuanto a columnas y registros, la tabla *Event* que contiene todos los eventos realizados por los jugadores durante un partido.

- **event_id**: identificador para cada evento como clave primaria.
- **event_match_id**: identificador del partido donde se realizó dicho evento.
- **event_minute, event_second**: campos para determinar en el momento exacto en el que se produjo el evento durante el partido.
- **event_type**: tipo de evento
- **event_posession_team_id**: el equipo que tenía el balón cuando se realizó el evento.
- **event_team_id**: identificador del equipo al que pertenece el jugador que realiza el evento.
- **event_player_id**: identificador del jugador que realiza el evento

- **event_location_x, event_location_y**: campos que registran las coordenadas X e Y de la posición en el campo donde se realizó el evento.

Figura 15: Tabla Event

3.4.13. Event Related

Esta tabla contiene la relación que tienen los eventos entre sí.

- **event_id, event_rel_id**: identificadores de los eventos relacionados como par de clave primaria.

Figura 16: Tabla Event Related

3.5. Modelo ampliado

Tras implementar e insertar casi todos los datos referentes a las tablas del modelo inicial, quedaba por último realizar las inserciones de eventos, pero se detectó que cada tipo de evento contenía información única y específica del mismo. Por lo tanto se decidió realizar una tabla para cada tipo de evento donde la clave primaria sería la misma que la tabla *Event* y se añadiría todos los campos específicos necesarios para recoger aquella información relevante del evento que no estuviera ya incluidas en la tabla principal.

De esta manera así quedaría la ampliación al modelo inicial:

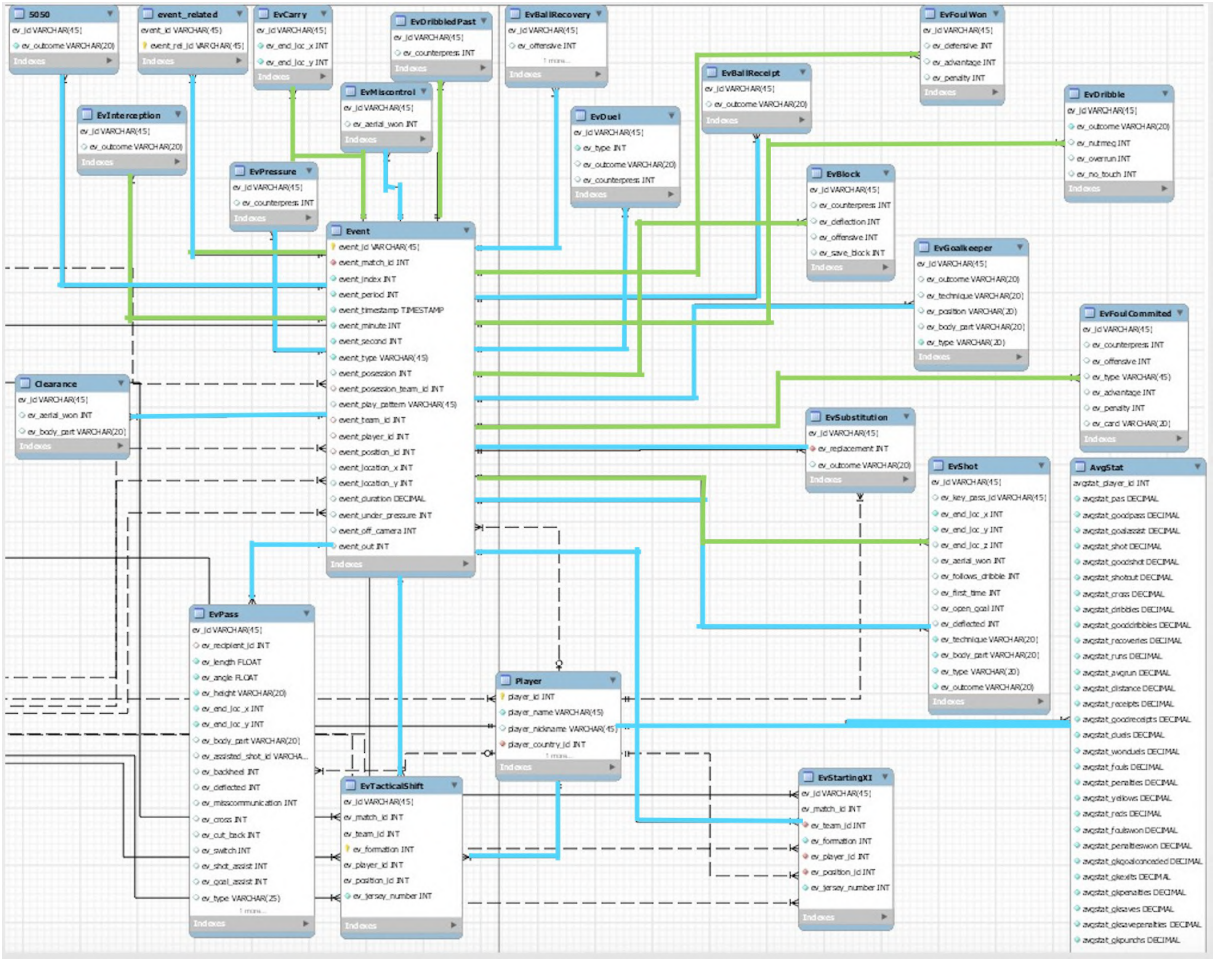

Figura 17: Modelo Ampliado

3.5.1. EvStartingXI

Esta tabla ofrece toda la información referente al evento de las alineaciones de los partidos, donde almacenamos a cada jugador de ambas alineaciones del partido.

Figura 18: Tabla EvStartingXI

- **ev_id, ev_match_id, ev_team_id, ev_player_id**: identificadores del evento, partido, equipo y jugador como tupla de clave primaria.
- **ev_position_id**: identificador de la posición del jugador.

3.5.2. EvPass

Esta tabla ofrece toda la información guardada de los pases realizados por los jugadores durante un partido.

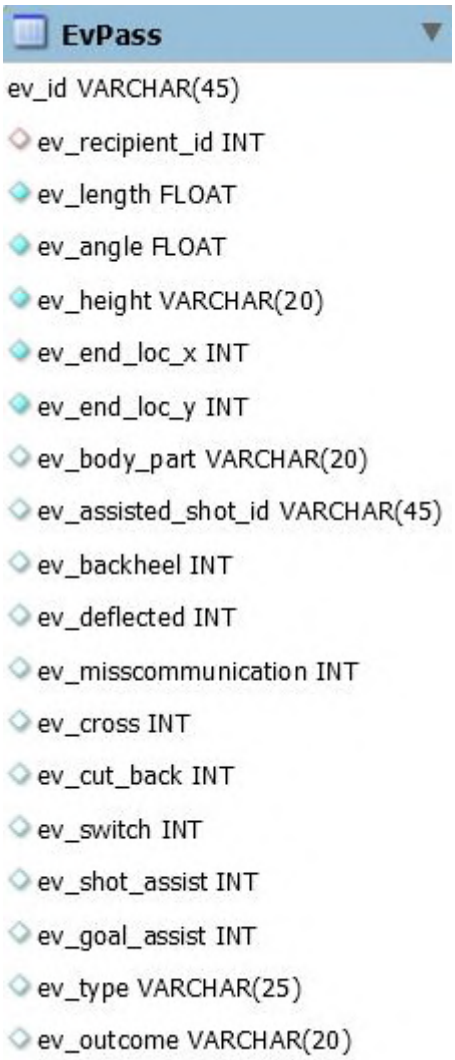

Figura 19: Tabla EvPass

- **ev_id**: identificador del evento.
- **ev_recipient_id**: identificador del jugador que recibe el pase.
- **ev_end_loc_x, ev_end_loc_y**: localización de la posición donde se efectuó el disparo con las coordenadas X y Y.

- **ev_length, ev_angle, ev_height**: longitud, ángulo y altura que coge el pase.
- **ev_body_part**: parte del cuerpo utilizada para el pase.
- **ev_assisted_shot_id**: identificador del disparo ejecutado tras este pase.
- **ev_backheel**: variable que indica si el pase se realizó de tacón.
- **ev_deflected**: variable que indica si el pase fue desviado.
- **ev_misscommunicaction**: variable que indica si el pase realizado se hizo a un jugador que no se lo esperaba.
- **ev_cross**: variable que indica si el pase que se ejecutó fue un centro al área.
- **ev_cut_back**: variable que indica si el pase efectuado fue un pase hacia atrás dentro del area.
- **ev_switch**: variable que indica si el pase que se ejecutó fue un cambio de banda.
- **ev_shot_assist**: variable que indica si el pase realizado fue un pase para que otro jugador realizara un disparo.
- **ev_goal_assist**: variable que indica si el pase fue una asistencia de gol.
- **ev_type**: tipo de pase.
- **ev_outcome**: resultado del pase.

3.5.3. EvShot

Esta tabla contiene la información de los disparos efectuados y la técnica empleada por los jugadores durante un partido.

- **ev_id**: identificador del evento.
- **ev_key_pass_id**: identificador del evento del pase.
- **ev_end_loc_x, ev_end_loc_y, ev_end_loc_z**: localización de la posición donde se efectuó el disparo con las coordenadas X,Y y Z.

Figura 20: Tabla EvShot

- **aerial_won**:variable que indica si el disparo se realiza tras un duelo aéreo ganado.
- **ev_follows_dribble**: variable que indica si el disparo se efectuó tras un regate.
- **ev_first_time**: variable que indica que el disparo se realizó al primer toque.
- **ev_open_goal**: variable que indica si el disparo se hizo a portería vacía.
- **ev_deflected**: variable que indica si el disparo fue desviado.
- **ev_technique**: información sobre la técnica usada en el disparo.
- **ev_body_part**: parte del cuerpo utilizada para el disparo.
- **ev_type**: tipo de disparo.
- **ev_outcome**: resultado del disparo.

3.5.4. EvDribble

Esta tabla almacena todos los regates realizados por los jugadores durante un partido.

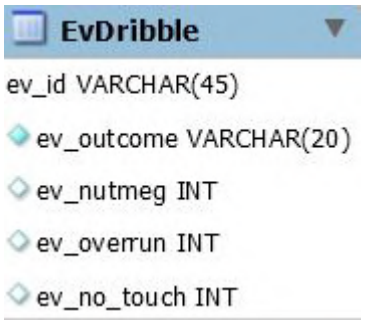

Figura 21: Tabla EvDribble

- **ev_id**: identificador del evento.
- **ev_outcome**: información sobre el regate.
- **ev_nutmeg**: variable que indica que si el regate se realizó pasando la pelota entre las piernas del rival.
- **ev_overrun**: variable que indica si se sobrepasó al jugador que le defendía la marca
- **ev_no_touch**: variable que indica si el regate se realizó sin tocar la pelota.

3.5.5. EvGoalkeeper

Esta tabla ofrece todas las acciones realizadas por los porteros durante el partido.

- **ev_id**: identificador del evento.
- **ev_outcome**: resultado sobre la acción.
- **ev_technique**: información sobre el movimiento técnico efectuado por el portero.
- **ev_position**: posición en la que se encontraba el portero durante la acción.
- **ev_body_part**: parte del cuerpo que utiliza.
- **ev_type**: tipo de acción realizada por el portero.

Figura 22: Tabla EvGoalkeeper

3.5.6. EvTacticalShift

Esta tabla detalla todos los cambios de formación realizados por un equipo durante un partido.

Figura 23: Tabla EvTacticalShift

- **ev_id, ev_match_id, ev_team_id,ev_formation, ev_player_id,ev_position_id**: identificadores del evento, partido, equipo, formación, jugador y posición como tupla de clave primaria.

3.5.7. EvSubstitution

Esta tabla contiene toda la información referente a las sustituciones realizadas durante los partidos.

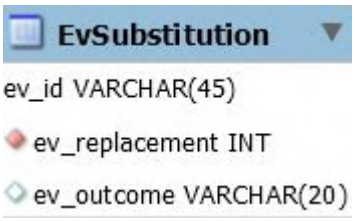

Figura 24: Tabla EvSubstitution

- **ev_id**: identificador del evento.
- **ev_replacement**: identificador del jugador.
- **ev_outcome**: motivo de la sustitución.

3.5.8. EvCarry

Esta tabla ofrece todos los movimientos de los jugadores durante un partido.

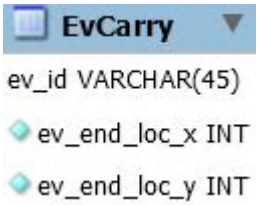

Figura 25: Tabla EvCarry

- **ev_id**: identificador del evento.
- **ev_end_loc_x, ev_end_loc_y**: localización de las coordenadas X e Y del punto final de la carrera realizada.

3.5.9. EvDribbledPast

Esta tabla recoge todos los eventos cuando un jugador ha sido regateado durante un partido.

Figura 26: Tabla EvDribbledPast

- **ev_id**: identificador del evento.
- **ev_counterpress**: variable que indica si se ejercía presión.

3.5.10. EvBallReceipt

Esta tabla contiene la información de todos los controles realizados por los jugadores durante un partido.

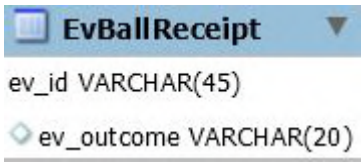

Figura 27: Tabla EvBallReceipt

- **ev_id**: identificador del evento.
- **ev_outcome**: resultado sobre el control.

3.5.11. EvBallRecovery

Esta tabla se compone de todas las recuperaciones de balón que se producen durante un partido.

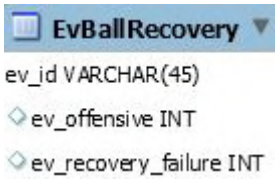

Figura 28: Tabla EvBallRecovery

- **ev_id**: identificador del evento.
- **ev_offensive**: variable que indica si se ha producido el robo en campo contrario.
- **ev_recovery_failure**: variable que indica si la recuperación se ha producido tras un fallo del rival.

3.5.12. EvDuel

Esta tabla ofrece todos los duelos que se han producido durante un partido.

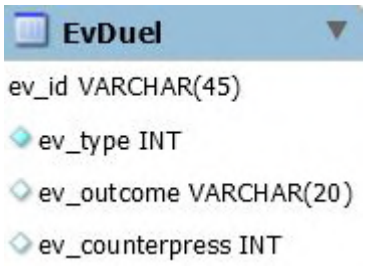

Figura 29: Tabla EvDuel

- **ev_id**: identificador del evento.
- **ev_type**: tipo de duelo
- **ev_outcome**: resultado sobre el duelo.
- **ev_counterpress**: variable que indica si se ejercía presión.

3.5.13. EvPressure

Esta tabla está compuesta por todos los movimientos de presión realizados por los jugadores durante un partido.

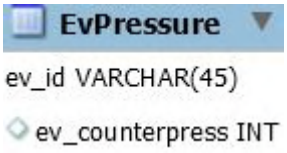

Figura 30: Tabla EvPressure

- **ev_id**: identificador del evento.
- **ev_counterpress**: variable que indica si se ejercía presión.

3.5.14. EvBlock

Esta tabla reúne todos los bloqueos realizados por un jugador a la pelota durante un partido.

Figura 31: Tabla EvBlock

- **ev_id**: identificador del evento.
- **ev_counterpress**: variable que indica si se ejercía presión.
- **ev_deflection**: variable que indica si el bloqueo se produjo de una desviación.
- **ev_offensive**: variable que indica si se produjo en campo contrario.
- **ev_save_block**: variable que indica si el bloqueo evito un disparo a portería.

3.5.15. Clearance

Esta tabla muestra todos los despejes de balón realizados por un jugador durante un partido.

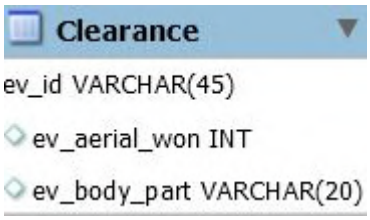

Figura 32: Tabla Clearance

- **ev_id**: identificador del evento.
- **ev_aerial_won**: variable que indica si el despeje se realiza tras un duelo aéreo ganado.
- **ev_body_part**: indica la parte del cuerpo utilizada.

3.5.16. EvMiscontrol

Esta tabla contiene todos los controles perdidos por un jugador al recibir la pelota durante un partido.

Figura 33: Tabla EvMiscontrol

- **ev_id**: identificador del evento.
- **ev_aerial_won**: variable que indica si el despeje se realiza tras un duelo aéreo ganado.

3.5.17. EvInterception

En esta tabla se observa todas las intercepciones realizadas por un jugador a otro durante un partido.

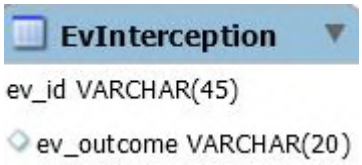

Figura 34: Tabla EvInterception

- **ev_id**: identificador del evento.
- **ev_outcome**: resultado sobre la intercepción.

3.5.18. EvFoulWon

En esta tabla se recogen todas las faltas ganadas que consiguió un jugador durante los partidos.

Figura 35: Tabla EvFoulWon

- **ev_id**: identificador del evento.
- **ev_defensive**: variable que indica si se realizó mientras se defendía.
- **ev_advantage**: variable que indica si se concedió ley de la ventaja cuando se cometió la falta.
- **ev_penalty**: variable que indica si se cometió penalti con la falta.

3.5.19. EvFoulCommetid

En esta tabla se guardan todas las faltas cometidas por un jugador a otro durante un partido.

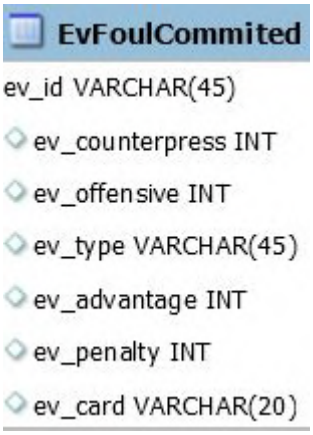

Figura 36: Tabla EvFoulCommited

- **ev_id**: identificador del evento.
- **ev_offensive**: variable que indica si se cometió en campo contrario.
- **ev_type**: indica el tipo de falta que se cometió.
- **ev_advantage**: variable que indica si se concedió ley de la ventaja cuando se cometió la falta.
- **ev_penalty**: variable que indica si se cometió penalti con la falta.
- **ev_card**: indica si la falta acarreó alguna tarjeta y de que tipo.

3.5.20. Ev5050

Esta tabla contiene la información de quien ha salido ganador de alguna disputa entre los dos equipos.

- **ev_id**: identificador del evento.
- **ev_outcome**: resultado sobre la disputa.

Figura 37: Tabla Ev5050

3.6. Estadísticas

Para realizar algunos cálculos estadísticos se ha aplicado la media aritmética, debido al tiempo de ejecución de algunas de las consultas se decidió almacenar las estadísticas medias por partido de todos los jugadores y para cada temporada para ahorrar tiempo de cálculo.

- **avgstat_player_id, avgstat_season_id**: identificadores del jugador y la temporada como par de clave primaria.
- **avgstat_pas**: media de pases intentados por partido.
- **avgstat_goodpass**: media de pases acertados por partido.
- **avgstat_goalassist**: media de asistencias de gol por partido.
- **avgstat_shot**: media de tiros intentados por partido.
- **avgstat_goodshot**: media de goles por partido.
- **avgstat_shoutout**: media de tiros fuera por partido.
- **avgstat_cross**: media de centros por partido.
- **avgstat_dribbles**: media de regates intentados por partido.
- **avgstat_gooddribbles**: media de regates acertados por partido.
- **avgstat_recoveries**: media de recuperaciones por partido.
- **avgstat_runs**: media de carreras realizadas por partido.
- **avgstat_avgrun**: media de longitud de carreras realizadas por partido.
- **avgstat_distance**: media de distancia recorrida por partido.

- **avgstat_receipts**: media de controles intentados por partido.
- **avgstat_goodreceipts**: media de controles acertados por partido.
- **avgstat_duels**: media de duelos realizados por partido.
- **avgstat_wonduels**: media de duelos ganados por partido.
- **avgstat_fouls**: media de faltas realizadas por partido.
- **avgstat_penalties**: media de penaltis cometidos por partido.
- **avgstat_yellows**: media de amarillas por partido.
- **avgstat_reds**: media de rojas por partido.
- **avgstat_foulswon**: media de faltas ganadas por partido.
- **avgstat_penaltieswon**: media de penaltis provocados por partido.
- **avgstat_gkgoalconceded**: media de goles concedido por partido como portero.
- **avgstat_gkexits**: media de salida de portería por partido como portero.
- **avgstat_gkpenalties**: media de penaltis cometidos como portero por partido.
- **avgstat_gksaves**: media de paradas por partido como portero.
- **avgstat_gkgoalsavepenalties**: media de penaltis parados por partido.
- **avgstat_gkpunchs**: media de despejes realizados como portero por partido.
- **avgstat_starts**: numero de partidos jugados como titular durante toda su carrera.
- **avgstat_games**: número de partidos jugados.

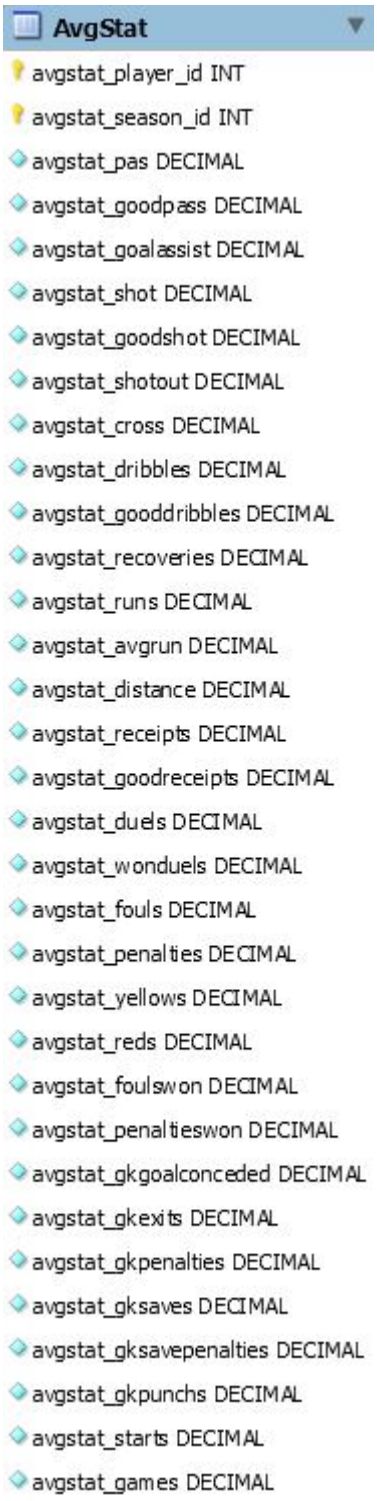

Figura 38: Tabla Avgstat

4

Desarrollo del Sistema de Información

Este capítulo va a tratar sobre el desarrollo del sistema de información del proyecto, donde como hemos comentado anteriormente en las tecnologías, he utilizado como lenguaje de programación Python y usando el framework Flask. Para realizar las peticiones a la base de datos, decidí usar el framework disponible en Flask de SQLAlchemy el cual facilita mucho la implementación de las peticiones a la base de datos.

Voy a detallar a continuación todos los *endpoint*, métodos de cálculo y métodos auxiliares que dispone el servidor de este proyecto.

4.1. Endpoints

En esta sección vamos a tratar cada punto de acceso los cuales los vamos a definir con la ruta que utiliza dentro del servidor.

4.1.1. /

Esta primera ruta corresponde con la página principal de la interfaz web, si realizamos una respuesta de *GET* nos devolverá simplemente la plantilla correspondiente a la página de inicio. Si por el contrario se recibe una respuesta *POST*, se recogerá el valor que ha sido introducido en el campo de búsqueda de la interfaz web y se realizará una petición a la base de datos para obtener los datos históricos del jugador introducido, además de las temporadas en las que ha jugado.

4.1.2. /listado

Esta ruta controla la página del listado de jugadores de la interfaz web, disponiendo únicamente de una respuesta de tipo *GET* la cual realizará una petición a la base de datos para devolver un listado con todos los jugadores disponibles en la base de datos. A continuación dividirá ese listado en 4 listas que filtrará según la posición que juegue el jugador:

- **Porteros**
- **Defensas**
- **Centrocampistas**
- **Delanteros**

4.1.3. /jugador/pid

Esta ruta también está asociada a la página de listado donde recibe una respuesta de tipo *GET* acompañada de un id de un jugador, el cual utilizaremos para realizar una petición a la base de datos para devolver todos los datos históricos asociados a ese jugador y cargar la plantilla de la página principal con los datos históricos del jugador ya cargados.

4.1.4. /comparativa

Esta ruta está ligada a la página de comparativa de la interfaz web, donde si recibe una respuesta de tipo *GET* cargará la plantilla de la página. En cambio, si recibe una respuesta de tipo *POST* recoge los dos campos introducidos en la comparativa y realiza peticiones a la base de datos para obtener los datos históricos de ambos jugadores.

Además de realizar llamadas a los métodos de cálculo que serán requeridos en la interfaz de la comparativa, controlando de que si ya está almacenada la gráfica que se pretende calcular, se devuelve directamente sin llamar al método.

4.1.5. /season

Esta llamada, está vinculada a la selección de temporada dentro de la página principal, recibiendo únicamente una respuesta de tipo *POST* que es la temporada seleccionada. Por lo que se realizará una petición a la base de datos para devolver los partidos disputados durante la temporada seleccionada.

Adicionalmente también se llamará al método de crear la gráfica comparativa entre los datos históricos del jugador con los datos de esa temporada para observar como fue el rendimiento de esa temporada frente a su histórico.

4.1.6. /match

Esta ruta está asociada a la selección de partido dentro de la página principal, donde se recibe solamente una respuesta de tipo *POST* que es el partido seleccionado, además de la temporada que ya estaba seleccionada previamente. Con el ID del jugador, la temporada y el partido realizaremos varias peticiones a la base de datos para obtener todos los datos estadísticos más relevantes del jugador en ese partido.

También realizaremos una llamada al método que calcula la comparación entre el rendimiento del jugador en el partido para compararlo con su rendimiento en la temporada y en el histórico. Además de llamar también a uno de los métodos auxiliares para comprobar que los datos obtenidos no tienen errores o están vacíos.

4.1.7. /chart

Esta ruta será una llamada que se realizará desde la interfaz web una vez finalizada la llamada *'/match'* y que recibirá una respuesta de tipo *POST*, donde se recogerá de la interfaz web el id del jugador, la temporada y el partido seleccionado para llamar a todos los métodos de cálculos requeridos y devolver todas esas gráficas.

4.2. Generación de gráficos

En esta sección vamos a detallar todos los métodos que utilizamos para generar las gráficas para la interfaz web. Todos los métodos han sido implementados gracias a la librería Matplotlib [9], la cual ofrece multitud de formas de interpretar los datos y generar diferentes maneras de mostrar los datos.

Debido a que el procesamiento de datos para realizar alguna gráfica puede ser un tiempo considerable, se almacenará la imagen generada en el servidor, para que cada vez que se ejecute algún método para generar una gráfica, se comprobará si ya existe para mostrarla directamente, disminuyendo así los tiempos de carga.

4.2.1. drawChartH

Este método se encarga de generar la gráfica comparativa entre los datos históricos y los datos de una temporada en concreto de un jugador. Recibe como parámetros el id del jugador y la temporada en cuestión, los cuales se utilizarán para realizar peticiones a la base de datos para recoger todos los datos necesarios para incluir en la gráfica.

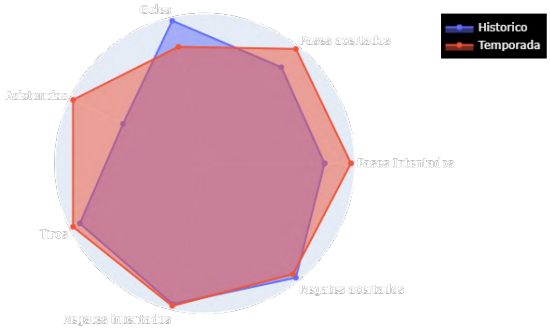

Figura 39: Gráfica generada

4.2.2. drawChartM

Este método tiene el mismo comportamiento que drawChartH, pero la diferencia es que añade a la comparación los datos de un partido concreto, ya que recibe por parámetro el id del jugador, una temporada y un partido seleccionado, los cuales se utilizaran para realizar peticiones a la base de datos para recoger todos los datos necesarios para incluir en la gráfica.

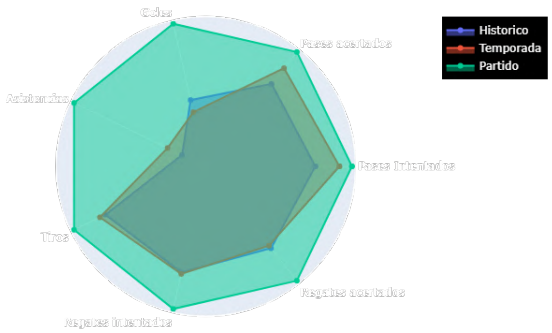

Figura 40: Gráfica generada

4.2.3. drawChartComparative

Este método realiza una comparación entre los datos históricos de dos jugadores, recibiendo como parámetro los id de ambos jugadores, los cuales se utilizaran para realizar peticiones a la base de datos para recoger todos los datos necesarios para incluir en la gráfica.

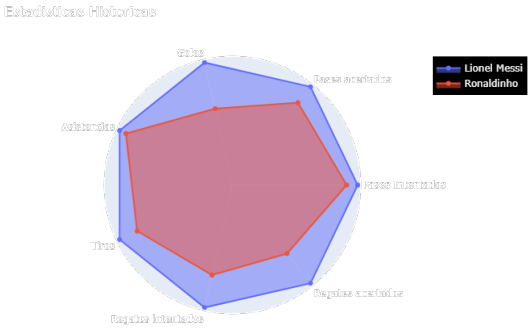

Figura 41: Gráfica generada

4.2.4. drawActionMap

Este método muestra el campo dividido en 32 zonas con 4 filas y 8 columnas, donde cada zona mostrará el porcentaje de eventos realizados por un jugador en esa zona específica. Este método puede recibir como parámetro:

- El id del jugador y el id de la temporada para mostrar las zonas de acción durante toda la temporada.
- El id del jugador y el id de un partido en concreto para mostrar las zonas de acción durante ese partido concreto.

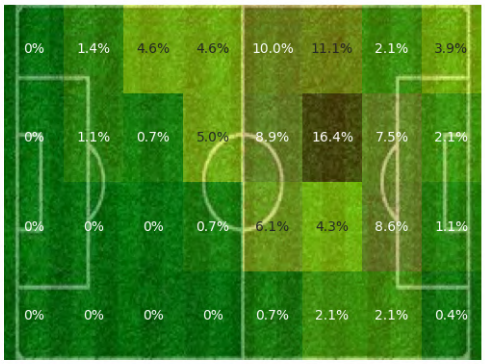

Figura 42: Gráfica generada

4.2.5. drawShots

Este método muestra la portería dividida en 12 zonas con 3 filas y 4 columnas, por los cuales cada zona muestra el porcentaje de tiros realizados por un jugador en esa zona específica. Este método puede recibir como parámetros:

- El id del jugador y el id de la temporada para mostrar los disparos a portería del jugador durante toda la temporada.
- El id del jugador y el id de un partido en concreto para mostrar los disparos de ese partido en concreto.

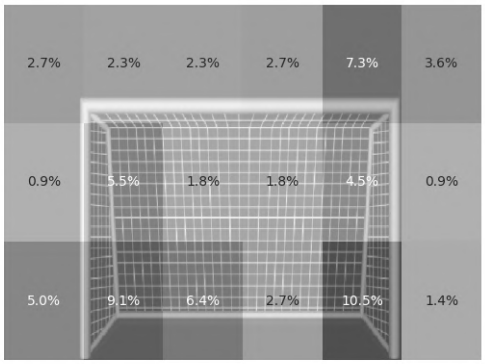

Figura 43: Gráfica de disparos generada

4.2.6. drawHeatMap

Este método muestra un mapa de calor de las acciones del jugador en el campo, donde según se vayan acumulando más eventos en un mismo punto mostrará un color más cálido. Este método puede recibir como parámetros:

- Únicamente el id del jugador para mostrar el mapa de calor histórico del jugador.
- El id el jugador y el id de una temporada específica para mostrar el mapa de calor de esa temporada.
- El id del jugador, el id de la temporada y el id de un partido específico, para mostrar el mapa de calor de ese partido en concreto.

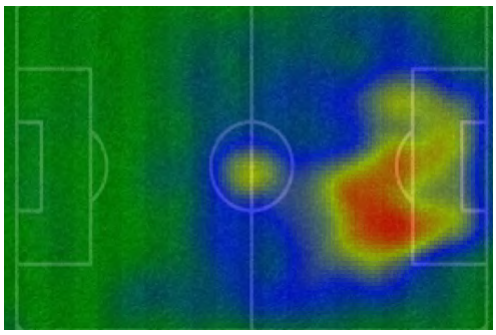

Figura 44: Gráfica generada

4.3. Métodos auxiliares

Ahora vamos a explicar todos aquellos métodos que realizan alguna función secundaria para asegurar el correcto funcionamiento de algunos métodos o para mostrar los datos.

4.3.1. makeLabels(array)

Este método recibe como parámetro un array de string y le añade al final un *'%'* para los métodos de *drawActionMap* y *drawShots*.

4.3.2. percentage(a,b)

Este método recibe dos números y realiza la división entre ellos y lo multiplica por cien para obtenerlo como porcentaje. En adición se realiza una comprobación que los valores pasados como parámetros son nulos o cero.

4.3.3. check(a,b,n)

Este método recibe 3 valores como parámetros, donde los dos primeros son valores que queremos realizar alguna comprobación y el tercer parámetro es un valor que indica la comprobación que queremos realizar:

- Si n vale 0, indica que solo vamos a evaluar si el primer parámetro es cero o nulo para devolver cero o el valor en caso contrario.
- Si n vale 1, comprobamos que los dos valores no son nulos o ceros y se devuelve la división entre ambos.
- Si n vale 2, comprobamos que los dos valores no son nulos o ceros y se devuelve la resta entre ambos.

4.3.4. checkSQL(sql)

Este método recibe como parámetro el resultado de una consulta y comprueba si contiene valores nulos para convertirlos en cero.

4.3.5. bigger(a,b)

Este método recibe como parámetros dos valores y devuelve el mayor de ambos.

5
Desarrollo de Interfaz

En este capítulo voy a explicar como se ha llevado a cabo todo el desarrollo de la interfaz, comenzando con los bocetos de cada interfaz y la idea que se quería alcanzar en cada apartado. Además de estos bocetos, se explicará detalladamente las funcionalidades que ofrece cada uno y el esquema de navegación de la página.

5.1. Prototipo

La finalidad de la página web es mostrar los datos de un jugador de manera visual para analizar su rendimiento, por lo tanto, dedicaremos un apartado de la página para visualizar las estadísticas de dicho jugador. En primer lugar, la interfaz web dispondrá en todas las páginas de una barra de navegación que nos permitirá acceder a cualquier apartado de la página. Por otro lado, para realizar la búsqueda de un jugador desde el sistema de información se han planteado dos maneras diferentes para llevarlo a cabo y estas son mediante la búsqueda del nombre del jugador y otra se realizará tras seleccionar a un jugador desde el listado.

5.1.1. Listado

Este apartado será dedicado especialmente a la página de *Listado* para tenerlo independiente de la página del jugador, dicho listado se compondrá de los jugadores mostrados con una imagen suya y pinchando en ella nos llevara a la página de ese jugador con los datos históricos cargados. Para hacerlo más visual se dividirá a los jugadores según su posición ya sea portero, defensa, centrocampista o delantero.

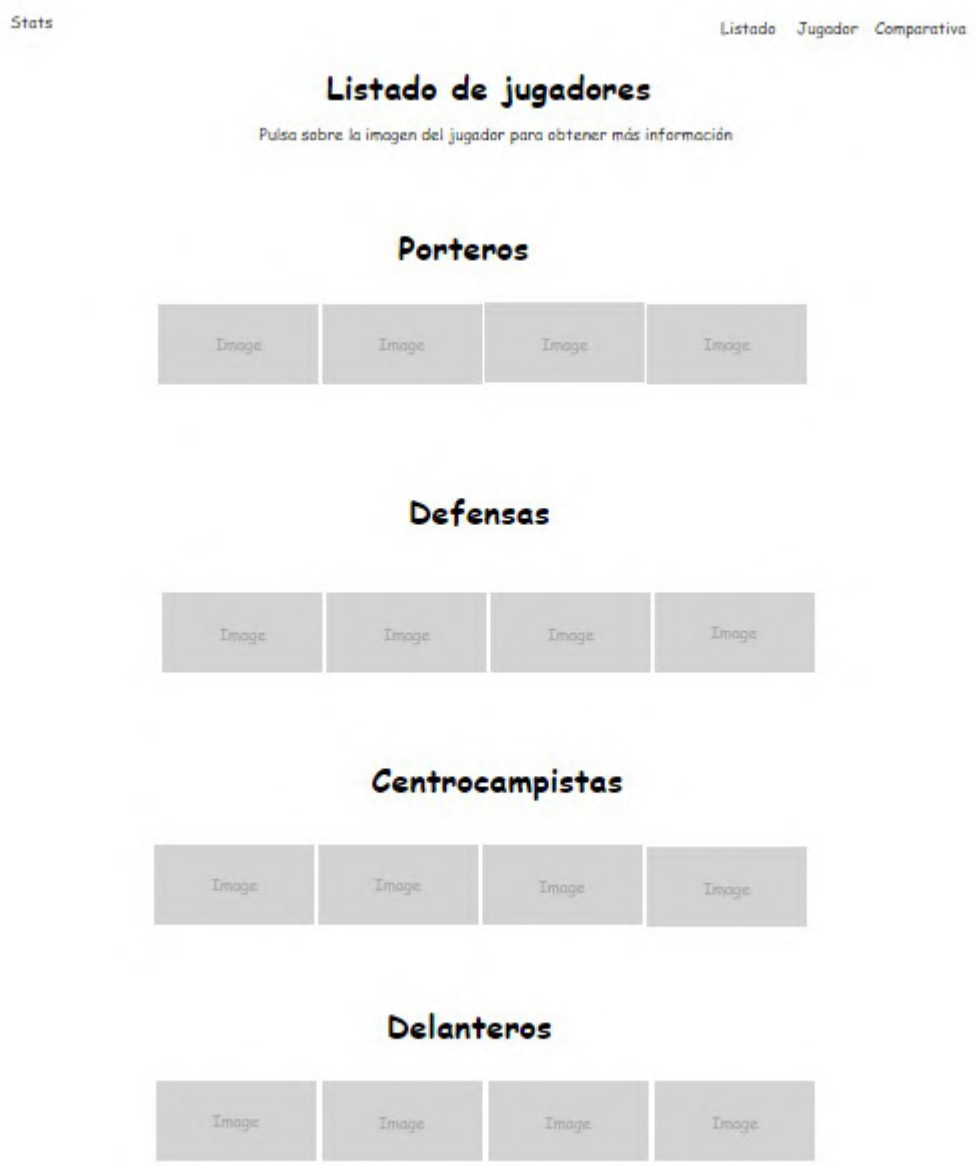

Figura 45: Prototipo de la interfaz Listado

5.1.2. Jugador

Como hemos mencionando anteriormente, tendremos un apartado donde visualizamos las estadísticas de los jugadores y para seleccionar el jugador dispondremos de una barra búsqueda que recogerá el nombre del jugador introducido y realizará una petición a la base de datos para que nos devuelva los datos de ese jugador si existe en la base de datos.

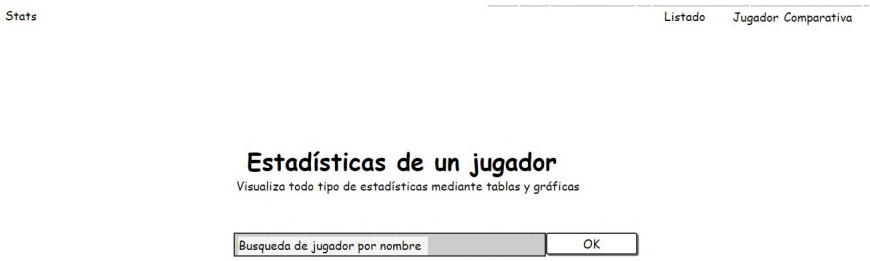

Figura 46: Prototipo de la interfaz Jugador inicialmente

En caso de no encontrar resultados, mostrará un mensaje por pantalla avisando de que no hay resultados para esa búsqueda.

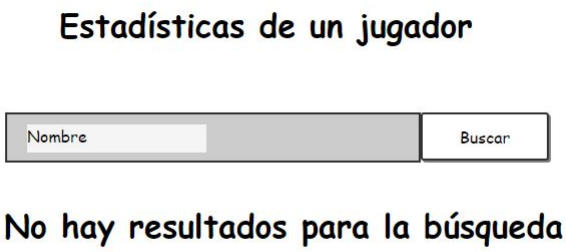

Figura 47: Prototipo de la interfaz Jugador sin resultado de búsqueda

Por el contrario, si encontramos un jugador que coincida con nuestra búsqueda, nos mostrará su foto y sus datos históricos, así como también la posibilidad de elegir entre una de sus temporadas disputadas.

Figura 48: Prototipo de la interfaz Jugador con datos históricos

Una vez seleccionamos una temporada, se cargarán tanto los datos del jugador además de una gráfica comparativa entre sus datos históricos y los datos de dicha temporada, donde se apreciará si el rendimiento del jugador mejoró, empeoró o se mantuvo en su línea. Adicionalmente se cargará un select con todos los partidos disputados de esta temporada para así elegir uno.

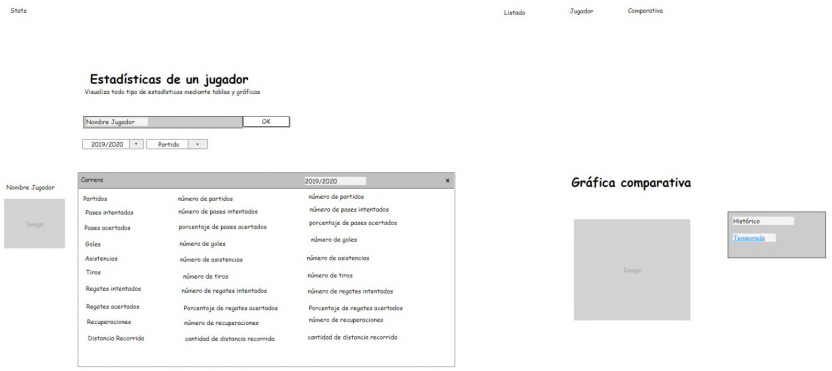

Figura 49: Prototipo de la interfaz Jugador con datos de la temporada

A continuación, si seleccionamos un partido, se mostrarán (tras cargar el resultado de la petición a la base de datos) todos los datos del jugador en ese partido y la gráfica ofrecerá también dentro de la comparativa los datos del partido. De esta forma se podrá evaluar cómo fue el rendimiento del jugador en el partido en comparación con los datos históricos y los de la temporada.

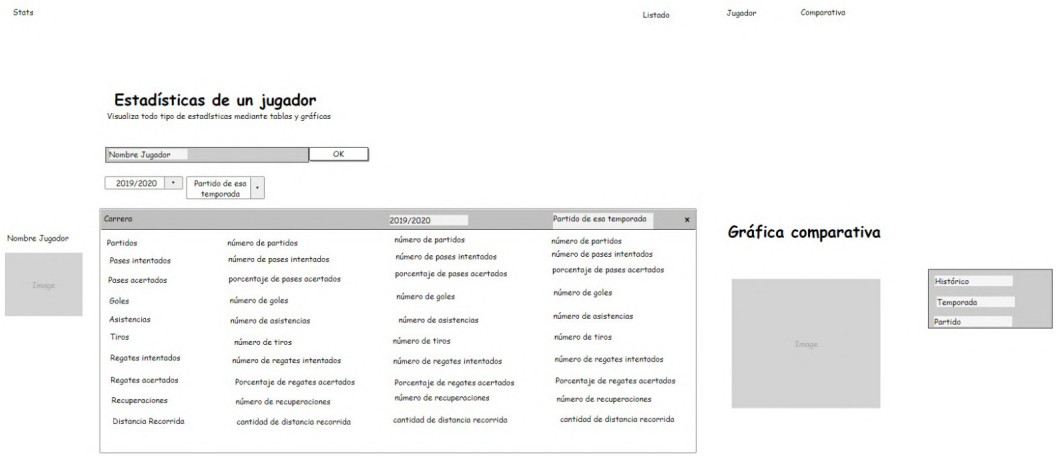

Figura 50: Prototipo de la interfaz Jugador con datos del partido

A su vez, se realizará la llamada de otros métodos al servidor para cargar más gráficas, lo que conlleva un pequeño tiempo de cálculo y es por ello por lo que decidí dividir este paso en 2 llamadas. De esta forma se podrá mostrar en primer lugar la gráfica comparativa mientras se cargan las gráficas las cuales ofrecerán más detalles.

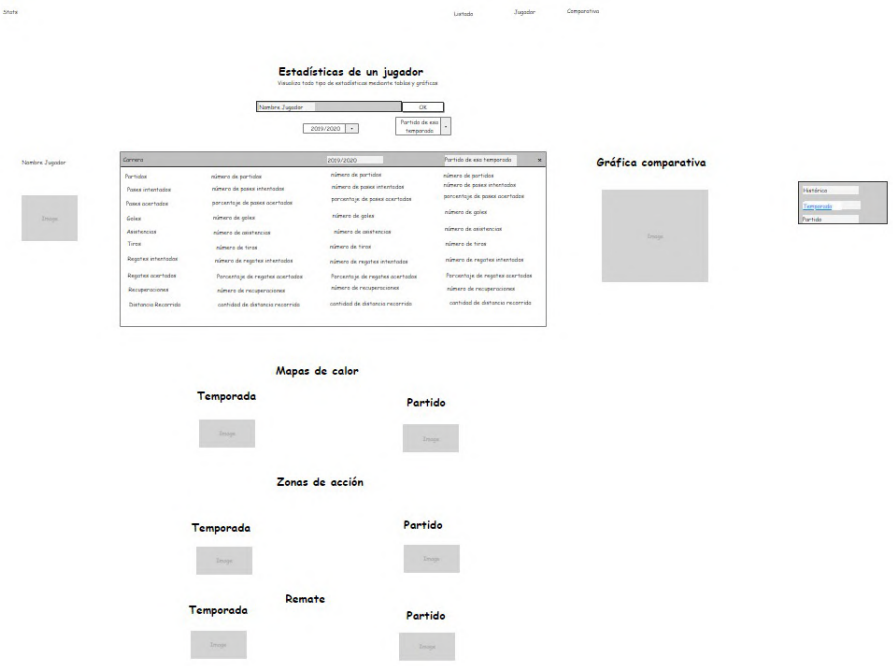

Figura 51: Prototipo de la interfaz Jugador con más detalles

5.1.3. Comparativa

Aunque inicialmente no se contempló esta idea, finalmente se implementó dentro de la interfaz un apartado para realizar una comparativa entre dos jugadores, donde se reutilizará toda la lógica ya desarrollada en el apartado *Jugador* para implementar dos jugadores a modo de comparativa, comenzando con la búsqueda por nombre de ambos.

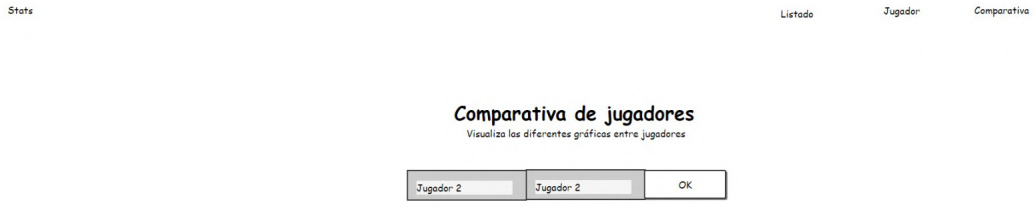

Figura 52: Prototipo de la interfaz Comparativa

Una vez introducido el nombre de ambos jugadores y obteniendo una respuesta de la petición de la base de datos, se mostrará el nombre y la foto de ambos, así como una gráfica comparativa entre los datos históricos de los jugadores y bajo éstos las estadísticas de manera detallada, ofreciendo varias gráficas de ambos jugadores para compararlas entre sí.

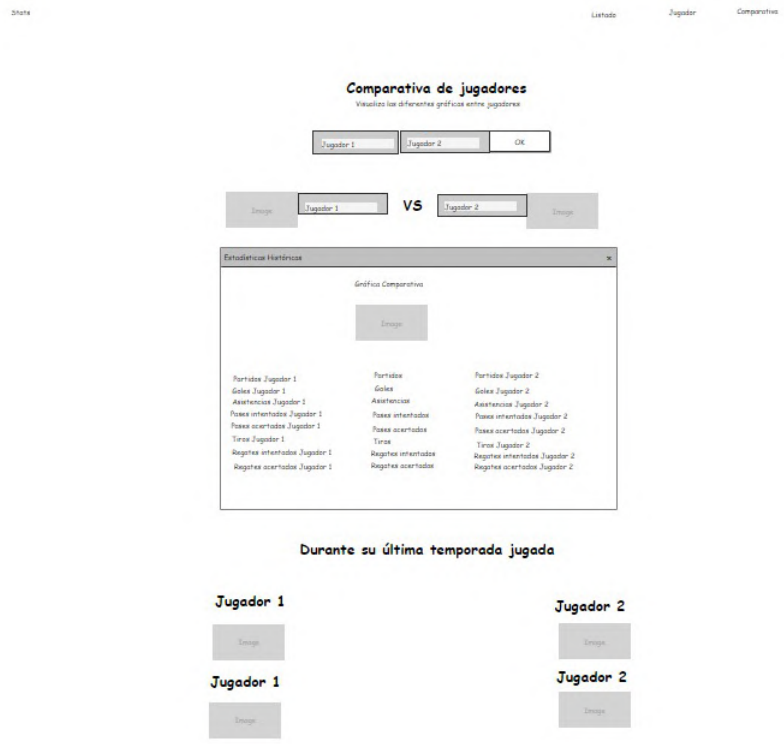

Figura 53: Prototipo de la interfaz Jugador con más detalles

5.2. Diseño final

Una vez terminado el desarrollo de la interfaz este ha sido el resultado:

5.2.1. Listado

Aquí tenemos el resultado final de la interfaz de listado:

Figura 54: Interfaz Listado

Si dejamos el ratón por encima de la foto de un jugador, esta nos mostrará el nombre del jugador y si pinchamos en ella nos llevará a las estadísticas del jugador ya cargadas en el apartado *Jugador*.

Figura 55: Interfaz Listado con ratón sobre jugador

5.2.2. Jugador

En primer lugar, tenemos la vista inicial de este apartado, mostrando la barra de búsqueda para introducir el nombre de un jugador.

Figura 56: Interfaz Jugador inicialmente

Si introducimos el nombre de un jugador, pero no obtenemos ningún resultado de la petición a la base de datos nos aparecerá este mensaje advirtiendo que no obtuvimos resultados.

Figura 57: Interfaz Jugador busqueda fallida

En cambio, si obtenemos un jugador se mostrará en la pantalla su foto y una tabla compuesta por sus datos históricos por partido y se cargará el select de temporada con todas las temporadas disputadas por el jugador.

Figura 58: Interfaz Jugador búsqueda con éxito

Al seleccionar una temporada se enviará una petición a la base de datos la cual nos devolverá aquellos datos del jugador cosechados durante la temporada. Se desplegará en la tabla una nueva columna con estos datos y a su vez, a la derecha aparecerá una gráfica comparativa entre los datos históricos y los datos de esa temporada del jugador. Además, se cargará el select de partidos con aquellos partidos disputados por el jugador durante la temporada seleccionada.

Figura 59: Interfaz Jugador con datos temporada

Si seleccionamos un partido, realizará la petición a la base de datos y devolverá los datos del jugador referentes a ese partido, donde se añadirá una nueva columna a la tabla y la gráfica comparativa de los datos del partido. En adición, mientras se realizan los cálculos del resto de gráficas, se ha colocado un gif con un mensaje avisando de la carga de más datos.

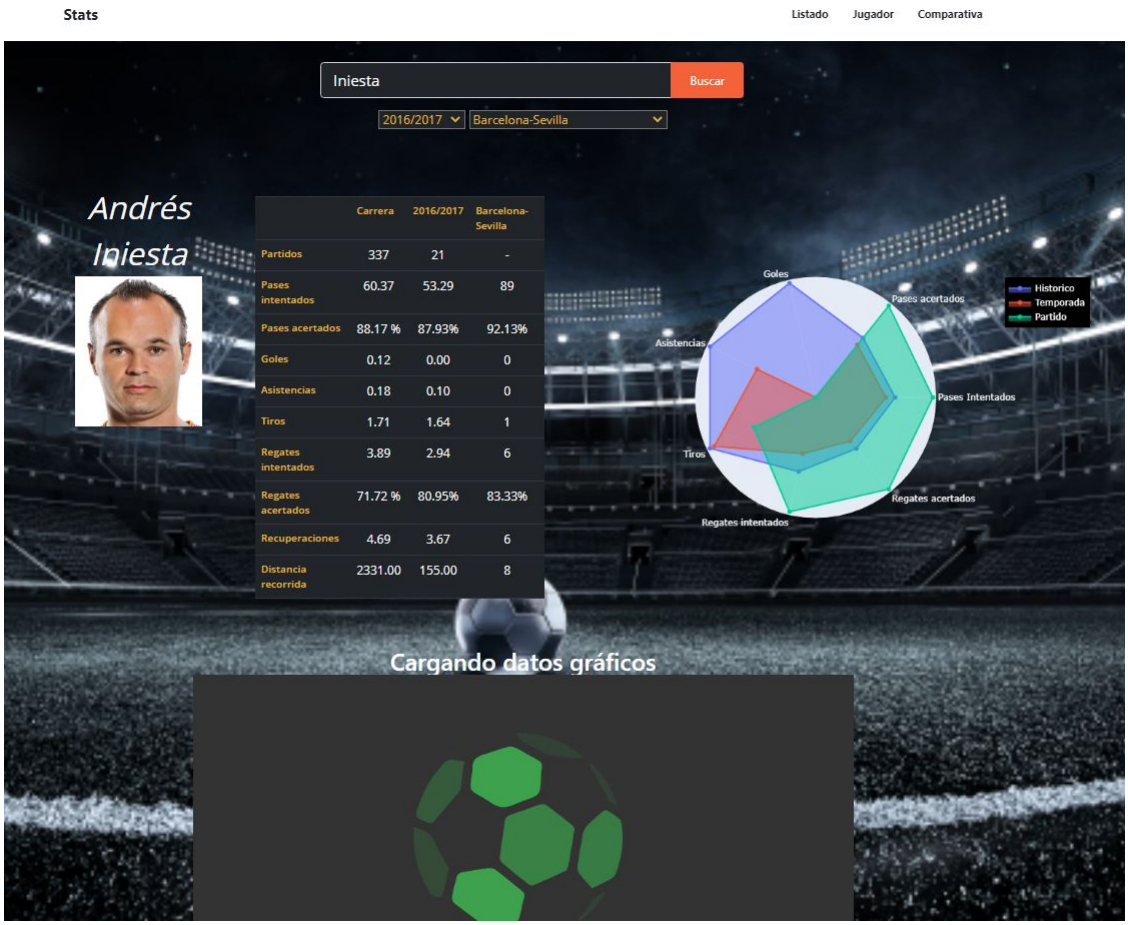

Figura 60: Interfaz Jugador con datos de partido

Una vez se completa la carga desaparece el gif y aparecen las gráficas donde tenemos información relevante como el mapa de calor, las zonas de acción y de disparo. Podemos ver tanto las gráficas de la temporada como las del partido.

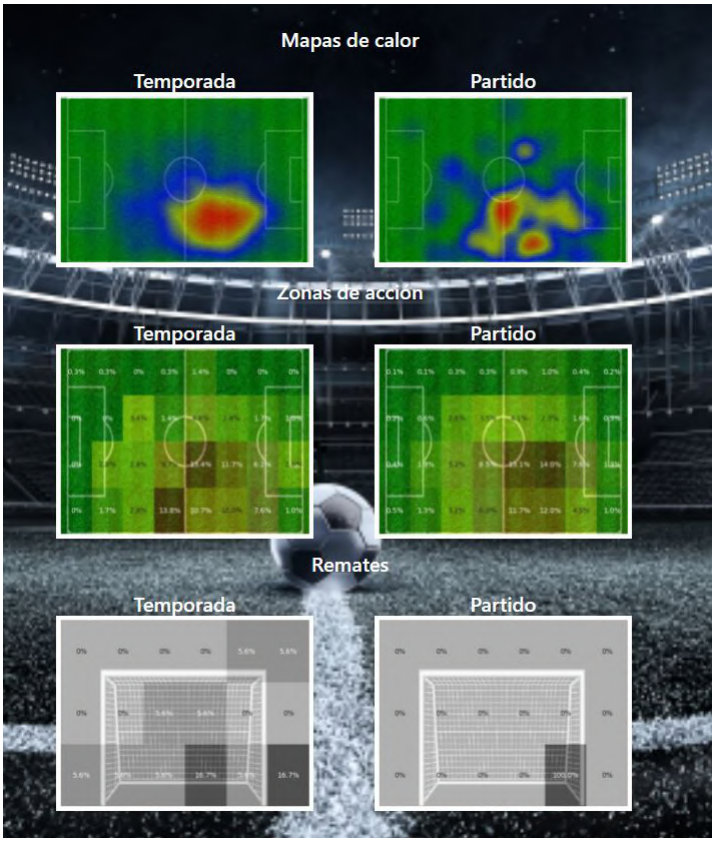

Figura 61: Interfaz Jugador con gráficas cargadas

5.2.3. Comparativa

En primer lugar tenemos la barra de búsqueda para introducir los nombres de dos jugadores.

Figura 62: Interfaz Comparativa

Si de la respuesta a la petición de la base de datos no obtenemos resultados, aparecerá un mensaje como en el apartado *Jugador* avisando que no se han encontrado resultados para la búsqueda.

Figura 63: Interfaz comparativa con búsqueda fallida

En cambio, si obtenemos los resultados de la petición, se mostrará en la pantalla tanto las fotos, como los nombres de ambos futbolistas, así como también una gráfica comparativa con los datos históricos de estos. Bajo dicha gráfica aparecerán los detalles de la misma con un mayor número de detalles. A su vez, en la parte inferior podremos observar una serie de gráficas relevantes sobre datos de la carrera de los dos jugadores para poder compararlas, siendo estas el mapa de calor y las zonas de disparo.

Figura 64: Interfaz Comparativa con los datos cargados

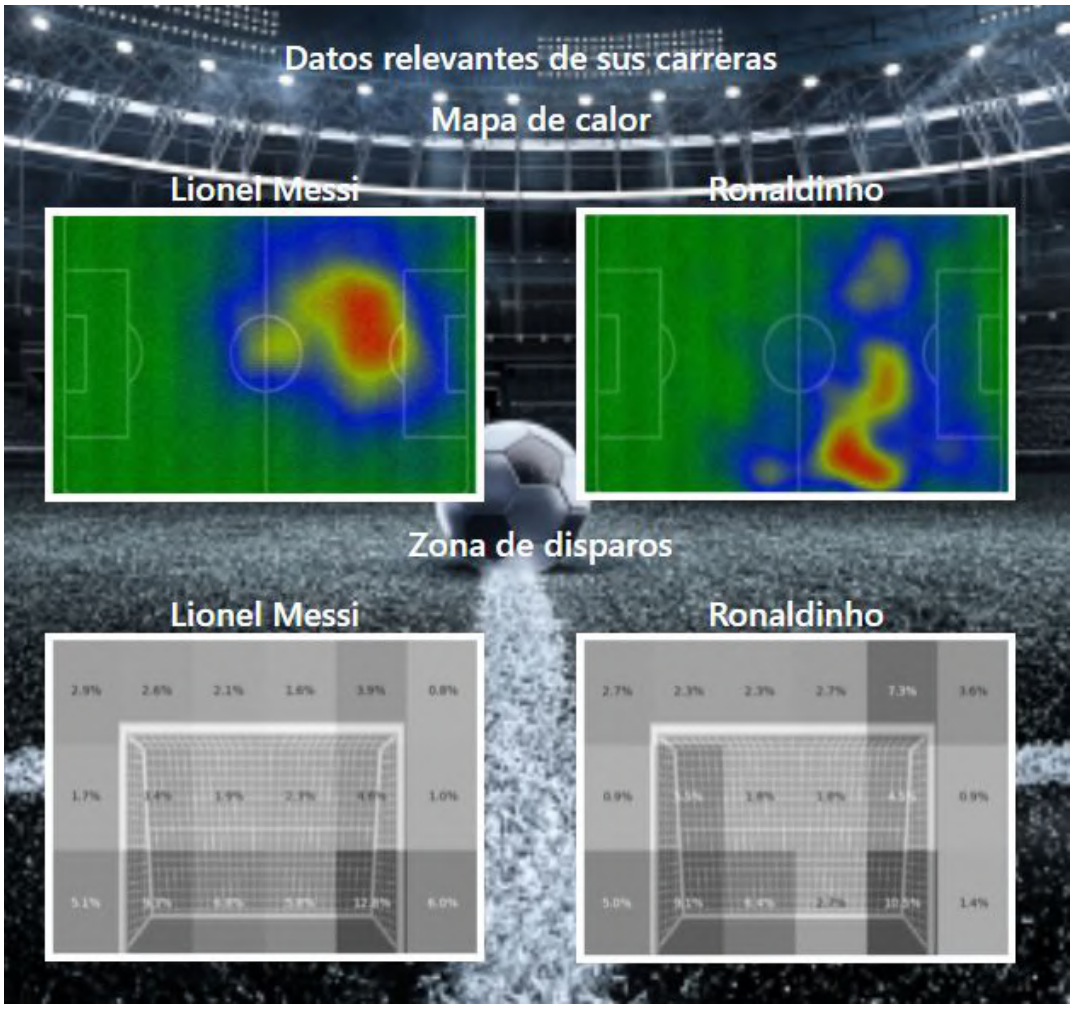

Figura 65: Interfaz Comparativa con la gráfica cargada

5.3. Esquema de navegación

Aquí tenemos el esquema de navegación de la interfaz web:

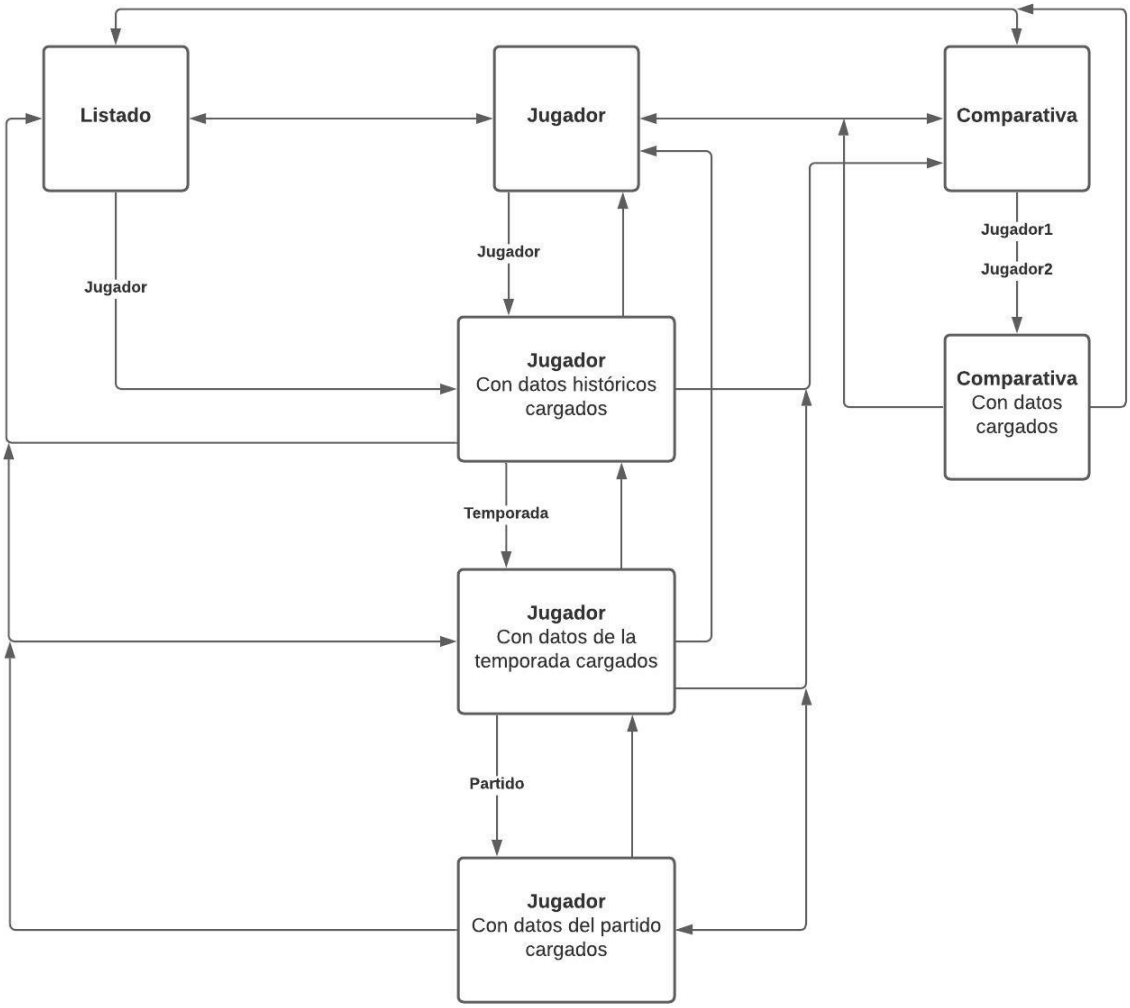

Figura 66: Esquema de navegación

Donde comenzamos en la página principal que es el apartado de *Jugador*, desde la que podemos ir tanto a *Listado* como a *Comparativa*, gracias a la barra de navegación. Esto sucede de igual manera para todas las páginas de la interfaz web. Cabe destacar el flujo que se produce en el apartado *Jugador* en el cual podemos ir avanzando según se van introduciendo nuevos datos del jugador. Además de poder acceder desde *Listado* a las estadísticas históricas ya cargadas de un jugador.

6

Conclusiones y Líneas Futuras

Con este capítulo acabamos la memoria de este trabajo fin de grado. Aquí se describirán las prinicpales conclusiones que se pueden extraer del desarrollo de este trabajo y qué aspectos pueden ser añadidos o mejorados en posteriores versiones del mismo.

6.1. Conclusiones

En primer lugar, se tenía como objetivo crear un sistema de información para almacenar de forma eficiente toda la información recogida, definir unas métricas cuantificables que nos permitan dar una visión precisa del rendimiento de un jugador de fútbol y crear un sistema web con una interfaz intuitiva y visual que nos permita interaccionar con el sistema.

Este proyecto surge de la fiebre de los datos que están experimentando todos los deportes, en especial el fútbol, donde se analiza minuciosamente cada acción, cada detalle y cada centímetro recorrido en el campo. Un partido de fútbol puede generar una inmensa cantidad de datos, con los cuales se pueden realizar un extenso análisis de datos y conseguir sacar información relevante en cuanto al rendimiento a nivel individual de un futbolista en concreto y también a nivel colectivo acogiendo datos del equipo al completo.

Por lo tanto, este Trabajo Fin de Grado ofrece una herramienta que dispone de las funcionalidades tanto de almacenaje de datos generados durante los partidos, así como también poder visualizar de manera intuitiva y clara los análisis generados. Esto nos permite extraer información y conclusiones sobre el rendimiento producido por un jugador y poder compararlo tanto con sus datos históricos como con datos extraídos de una temporada en concreto.

De esta forma se podrá evaluar si el rendimiento en los partidos han sido superiores o inferiores a la media registrada.

Para concluir con el trabajo, he de decir que se han cumplido con los objetivos inicialmente propuestos, ha sido un gran reto para mí desarrollar e implementar todo el sistema de información a través de la fuente de datos proporcionada por StatsBombs[20]. Realizar tanto el tratamiento de datos como la insercción de los datos dentro de la base de datos, ha resultado un gran esfuerzo debido a la gran cantidad datos, obteniendo así tablas con casi tres millones de filas almacenadas, lo cual presentaba un problema a la hora de realizar consultas a dicha tabla.

He puesto en prácticas muchos de los conocimientos adquiridos durante la carrera como son la creación y la gestión de una base de datos, así como también conceptos propios de la asignatura *Introducción a la Ingeniería de Software* como el uso de una metodología de trabajo ágil. El desarrollo de un servidor y una página web de cero, tal y como se enseña en la asignatura de *Ingeniería Web* o tecnologías referentes al frontend que se imparten en *Interfaces de Usuario*, aunque también he utilizado otras cosas donde tenía menos experiencia como el uso de un *ORM* o un lenguaje de programación como *Python*[3], del cual he aprendido mucho durante este proyecto.

En líneas generales, he conseguido completar el desarrollo con éxito, obtener un gran resultado y haber aprendido muchas de diferentes tecnologías que me pueden ser útil el día de mañana.

6.2. Líneas Futuras

Este proyecto presenta muchas posibilidades de ampliación tanto en la parte del sistema de información como en la interfaz web. Por ejemplo, dentro de la fuente de datos obtenida tenemos almacenados datos como los estadios, los árbitros o los managers, los cuáles podrían realizar un análisis de como han podido afectar o no al rendimiento de un jugador.

Otra posible ampliación podría ser realizar un análisis más profundo en cuanto al tipo de posición que ocupa un jugador dentro del campo y ver como podría influir en sus estadísticas jugar en una determinada posición o en otra. A raíz de esta ampliación, también se podría implementar dentro de la interfaz web, gráficas comparativas para las posiciones de los jugadores, ya que actualmente dispone el trabajo, está centrada en métricas más generalizadas y no tan específicas como podrían ser para cada posición y la fuente de datos lo proporciona ese nivel de detalle.

Siguiendo en la línea de la interfaz web, se podría realizar una representación gráfica de los pases realizados tanto por un jugador en concreto como por un equipo ya sea de un partido en concreto o en una temporada completa, lo cual puede ser una información bastante interesante. Todas las mejoras mencionadas anteriormente en la parte de la interfaz web pueden ser añadida al apartado de la comparativa entre dos jugadores.

Referencias

[1] OptaSports *Data Source*. OptaSports. [Online] Disponible: https://www.optasports.com/services/data-feeds/

[2] MySQL *Database* . [Online] Disponible: https://www.mysql.com/

[3] Python *Programming Language*. [Online] Disponible: https://www.python.org/

[4] Flask *Framework*. [Online] Disponible: https://flask.palletsprojects.com/

[5] SQLAlchemy *Object Relational Mapper*. [Online] Disponible: https://pub.dev/packages/sqflite

[6] apache *HTTP Server*. [Online] Disponible: https://apache.org/

[7] XAMPP *Cross-platform Web Server*. [Online] Disponible: https://xampp.en.lo4d.com/windows

[8] PHPMyAdmin *Cross-platform Web Server*. [Online] Disponible: https://www.phpmyadmin.net/

[9] Matplotlib *Graph package*. [Online] Disponible: https://matplotlib.org/

[10] HTML *Web Language*. [Online] Disponible: https://www.w3schools.com/html/

[11] Bootstrap *CSS Framework*. [Online] Disponible: https://getbootstrap.com/

[12] CSS *Style Language*. [Online] Disponible: https://www.w3schools.com/Css/

[13] Javascript *Programming Language*. [Online] Disponible: https://www.javascript.com/

[14] Visual Studio Code. *Code Editor*. [Online] Disponible: https://code.visualstudio.com

[15] Git. *Herramienta de Control de Versiones*. [Online] Disponible: https://git-scm.com

[16] GitHub. *Repositorio de Código Online*. [Online] Disponible: https://github.com

[17] Microsoft Teams. *Aplicación de Reuniones.* [Online] Disponible: https://www.microsoft.com/es-es/microsoft-teams/group-chat-software

[18] LaTeX. *Sistema de Composición de Textos.* [Online] Disponible: https://www.latex-project.org

[19] Overleaf *LaTeX Code Editor.* [Online] Disponible: https://www.overleaf.com/

[20] StatsBomb *Data Source.* [Online] Disponible: https://statsbomb.com/es/

[21] QuickMockup *Interface Design Tool.* [Online] Disponible: https://jdittrich.github.io/quickMockup/

[22] LucidChart *Modeling Tool.* [Online] Disponible: https://www.lucidchart.com/

Apéndice A
Manual de Usuario

Vamos a comenzar con la página principal siendo esta la denominada como *Jugador*. En ella podemos observar que en la parte superior se encuentra la barra de navegación, la cual es un elemento común y fijo en todos los apartados de la interfaz web. A este apartado podemos llegar pinchando siempre sobre *Jugador* en la barra de navegación.

A.1. Barra de navegación y búsqueda de jugador

Figura 67: Explicación barra de navegación y barra de búsqueda

Como podemos observar en la figura, la barra de navegación dispone 3 botones de navegación y si pulsamos:

- **Listado** desplegara el apartado del listado de jugadores de la interfaz.
- **Jugador** mostrará la página principal de la interfaz, tal como se ve en la figura anterior.
- **Comparativa** ofrecerá el apartado de la comparativa entre 2 jugadores que dispone la interfaz.

Por otro lado, podemos ver la barra de búsqueda por nombre de un jugador donde debemos introducir el nombre del jugador que se requiere. Puede darse dos casos, uno donde no obtenemos ningún resultado y nos aparecerá un mensaje informando de que no se han obtenido

resultados y lo volvamos a intentar. El otro caso sería el caso con éxito, donde nos muestra los datos del jugador de la siguiente manera:

Figura 68: Seleccionar temporada

Si pinchamos en el select de *Temporada* se nos mostrará una lista con las temporadas disputadas por el jugador, como podemos apreciar a continuación:

Figura 69: Elección de temporada

Una vez elegimos la temporada, se muestran los datos de la temporada del jugador, además de la gráfica comparativa y se cargan los partidos disputados dentro del select de partido.

Figura 70: Selección de partido

Si pinchamos en el select de *Partido* nos aparecerá una selección de la siguiente forma:

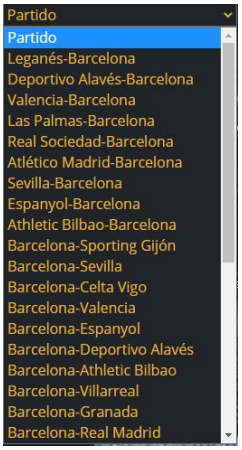

Figura 71: Selección de partido

Por último, se añade una columna más a la tabla con los datos del partido, además de añadir a la gráfica comparativa dichos datos como podemos observar:

Figura 72: Datos partidos cargados

Y en la parte inferior se despliegan todas las gráficas con información sobre el jugador.

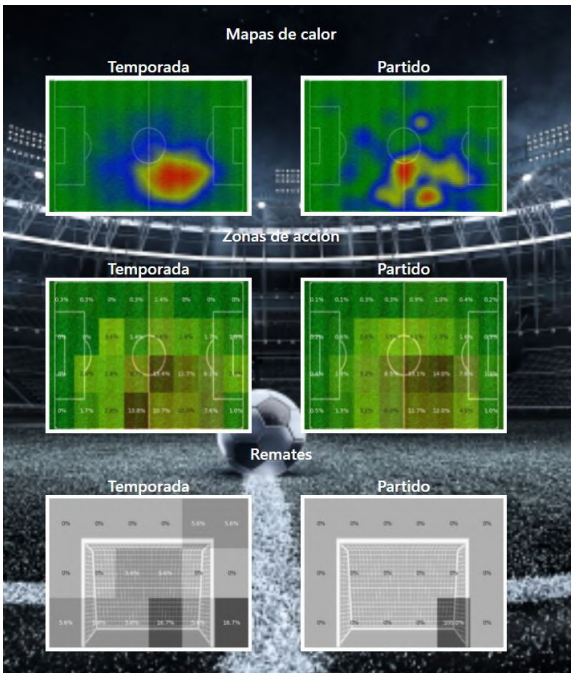

Figura 73: Datos partidos cargados

A.2. Listado

Si pulsamos sobre *Listado* en la barra de navegación accederemos al siguiente apartado de la interfaz:

Figura 74: Listado

Si ponemos el ratón encima de la foto de un jugador nos aparecerá el nombre de esta manera:

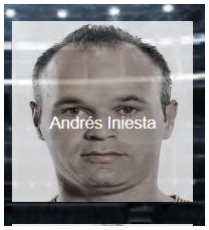

Figura 75: Listado jugador preseleccionado

Si decidimos pinchar sobre la foto de un jugador en cuestión se nos mostrará la misma página que en la figura 68

A.2.1. Comparativa

Si pulsamos sobre *Comparativa* en la barra de navegación accederemos al siguiente apartado de la interfaz, donde se deberá introducir los dos nombres de los jugadores que se pretende generar la comparación y pulsar el botón *Buscar* o pulsar la tecla *Enter*:

Figura 76: Comparativa

Una vez cargado se desplegará de la siguiente forma:

Figura 77: Comparativa cargada

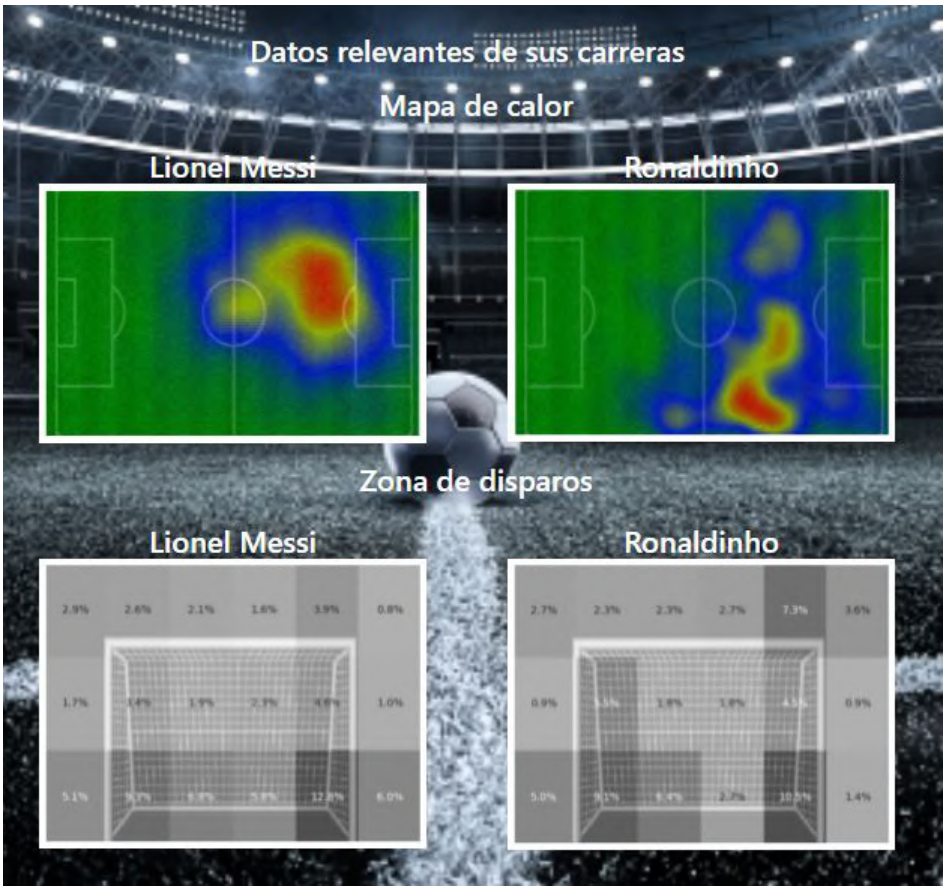

Figura 78: Comparativa cargada

Apéndice B
Manual de despliegue

Para desplegar el proyecto tendremos que acceder a https://github.com/Joslu12/StatTFG y clonar todo el repositorio. Los siguientes pasos a realizar son:

1. Descomprimir las partes del script de la base de datos, que se tuvo que dividir para subir a Github debido a su tamaño.
2. Importar el script de la base de datos al sistema de gestión de base de datos.
3. Abrir la carpeta del proyecto en un editor de código, como Visual Studio[14].
4. Instalar todas las dependencias que se encuentran en el archivo *requirements.txt* usando en la terminal el comando *pip install -r requirements.txt*
5. Instalar *XAMPP*[7] si no se dispone del software o uno similar.
6. Instalar *PHPMyAdmin*[8] o similar para visualizar el contenido web, si no se dispone de un programa que realice dicha función.
7. Iniciar el servidor utilizando programas como *XAMPP*[7] e iniciar los servidores de *Apache*[6] y *MySQL*[2].
8. Inicializar el proyecto ejecutando en un terminal el comando *py main.py*

UNIVERSIDAD DE MÁLAGA | uma.es

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga

E.T.S. DE INGENIERÍA INFORMÁTICA