Universidades de Burgos, León y Valladolid

Máster universitario

Inteligencia de Negocio y Big Data en Entornos Seguros

TFM del Máster Inteligencia de Negocio y Big Data en Entornos Seguros

Búsqueda eficiente de similaridad aplicada al scouting deportivo

Presentado por Cristian Sánchez Hernández
en Universidad de Valladolid — 24 de febrero de 2023
Tutor: José Francisco Díez Pastor

Resumen

En este tabajo fin de máster se aplicarán conceptos orientados a la Inteligencia de Negocio y al Big Data, mediante el desarrollo de una aplicación orientada al ámbito del scouting deportivo, concretamente en el mundo del fútbol. Aplicando el concepto matemático de similaridad, se buscarán los jugadores que más se parezcan a los que ya tenemos en el equipo y que consideramos que están a un gran nivel deportivo, con los objetivos de crear un equipo más competitivo, tener jugadores de recambio que mantengan el mismo nivel de juego y poder suplir las bajas de los jugadores actuales, en caso de lesión o venta del jugador. Además, con el objetivo de aprender a utilizar algunas herramientas, esta búsqueda de la similaridad se realizará mediante la librería Facebook AI Similarity Search (FAISS). Para el almacenamiento de los datos,la aplicación utilizará una base de datos NoSQL orientada a Big Data, como es MongoDB. En cuanto a la infraestructura utilizada para el Big Data, la aplicación se desplegará en Docker, aprovechando la capacidad de abstracción y de virtualización que ofrece este sistema.

Descriptores

Big Data, Inteligencia de Negocio, scouting deportivo, similaridad, FAISS, NoSQL, MongoDB, virtualización, Docker

Abstract

In this master's thesis we will apply concepts oriented to Business Intelligence and Big Data, by developing an application oriented to the field of sports scouting, specifically in the world of football. Applying the mathematical concept of similarity, we will search for players who are most similar to those we already have in the team and who we consider to be at a high level of sport, with the objectives of creating a more competitive team, having replacement players who maintain the same level of play and being able to replace the casualties of current players, in case of injury or sale of the player. Furthermore, with the aim of learning to use some tools, this search for similarity will be carried out using the Facebook AI Similarity Search (Faiss) library. For data storage, the application will use a NoSQL database oriented to Big Data, such as MongoDB. As for the infrastructure used for Big Data, the application will be deployed in Docker, taking advantage of the abstraction and virtualisation capacity offered by this system.

Keywords

Big Data, Business Intelligence, sports scouting, similarity, FAISS, NoSQL, MongoDB, virtualisation, Docker

Índice general

Índice de figuras

Índice de tablas

Memoria

Introducción

El Big Data se ha convertido en un elemento clave en el mundo del deporte, y específicamente en el scouting deportivo. La cantidad masiva de datos generados en cada partido, entrenamiento y evento deportivo puede ser analizada y utilizada para mejorar la toma de decisiones en cuanto a la selección de jugadores y la planificación de estrategias.

Al combinar datos de diferentes fuentes, como el rendimiento en el campo, la condición física, la táctica y la psicología, los equipos pueden tener una comprensión más profunda y precisa de sus jugadores y del juego en general. Con la ayuda del Big Data, los equipos pueden tener una visión más clara y precisa del rendimiento y el potencial de los jugadores, lo que les permite tener herramientas que les sirvan de soporte en al toma de decisiones importantes, como puede ser realizar el fichaje de nuevos jugadores para su equipo, esperando el máximo rendimiento de los mismos.

En este marco se ha desarrollado este proyecto fin de máster, con el objetivo de crear un sistema que sirva como apoyo en la toma de decisiones a la hora de valorar un posible fichaje de un jugador de fútbol para el equipo, con los objetivos de poder crear un equipo más competitivo, buscar jugadores para tener cambios que aporten el mismo nivel de juego o, en caso de venta de un jugador, poder fichar a otro para cubrir su baja y que el rendimiento del equipo no se vea afectado.

La idea de este sistema se ha basado en el principio de similaridad, a la hora de poder encontrar elementos similares en un conjunto , donde en este caso, los elementos de ese conjunto serían los jugadores y la similaridad se buscará entre ellos. Para realizar esta búsqueda de similaridad, se ha utilizado la librería de FAISS (Facebook AI Similarity Search), creada por META (Facebook) con el objetivo de comparar documentos multimedia, sobre la

que además, se quiere comprobar si puede ser eficiente en la búsqueda de similaridad sobre otros elementos distintos para los que fue inicialmente diseñada, como en este caso, serían personas identificadas por una serie de dimensiones relacionadas con el rendimiento deportivo.

Para poder llevar a cabo el objetivo de este trabajo, se ha implementado un sistema basado en unos servicios web que recibirán las peticiones de los jugadores sobre los que se quiere buscar esa similaridad y devolverán, después de calcularla, el resultado de la misma con los jugares más similares encontrados. Para el almacenamiento de los datos de los jugadores que conforman el conjunto, se ha utilizado una base de datos NoSQL como es MongoDB. Además, para poder visualizar los resultados se ha desarrollado una aplicación web que permitirá al usuario interaccionar con los servicios web de manera transparente e intuitiva a la hora de poder obtener y visualizar la información deseada.

El contenido del resto de la memoria se estructura de la siguiente manera:

Objetivos del proyecto, donde se explicarán los objetivos que se pretenden conseguir con la realización de este trabajo.

Conceptos teóricos, apartado donde se profundizará en los conceptos teóricos utilizados para el desarrollo del proyecto.

Técnicas y herramientas, donde se expondrán las herramientas y técnicas usadas para el desarrollo del proyecto.

Aspectos relevantes del desarrollo, donde se incidirá en los aspectos más importantes durante el proceso de desarrollo.

Conclusiones y líneas de trabajo futuras, donde se expondrán las conclusiones a las que se ha llegado después del desarrollo del proyecto, así como futuras líneas de desarrollo y posibles ampliaciones del sistema desarrollado.

Además, se incluyen los siguientes anexos:

Plan de Proyecto Software, donde se expone el plan y la viabilidad del proyecto.

Especificación de diseño, donde se incidirán los aspectos relacionados con el diseño del sistema implementado en este desarrollo.

Documentación técnica de programación, donde se indican las soluciones adoptadas a nivel técnico y de programación para el desarrollo de los diferentes componentes que se engloban dentro de este trabajo fin de máster.

Documentación de usuario, donde se indican las instrucciones y se detalla la funcionalidad de la aplicación a nivel usuario para que una persona pueda utilizarla.

Objetivos del proyecto

En este apartado se explican los objetivos que se quieren alcanzar con la realización de este trabajo fin de máster, tanto desde el punto de vista del producto a construir como desde una perspectiva personal. Los objetivos concretos se especifican a continuación:

2.1. Objetivos del software

El objetivo del desarrollo de esta aplicación es conseguir un sistema que nos dé soporte en la toma de decisiones en un ámbito de negocio, como en este caso es el afrontar el mejor fichaje posible de un jugador de fútbol con la idea de que dé el mejor rendimiento esperado dentro del equipo.

Mediante este software, partiremos de los jugadores que ya tenemos en nuestro equipo o jugadores conocidos que consideremos que tienen un buen rendimiento. A partir de estos datos, el objetivo es encontrar otros jugadores que nos puedan dar un rendimiento similar a nivel deportivo, por ejemplo, para tener un jugador reserva que cuando participe dé el mismo rendimiento que el jugador titular o, en caso de vender un jugador con un rendimiento muy bueno, poder encontrar otro que aporte el mismo valor al equipo.

La aplicación nos permitirá filtrar por distintas características deportivas de los jugadores a la hora de buscar el jugador más acorde a nuestras necesidades, de manera que nos encaje dentro de nuestra perspectiva y de nuestro modelo de negocio.

2.2. Objetivos personales

Entre los objetivos personales que se quieren conseguir con la realización de este trabajo, se encuentra el de poder acercarme a la realidad de desarrollar un proyecto de Big Data en todas sus fases, desde la obtención de los datos hasta la visualización de los mismos. Además, surge el interés de aprender a utilizar algunas tecnologías, sobre las que antes de realizar este proyecto no tengo un gran conocimiento, como pueden ser el uso en un entorno real de una base de datos NoSQL o el despliegue de los servicios en una infraestructura Big Data. Finalmente, otro de los objetivos es poder trabajar con la biblioteca FAISS, a la hora de realizar una búsqueda de similaridad sobre elementos para los que inicialmente no estaba diseñada, y poder probar si se puede utilizar para otros fines.

Conceptos teóricos

En este apartado se explicarán los conceptos teóricos utilizados en el desarrollo de este proyecto para una mejor comprensión.

3.1. Espacio Euclídeo

Un espacio euclídeo $n-dimensional$ denotado por $\mathbb{R}^n$ está compuesto por el conjunto de tuplas ordenadas de dimensión n de números reales [37]:

$$\mathbb{R}^n = \{(x_1, ..., x_n) | x_1, ..., x_n \in \mathbb{R}^n\}$$

Es decir, un punto del espacio se puede representar por un vector de n números reales. Algunos de los espacios euclídeos más comunes son los de dimensión uno (que sería una recta), los de dimensión dos (que sería un plano) o los de dimensión tres (que sería un espacio tridimensional, como por ejemplo ancho x largo x alto).

3.2. Definición de medida de distancia

Si tenemos un conjunto de puntos en un espacio, la medida de distancia entre dos puntos A y B es una función $d(A, B)$ cuyo resultado satisface los siguientes axiomas [2]:

1. $d(A, B) \geq 0$ (La distancia no es negativa)
2. $d(A, B) = 0$ si y solo si $A = B$ (La distancia es siempre positiva, excepto si se mide desde un punto consigo mismo)

3. $d(A, B) = d(B, A)$ (La distancia es simétrica)

4. $d(A, B) \leq d(A, C) + d(C, B)$ (Desigualdad triangular [36])

3.3. Distancia Euclídea

La distancia euclídea está basada en la localización de los puntos en el espacio. Dentro del ámbito del trabajo de este máster, podemos considerar que un jugador puede ser representado como un punto de un espacio euclídeo, caracterizado por sus atributos más importantes como pueden ser sus estadísticas deportivas (pases, goles, resistencia, etc). Dentro de las distintas distancias euclídeas entre dos puntos, la más común es la que se deduce del teorema de Pitágoras, denotada por *norma* $\mathbb{L}_2$, y se calcula mediante la raíz cuadrada de la suma de los cuadrados de la distancia entre los puntos medidos en cada dimensión [2]:

$$d([x_1, x_2, ..., x_n], [y_1, y_2, ..., y_n]) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

3.4. Relación entre similaridad y distancia euclídea

Es habitual hablar de similaridad basada en indicadores de distancia y viceversa. Una vez definida la distancia y cómo se calcula, vamos a revisar algunas de sus propiedades y compararlas con las de la similaridad.

Si tenemos una distancia $d(A, B)$, algunas de las propiedades de la misma son:

1. El valor de la distancia d entre los objetos A y B estará comprendido entre cero e infinito.

2. Si los objetos A y B están muy cerca, la distancia d tendrá un valor muy pequeño.

3. Por el contrario, si los objetos A y B están muy alejados entre sí, la distancia d tendrá un valor muy grande.

4. Si el objeto A y el objeto B son el mismo, el valor de la distancia d entre ellos será cero.

En cuanto a las propiedades de una similaridad $s(A, B)$ se definirían de la siguiente forma:

1. El valor de la similaridad s entre los objetos A y B estará comprendido entre cero y uno.
2. Si los objetos A y B son muy semejantes, la similaridad s tendrá un valor muy grande (cercano a uno)
3. Por el contrario, si los objetos A y B no son muy semejantes, la similaridad s tendrá un valor muy pequeño (cercano a cero).
4. Si el objeto A y el objeto B son el mismo, el valor de la similaridad s será igual a uno.

Atendiendo a las propiedades mencionadas se puede apreciar que existe una relación entre ambos conceptos, y que obteniendo uno de ellos se puede crear una relación con el otro. Por ejemplo, una formúla matemática que cumple con los axiomas previamente indicados de medida de distancia y las propiedades anteriores que se suele usar es la siguiente [30]:

$$s(A, B) = \frac{1}{1+d(A,B)}$$

Técnicas y herramientas

En esta sección se mencionarán las tecnologías empleadas para el desarrollo del proyecto, así como las herramientas utilizadas en el mismo.

4.1. FAISS

FAISS (Facebook AI Similarity Search) es una biblioteca desarrollada inicialmente por Facebook, con el objetivo de permitir buscar elementos similares entre documentos multimedia de manera eficiente [19][3]. FAISS resuelve algunas limitaciones a la hora de realizar la búsqueda de elementos similares sobre otros métodos de búsqueda basados en HASH.

La representación de los elementos entre los que se desea buscar esa similaridad se hace a través de una representación vectorial, donde se considerarían como vectores similares aquellos que están cerca en el espacio euclidiano, es decir, aquellos vectores cuya distancia euclídea es menor.

Los métodos tradicionales no son eficientes en cuanto a tiempo de ejecución y uso de memoria a la hora de realizar una búsqueda de elementos similares o están muy limitados en cuanto a escalabilidad. La biblioteca de FAISS está optimizada en cuanto al uso de memoria y la velocidad en la que procesa este cálculo, dando un mejor rendimiento a la hora de realizar este tipo de operaciones.

Dados un conjunto de vectores x_i de dimensión d, FAISS crea una estructura de datos en memoria RAM con estos vectores, la cuál tratará como un índice.

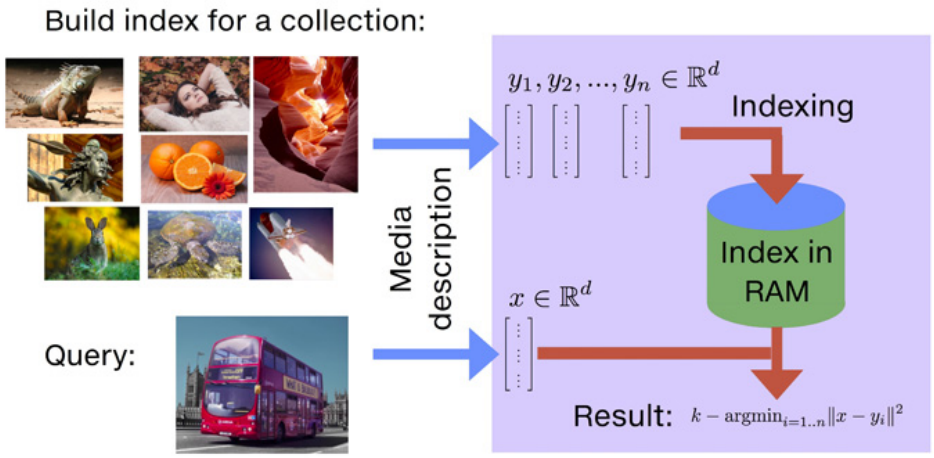


Figura 4.1: Representación del tratamiento de los vectores por FAISS [3]

Dado un nuevo vector x de dimension d, FAISS puede calcular la distancia euclídea *norma* $\mathbb{L}_2$ mínima entre el vector x y el conjunto de vectores x_i, y devolver el índice del elemento que cumple con esa distancia mínima. Además, también puede devolver no sólo el primer elemento más cercano si no los $k - elementos$ más cercanos.

De esta forma FAISS es capaz de calcular la distancia entre los distintos elementos de una manera eficiente en cuanto a tiempo y recursos.

4.2. MongoDB

En cuanto al almacenamiento de los datos, se presentan varias opciones a la hora de guardar la información que utilizaremos en la aplicación.

Dado que la aplicación desarrollada está orientada a trabajar con un gran volumen de datos y se prevé que los datos vayan aumentando progresivamente, además de ser necesario que tenga una gran escalabilidad, se ha optado por utilizar una base de datos NoSQL frente al modelo de base de datos relacional.

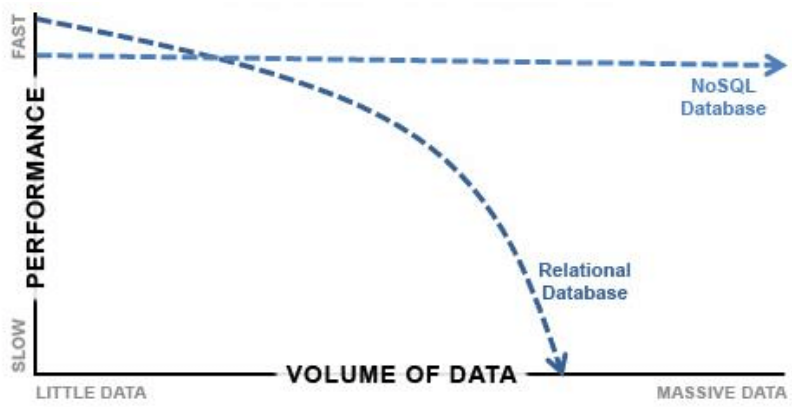


Figura 4.2: Escalabilidad bases de datos relacionales y NoSQL [1]

Dentro de las bases de datos NoSQL nos encontramos con distintos tipos. Atendiendo al modelo de almacenamiento de los datos, las bases NoSQL se pueden clasificar en los siguientes tipos [4] [28]:

Clave-Valor Cada elemento se almacena en la base de datos con un par clave-valor. En cierta forma, este tipo de almacenamiento se asemeja al almacenamiento en una tabla de dos columnas de una base de datos relacional, en la que un campo sería la clave o el nombre del atributo y el otro el valor asociado. De manera habitual, la clave suele ser un string y el valor suele ser un string o incluso algún dato más complejo, como listas, imágenes o documentos. Estas bases de datos son muy escalables y tienen un modelo de datos muy simple. Por otro lado, este último punto hace que no se pueden modelar estructuras muy complejas de datos ni realizar querys de una alto nivel de complejidad. Algunos ejemplos de este tipo de bases de datos son Dynamo DB, Voldemort o Riak.

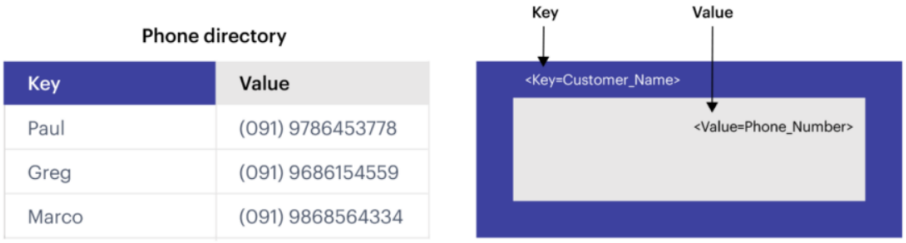


Figura 4.3: Ejemplo de almacenamiento en una base de datos NOSQL clave-valor [29]

Orientadas a documentos En este tipo de bases de datos la información se almacena en pares de clave-valor, donde el valor es un documento con un formato predefinido, como por ejemplo JSON, BSON, XML, etc. que a su vez contiene colecciones de pares claves-valor. Como ventajas, destacan que son altamente escalables, la curva de aprendizaje es pequeña y están orientadas al uso en aplicaciones web o móviles. Algunos ejemplos de este tipo de bases de datos son Mongo DB o Couch DB.

Key	Document
1001	{ "CustomerID": 99, "OrderItems": [{ "ProductID": 2010, "Quantity": 2, "Cost": 520 }, { "ProductID": 4365, "Quantity": 1, "Cost": 18 }], "OrderDate": "04/01/2017" }
1002	{ "CustomerID": 220, "OrderItems": [{ "ProductID": 1285, "Quantity": 1, "Cost": 120 }], "OrderDate": "05/08/2017" }

Figura 4.4: Ejemplo de almacenamiento en una base de datos NOSQL orientada a documentos [24]

Orientadas a columnas En el modelo de almacenamiento relacional, los elementos se almacenan orientados a filas. En este modelo de base de

datos NoSQL, los datos que se almacenan se guardan en columnas en lugar de filas y por cada entrada, hay una columna. Los datos de cada entrada no se encuentran en una misma fila, si no que están dispuestos en distintas columnas. Este tipo de bases de datos tiene una gran velocidad de lectura de datos, pero por otro lado, no son eficientes para la escritura de datos o a la hora de recuperar todos los registros almacenados. Algunos ejemplos de estas bases de datos son Cassandra o Hbase.

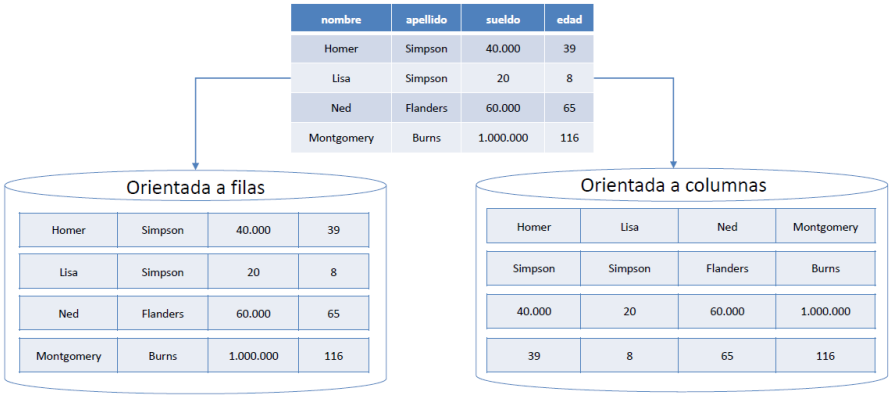


Figura 4.5: Ejemplo de almacenamiento en una base de datos NOSQL orientada a columnas [4]

Orientadas a grafos Este tipo de base de datos NoSQL representa la información almacenada mediante nodos de un grafo y las relaciones entre dichos elementos mediante aristas. Este tipo de base de datos está optimizada a la hora de buscar la relación entre los elementos, superando en rendimiento del uso de múltiples JOINs que se realizarían en una base de datos relacional. En general, en el ámbito de negocio, pocas empresas utilizan únicamente este tipo de base de datos basadas en consultas gráficas y se usan a modo de complemento con otro tipo de base de datos para complementar la funcionalidad, ya que no son tan escalables como las mencionadas en los puntos anteriores. Además, requieren un cambio de concepto importante a la hora del desarrollo con las mismas, con lo que tienen una gran curva de aprendizaje. Algunos ejemplos de funcionalidad de este tipo de bases de datos serían la detección de fraudes o redes sociales. Como ejemplo de este tipo de base de datos sería Neo4j.

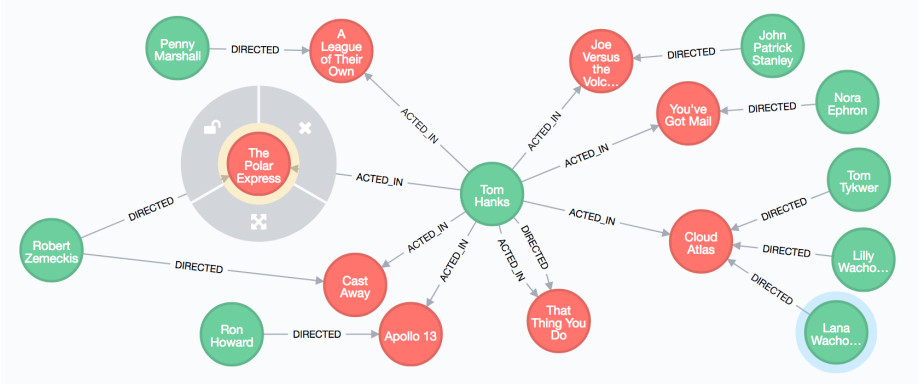


Figura 4.6: Ejemplo de base de datos orientada a grafos [22]

Como se puede comprobar, cada familia de bases de datos NoSQL tiene unas características propias que hay que tener en cuenta a la hora de seleccionar una base de datos de este tipo para el proyecto en concreto a desarrollar.

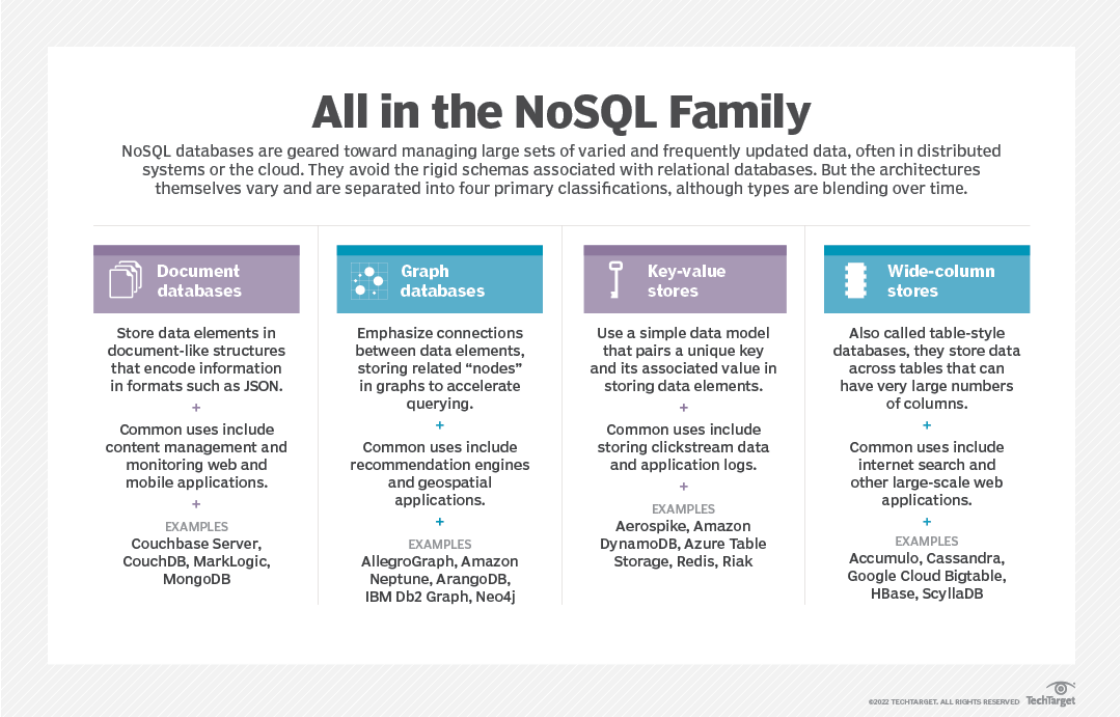


Figura 4.7: Tabla comparativa de familias de bases de datos NoSQL[31]

Después de lo comentado anteriormente, se ha optado por utilizar para el desarrollo de este proyecto MongoDB. Esta base de datos está orientada a documentos y almacena los mismos utilizando un sistema de clave-valor. El modelo de datos que utiliza se basa en colecciones de documentos que pueden ser en distintos formatos(JSON, XML, PDF, ...), ofreciendo una gran flexibilidad a la hora de almacenar los datos así como una gran escalabilidad. Además, cuenta con la ventaja, de que no tiene una gran curva de aprendizaje en comparación con otras bases de datos NoSQL, pudiendo además usar distinto lenguajes de programación para realizar operaciones sobre ella como JavaScript, Java, Python C# o C++. También hay que tener en cuenta que una de las características de este tipo de base de datos es que su uso, entre otros, está orientado a aplicaciones web o móviles como la que se desarrollará en este proyecto. Otra razón para la elección de esta base de datos, ha sido por la infraestructura que se usará en este proyecto para su despliegue, que es Docker, siendo compatible con el mismo. Para finalizar, podemos comprobar que es la base datos NoSQL que más tendencia e interés suscita en los últimos tiempos:

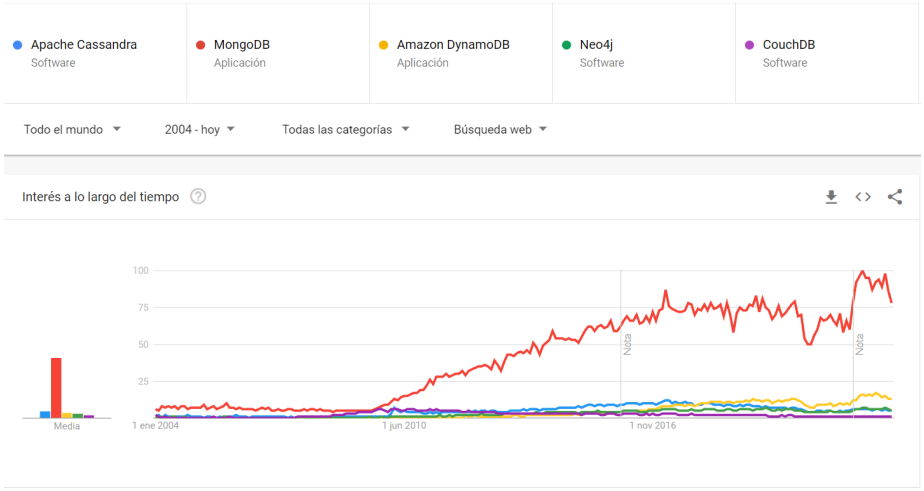


Figura 4.8: Tendencia base de datos NoSQL en Google Trends [33]

4.3. Docker

En cuanto a la infraestructura para el Big Data seleccionada, el despliegue de la aplicación se ha realizado mediante Docker [11].

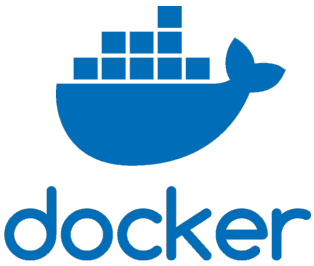


Figura 4.9: Logo de Docker [11]

Docker es una plataforma de código abierto que nos permite desarrollar, distribuir y desplegar aplicaciones haciendo transparente para el usuario o desarrollador la infraestructura sobre la que se está ejecutando la aplicación. Esta característica permite que el desarrollo, las pruebas y el despliegue final del producto se realice más rápidamente y sea más escalable, ya que con un único desarrollo, la aplicación se podrá ejecutar en distintos tipos de máquinas sin que se tenga que adaptar el desarrollo a los requisitos específicos de cada una de ellas.

Para realizar esto, Docker ejecuta las aplicaciones en contenedores. Un contender es la unidad estándar que utiliza Docker para ejecutar la aplicación desarrollada y que contiene el código, las dependencias de la misma, la configuración y todo lo necesario para que se pueda ejecutar.

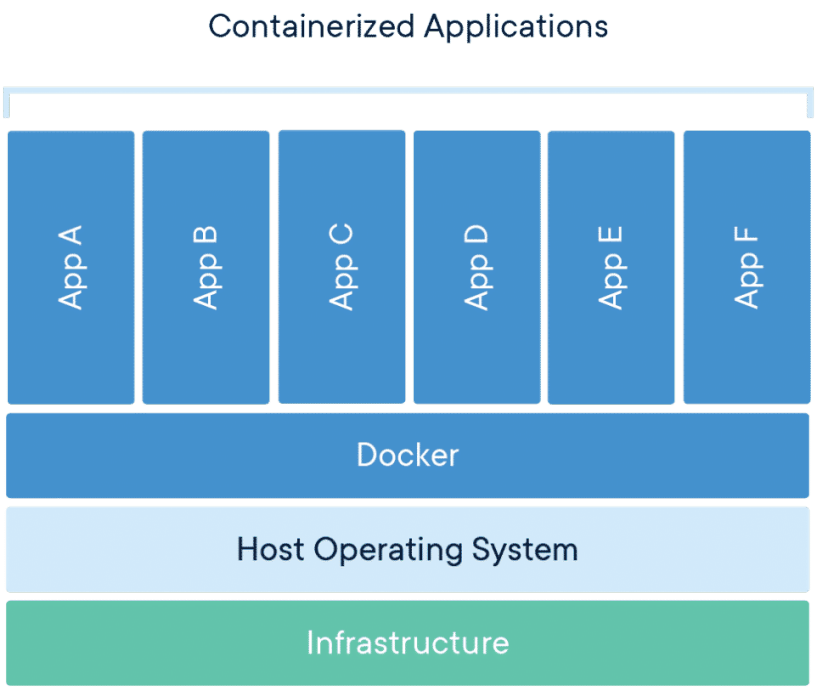


Figura 4.10: Contenedores de Docker [12]

Con esto, la aplicación se podrá ejecutar independientemente de si la máquina sobre la que se ejecuta está basada en Linux, Windows o cualquier otro sistema, ya que el software siempre se ejecutará igual en su contenedor. Los contenedores aíslan el software de su entorno y garantizan que funcione de manera uniforme a pesar de las diferencias entre las distintas infraestructuras.

Docker usa una arquitectura de cliente-servidor. El cliente se comunica a través de una API REST con el "Docker daemon", que es el proceso encargado de crear, ejecutar y distribuir los contenedores, acorde a lo que se le solicite desde el cliente.

Para la creación de los contenedores, Docker se basa en imágenes. Las imágenes son unas plantillas que contienen el código, las definiciones y las dependencias, además de unas instrucciones concretas para crear el

contenedor Docker. Se pueden utilizar imágenes ya creadas o personalizar una imagen a medida mediante el archivo DockerFile. Estas imágenes las podemos encontrar en un "Docker registry", que básicamente es un almacén de imágenes de Docker. Un ejemplo puede ser Docker Hub, que es un registro público de imágenes y que cualquier persona puede utilizar, o por el contrario, también se puede crear un registro privado de imágenes para uso personal.

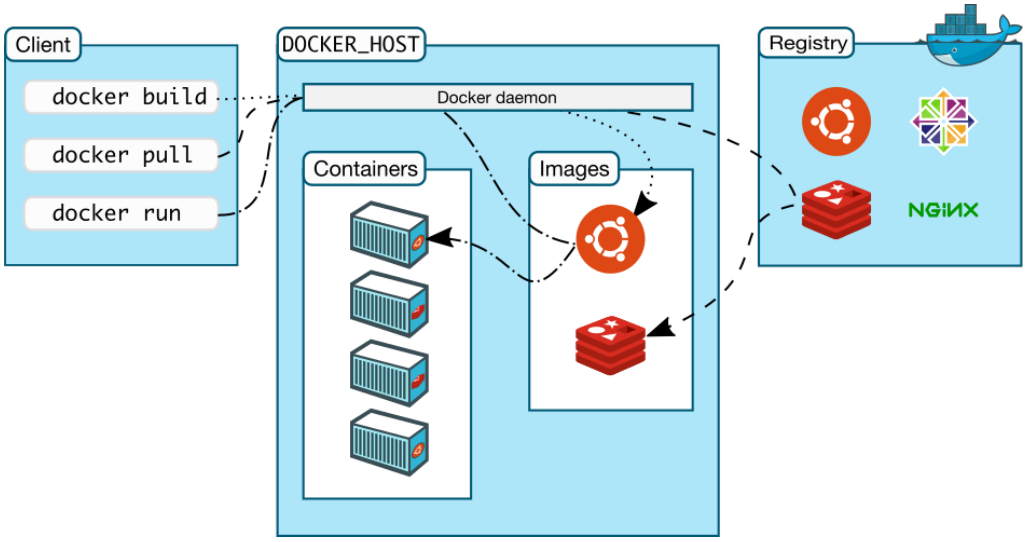


Figura 4.11: Arquitectura de Docker [11]

4.4. Pentaho

A la hora de realizar el proceso de ETL, dentro de las herramientas enfocadas en la obtención y adecuación de los datos, se ha realizado mediante PDI dadas sus características.

Figura 4.12: Logo de Pentaho [18]

Pentaho Data Integration (PDI), también conocida como Kettle (Hitachi Vantara),[35] es una herramienta que nos permite realizar procesos de ETL de manera eficiente y de la que se pueden destacar las siguientes características:

Open Source Se trata de una herramienta de Open Source (de código abierto), con una gran comunidad detrás.

Manejo de gran volumen de datos PDI nos va a permitir realizar acciones o procesos que debido a su volumen y/o complejidad, nos sería realmente complicado o imposible realizar de manera manual.

Flexibilidad Es una herramienta que nos ofrece gran flexibilidad para adaptarla a nuestras necesidades, con un gran conjunto de operaciones a aplicar sobre los datos a tratar.

Intuitiva Tanto las instalación como la configuración es muy sencilla

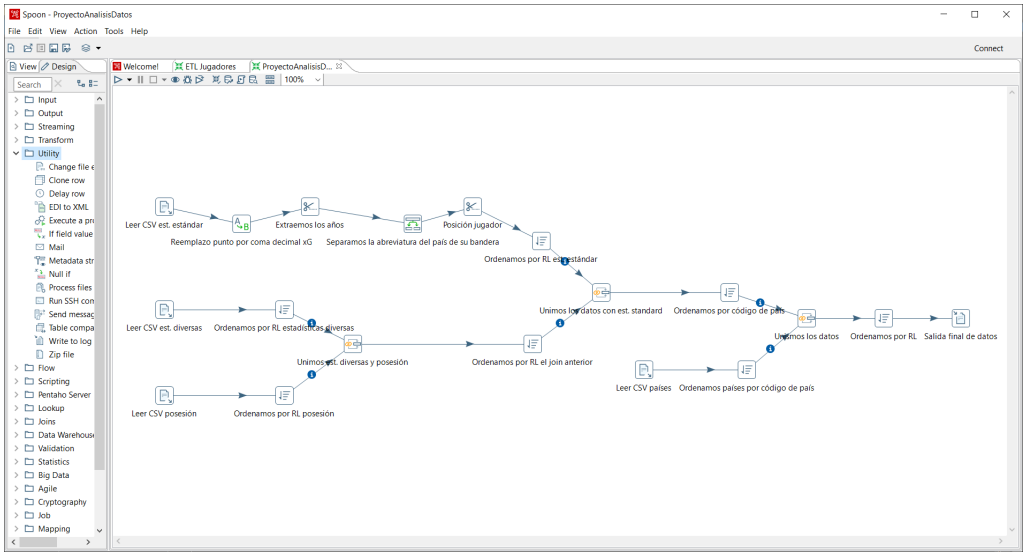


Figura 4.13: Ejemplo del proceso ETL realizado en el flujo de datos con PDI

Con la herramienta PDI de Pentaho se pueden realizar operaciones como sustituciones de valores de campos, utilizar expresiones regulares para crear o modificar campos o unir diferentes archivos CSV de manera fácil e intuitiva, siguiendo el flujo de los datos en todo momento.

Aspectos relevantes del desarrollo del proyecto

En este punto se exponen los aspectos más importantes del desarrollo del proyecto y los detalles de mayor relevancia dentro de las fases de desarrollo.

5.1. Comparativa búsqueda de similaridad

A la hora de realizar la búsqueda de similaridad de manera más eficiente, se ha optado por utilizar la librería de FAISS. Para poder comprobar la diferencia en cuanto al tiempo empleado a la hora de realizar esta búsqueda utilizando FAISS u otros métodos, se ha realizado dicha búsqueda utilizando el mismo juego de datos de jugadores y partiendo de un mismo jugador seleccionado como modelo, utilizando una búsqueda iterativa y realizando esa misma búsqueda con FAISS.

Para realizar la búsqueda como se haría de una manera clásica, mediante un método iterativo, se recorren todos los elementos del juego de datos, calculando la distancia que hay entre cada uno de ellos y el dato elegido como modelo. Finalmente nos quedaremos con los k elementos más similares, es decir, los k elementos que tengan menor distancia entre ellos.

Para ello se ha creado una función que recibe como parámetros el jugador modelo y el número de jugadores más similares que se quieren encontrar. La función devuelve la posición en la que se encuentran el número de jugadores recibidos más similares.

```
def calculaDistanciaNormal(posJugadorModelo, k):

    #Normalizamos el vector de datos
    data_norm = playersNumpy/np.linalg.norm(playersNumpy)

    #Cogemos un jugador sobre el que buscaremos la similaridad
    data_norm_model = data_norm[posJugadorModelo]

    #Calculamos el número de jugadores
    numJugadores = data_norm.shape[0]

    #Array para guardar las distancias encontradas
    distances = []

    # Recorremos con un bucle for todos los jugadores
    for i in range (numJugadores):

        #Calculamos la distancia del jugador modelo con respecto a cada
        ↪  jugador
        dist = np.linalg.norm(data_norm_model - data_norm[i])

        #Almacenamos la distancia en el array de distnacias
        distances.append(dist)

    #Nos quedamos con los k elementos más similares
    kSimilares = np.argsort(distances)[:k]

    return kSimilares
```

De igual forma, se ha creado otra función para realizar la búsqueda con FAISS, la cuál recibe los mismos parámetros de entrada y devuelve los mismos parámetros de salida.

```
def calculaDistanciaFAISS(posJugadorModelo,k):

    #Creamos la consulta a utilizar en FAISS con el jugador referencia
    query = []

    query.append(playersNumpy[posJugadorModelo])
    query = np.asarray(query)

    # Calculamos el número de dimensiones
    dimensiones = playersNumpy.shape[1]

    # Creamos un índice que será del tamaño de las dimensiones
    index = faiss.IndexFlatL2(dimensiones)
```

```
    # Añadimos al índice el conjunto de datos con ascontiguosarray para
    ↪  que funcione de manera correcta
    index.add(np.ascontiguousarray(playersNumpy))

    #Calculamos la similaridad y devolvemos los datos
    D, I = index.search(query,k)

    return I
```

Para validar la diferencia de rendimiento en cuanto al tiempo empleado en la búsqueda por los distintos métodos, se ha realizado la ejecución mediante la herramienta Jupyter Notebook [20], utilizando el comando **%%timeit** [17], el cuál nos devuelve el tiempo empleado para ejecutar el código que se encuentra en esa celda.

Los resultados obtenidos para la búsqueda iterativa han sido de 355 ms ± 40.8 ms en las diferentes iteraciones que se han comprobado:

```
%%timeit

#Calculamos la distancia con respecto al jugador que se encuentra en la posición pasada como parámetro
jug_similares_normal = calculaDistanciaNormal(16628,5)
```

```
355 ms ± 40.8 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

```
jug_similares_normal = calculaDistanciaNormal(16628,5)

print(jug_similares_normal)
```

```
[16628 11353 16297 16097  4094]
```

Figura 5.14: Tiempo empleado realizando la búsqueda de manera iterativa

Los resultados obtenidos realizando la misma búsqueda, pero utilizando FAISS han sido de 7.99 ms ± 174 µs en las distintas iteraciones que se han realizado:

```
%%timeit

#Calculamos la distancia con respecto al jugador que se encuentra en la posición pasada como parámetro
jug_similares_faiss = calculaDistanciaFAISS(16628,5)
```

```
7.99 ms ± 174 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

```
jug_similares_faiss = calculaDistanciaFAISS(16628,5)
print(jug_similares_faiss)
```

```
[[16628 11353 16297 16097  4094]]
```

Figura 5.15: Tiempo empleado realizando la búsqueda con FAISS

Como se pude comprobar, obteniendo el mismo resultado, hay una gran mejora en cuanto al tiempo que se emplea en realizar la búsqueda de similaridad utilizando FAISS con respecto a una búsqueda iterativa.

5.2. Origen de los datos

En cuanto al origen de los datos que serán la base del proyecto, se ha elegido un dataset publicado en la plataforma de Kaggle, concretamente el dataset "FIFA 21 complete player dataset" [21].

Este dataset contiene los datos de los jugadores de fútbol del juego multiplataforma FIFA 21, que salió al mercado en Octubre del año 2020. Contiene en total más de dieciocho mil jugadores con más de cien atributos cada uno. Algunos de estos atributos son el nombre, edad, posición, datos asociados al equipo en el que juega, y valoraciones de distintas áreas, basadas en las estadísticas y rendimiento del jugador en la vida real, como ataque, habilidades, movimientos, resistencia o mentalidad, entre otros.

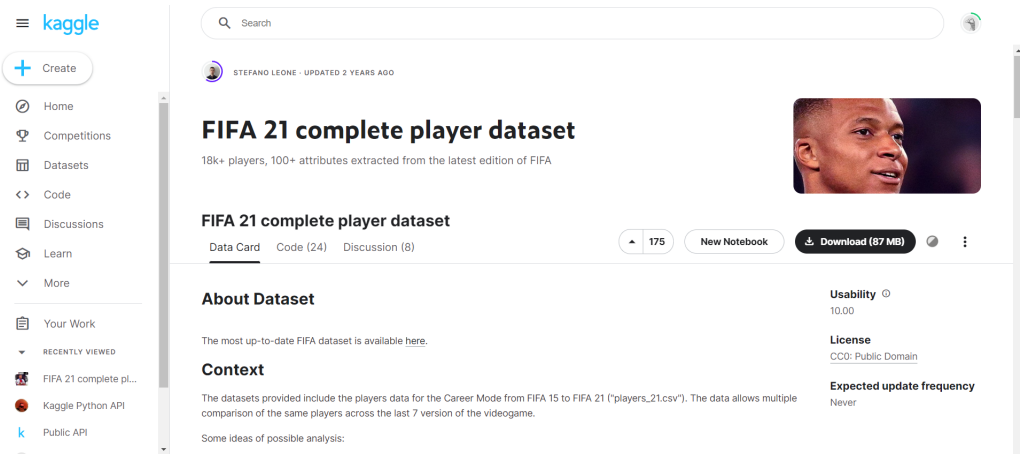


Figura 5.16: Origen de datos en Kaggle [21]

Uno de los motivos por los que se ha elegido este juego de datos es que la valoración de las distintas áreas y estadísticas deportivas del jugador, está realizada en base al rendimiento del propio jugador en la vida real, por tanto nos puede dar una buena imagen del rendimiento del jugador.

Otro de los motivos por los que se ha optado por utilizar estos datos del 2020, ha sido el poder comparar a día de hoy si los resultados obtenidos en el cálculo de la similaridad mediante el sistema desarrollado en este trabajo tienen sentido, pudiendo comparar dichos resultados con los fichajes de los jugadores que se hayan realizado durante este tiempo en la vida real.

5.3. ETL

El proceso de ETL (Extraction, Transformation, Load) de los datos se ha realizado con la herramienta Pentaho.

Se ha partido del dataset comentado en el punto anterior y se han realizado varias operaciones sobre dichos datos.

En primer lugar y partiendo del archivo CSV original con los datos de los jugadores, se han extraído los datos de los equipos de fútbol asociados a los jugadores por dos motivos:

- Se va a crear un nuevo archivo CSV que contendrá el nombre de cada equipo de fútbol y se le asociará un id único a cada uno de ellos. De

esta forma, podremos crear una colección con estos datos en la base de datos y poder usarlos en la aplicación web mediante los webservices.

- Una vez asignado el id, se concatenará este campo de nuevo sobre el dataset inicial para que se pueda tener la relación entre equipo y jugador por el identificador único. De esta forma aseguramos evitar posibles duplicidades de equipos que puedan compartir el mismo nombre.

Además, también se han extraído los países de los jugadores del archivo inicial por el siguiente motivo:

- Se va a crear otro CSV temporal que tenga el nombre del país y su código "ISO 3166-1 alpha-2" [38], el cuál identifica al país con dos letras de manera unívoca. Esto nos servirá para utilizarlo como abreviatura del país de manera única si es necesario y para representar la bandera del país a la hora de visualizar los datos. Para esto, se asociaron los códigos indicados a sus respectivos países y se añadió este campo sobre el dataset original.

En esta etapa también se han eliminado algunas dimensiones del CSV inicial, ya que no aportan valor al objetivo del proyecto, que es buscar la similaridad de los jugadores respecto al rendimiento deportivo, como por ejemplo, el tipo de objeto de animación utilizado en el juego para recrear la imagen del jugador, columnas de dimensiones repetidas o columnas cuyos todos sus valores son nulos.

Además, se han revisado algunas columnas ya que contenían algunos valores nulos. Interpretando la información del juego de datos, los valores de estas dimensiones eran nulos porque están relacionados con jugadores que en ese momento eran libres, es decir, no tenían ningún contrato firmado con ningún equipo.

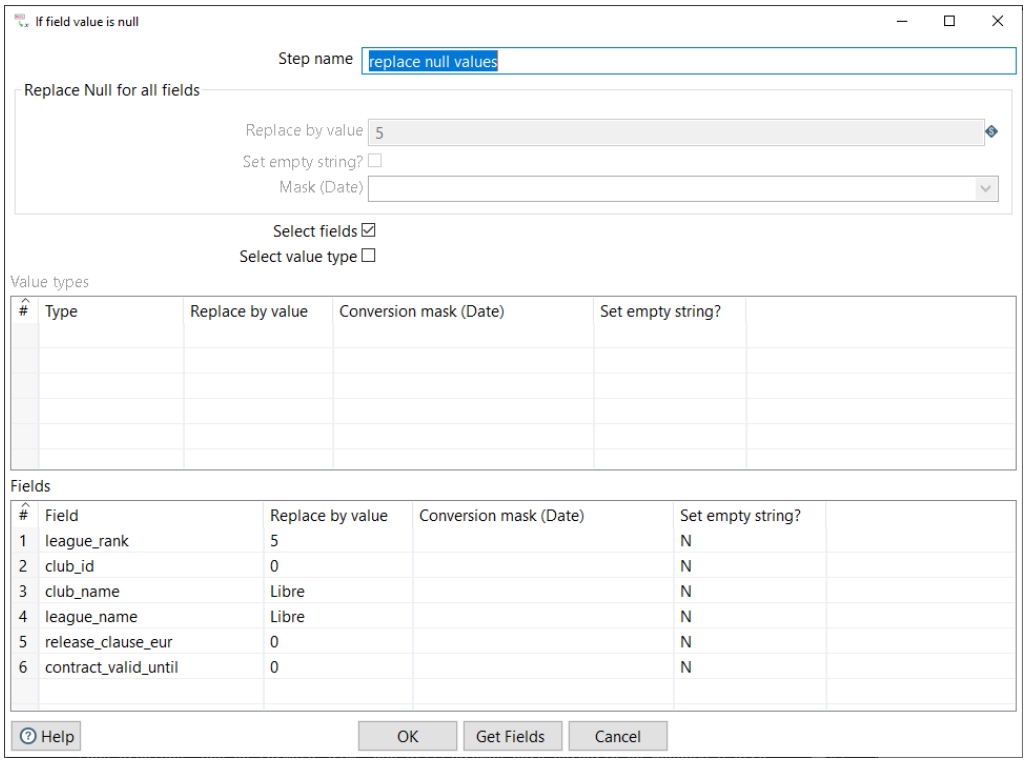


Figura 5.17: Dimensiones con valores nulos reemplazados

Para estas columnas se han cambiado los valores nulos iniciales, atendiendo a la siguiente lógica explicada a continuación:

league_rank Es un número que forma parte de un ránking que se asocia las diferentes ligas en las que participan los jugadores. Inicialmente este ránking comprende los valores del 1 al 4, siendo el 1 el valor que representa el máximo nivel de las ligas y el 4 el menor valor de la liga. Siguiendo esta escala, se ha asociado el número 5 para los jugadores que actualmente no pertenecen a ninguna liga por no tener contrato con ningún equipo.

club_id Es el identificador que se asocia a cada club. Los jugadores libres al no tener contrato con ninguno, no tienen este valor informado. En este caso se ha asociado el ID 0 para identificarlos.

club_name Es el nombre del equipo al que pertenece el jugador. Los jugadores libres no pertenecen a ninguno has que firmen un contrato, por tanto se ha optado porque se informe el literal "Libre".

league_name En la misma línea que el punto anterior, al no estar asociado a ninguna liga se ha informado el literal "Libre".

release_clause_eur Es la cláusula de rescisión de contrato. Para estos casos se ha completado la información con el valor 0.

contract_valid_until Esta dimensión informa el año en el que el jugador termina contrato. Al ser jugadores libres, no tienen firmado ningún contrato con ningún club, por tanto no tienen año informado. Se ha optado por informar este campo con el valor 0, para que en caso de que se hagan búsquedas de jugadores cuyo contrato termine antes de una fecha predeterminada, estos jugadores también puedan a parecer y se puedan tener en cuenta.

Para terminar se han renombrado algunos campos y como salida del proceso se ha generado un archivo en formato CSV que será el que utilizaremos para cargarlo en MongoDB.

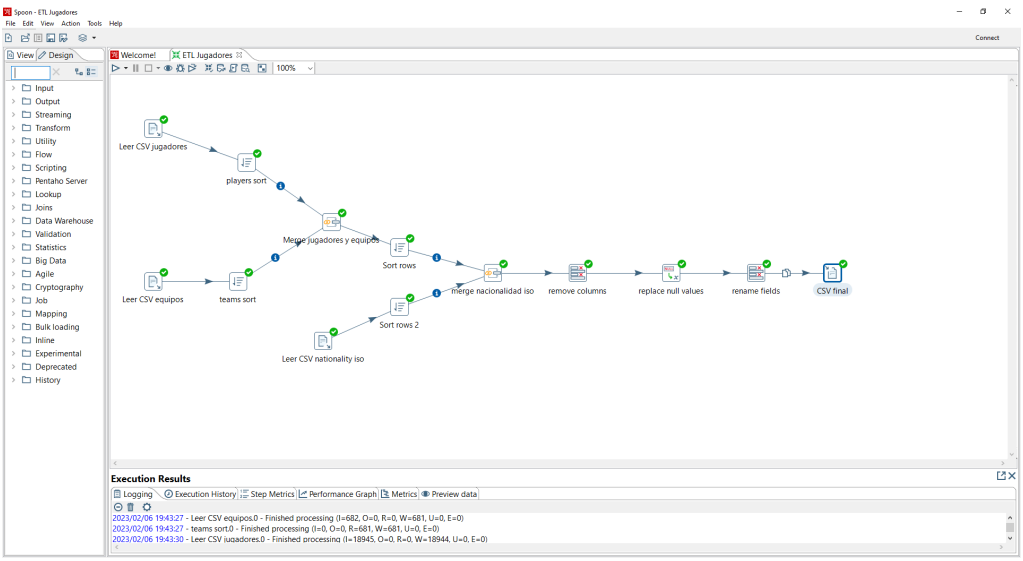


Figura 5.18: Proceso ETL realizado en el desarrollo del proyecto

Finalmente se han recogido 57 dimensiones, las cuáles se detallan a continuación agrupándolas por familias:

- Dimensiones de datos personales del jugador
- Dimensiones datos del equipo y la liga del jugador
- Dimensiones del rendimiento deportivo del jugador, que a su vez se dividen en las siguientes dimensiones:
  - Dimensiones generales del rendimiento deportivo
  - Dimensiones de ataque del rendimiento deportivo
  - Dimensiones de habilidades del rendimiento deportivo
  - Dimensiones de movimiento del rendimiento deportivo
  - Dimensiones de fuerza/resistencia del rendimiento deportivo
  - Dimensiones de mentalidad del rendimiento deportivo
  - Dimensiones de defensa del rendimiento deportivo
  - Dimensiones de habilidades de portero del rendimiento deportivo

Los valores de todas las dimensiones del rendimiento deportivo del jugador oscilan entre cero y cien, estando más cerca del cero si no destacan en esa habilidad y más próximos a cien si destacan en la misma.

El detalle de los campos de cada una de las familias anteriores se detalla a continuación en las tablas siguientes [14] [32]:

Dimensión	Descripción	Tipo de dato
id	identificador único para el jugador	Numérico
short_name	abreviatura del nombre del jugador	Texto
long_name	nombre del jugador	Texto
age	edad del jugador	Numérico
dob	fecha de nacimiento del jugador	Texto
height_cm	altura (en cm) del jugador	Numérico
weight_kg	peso (en kg) del jugador	Numérico
nationality	nacionalidad del jugador	Texto
nationality_iso	código “ISO 3166-1 alpha-2” del país del jugador	Texto
player_postions	posición del jugador en el campo	Texto
preferred_foot	pie preferido del jugador manejando el balón	Texto

Tabla 5.1: Dimensiones datos personales del jugador

Dimensión	Descripción	Tipo de dato
club_id	identificador único para el equipo del jugador	Numérico
club_name	nombre del equipo del jugador	Texto
league_name	nombre de la liga del equipo del jugador	Texto
league_rank	ranking de la liga del equipo del jugador	Numérico

Tabla 5.2: Dimensiones datos del equipo y la liga del jugador

Dimensión	Descripción	Tipo de dato
value_eur	valor (en EUR) del jugador	Numérico
wage_eur	salario (en EUR) del jugador	Numérico
release_clause_eur	claúsula de rescisión del contrato	Numérico
contract_valid_until	año fin del contrato del jugador	Numérico

Tabla 5.3: Dimensiones datos contractuales del jugador

Dimensión	Descripción	Tipo de dato
pace	ritmo de carrera	Numérico
shooting	fuerza y precision general de tiro	Numérico
passing	precisión de pase	Numérico
defending	habilidad para defender	Numérico
physic	estado físico del jugador	Numérico

Tabla 5.4: Dimensiones generales del rendimiento deportivo

Dimensión	Descripción	Tipo de dato
attacking_crossing	centros al área contraria	Numérico
attacking_finishing	finalización de jugada en gol	Numérico
attacking_heading_accuracy	precisión remate de cabeza	Numérico
attacking_short_passing	pase corto en ataque	Numérico
attacking_volleys	ejecución de voleas en ataque	Numérico

Tabla 5.5: Dimensiones de ataque del rendimiento deportivo

Dimensión	Descripción	Tipo de dato
skill_dribbling	regate	Numérico
skill_curve	habilidad de dar efecto al balón	Numérico
skill_fk_accuracy	precisión en el tiro libre	Numérico
skill_long_passing	habilidad para realizar pases largos	Numérico
skill_ball_control	habilidad para controlar el balón	Numérico

Tabla 5.6: Dimensiones de habilidades del rendimiento deportivo

Dimensión	Descripción	Tipo de dato
movement_acceleration	aceleración	Numérico
movement_sprint_speed	velocidad de sprint	Numérico
movement_agility	agilidad	Numérico
movement_reactions	reacción y reflejos del jugador	Numérico
movement_balance	balanceo del cuerpo	Numérico

Tabla 5.7: Dimensiones de movimiento del rendimiento deportivo

Dimensión	Descripción	Tipo de dato
power_shot_power	potencia de tiro	Numérico
power_jumping	potencia de salto	Numérico
power_stamina	determina la energía	Numérico
power_strength	fuerza del jugador	Numérico
power_long_shots	potencia de pase largo	Numérico

Tabla 5.8: Dimensiones de fuerza/resistencia del rendimiento deportivo

Dimensión	Descripción	Tipo de dato
mentality_aggression	agresividad	Numérico
mentality_interceptions	intercepciones de balón	Numérico
mentality_positioning	posicionamiento	Numérico
mentality_vision	vision de juego	Numérico
mentality_penalties	ejecución de penaltis	Numérico
mentality_composure	habilidad para estar calmado	Numérico

Tabla 5.9: Dimensiones de mentalidad del rendimiento deportivo

Dimensión	Descripción	Tipo de dato
defending_standing_tackle	entrada defensiva débil	Numérico
defending_sliding_tackle	entrada defensiva fuerte	Numérico

Tabla 5.10: Dimensiones de defensa del rendimiento deportivo

Dimensión	Descripción	Tipo de dato
goalkeeping_diving	estirada de portero	Numérico
goalkeeping_handling	manejo de las manos de portero	Numérico
goalkeeping_kicking	disparo de portero	Numérico
goalkeeping_positioning	posicionamiento de portero	Numérico
goalkeeping_reflexes	reflejos de portero	Numérico

Tabla 5.11: Dimensiones habilidades de portero del rendimiento deportivo

5.4. Almacenamiento de los datos

En este apartado se va a explicar el proceso que se ha llevado a cabo para cargar los datos en MongoDB así como la configuración de la base de datos.

Importación de los datos

Como ya se ha comentado en puntos anteriores, para el almacenamiento de los datos se ha utilizado la base NoSQL MongoDB.

Cuando se realiza el despliegue de MongoDB mediante Docker, se crea la base de datos **players_db** y se cargan en la misma las dos colecciones de datos que van a ser utilizadas por parte del sistema:

players En esta colección se almacena un documento por cada jugador, el cuál contiene las dimensiones asociadas al mismo, comentadas en el apartado del proceso ETL en el punto anterior. Los datos de esta colección serán los que se utilicen para la búsqueda de la similaridad.

teams En esta colección se crea un documento por cada equipo, el cuál contiene el nombre del equipo y el id del equipo. Este último es el que relaciona el equipo con el jugador. Además, los datos de esta colección servirán de soporte para la funcionalidad de la aplicación web, a la hora de visualizar los datos.

A partir de los dos ficheros CSV generados durante el proceso de ETL, se realizó una primera carga de las colecciones de datos en MongoDB, para validar los mismos y configurar el formato de cada campo. Este trabajo

se realizó para ambas colecciones utilizando la herramienta **MongoDB Compass** [25]:

En primer lugar se cargaron a través de la herramienta las dos colecciones desde los dos archivos CSV (los datos de los jugadores y los datos de los equipos). En primer lugar se realizó una validación del tipo de dato de cada dimensión, para asegurar la integridad del dato y que no haya futuros problemas a la hora de realizar consultas sobre los mismos.

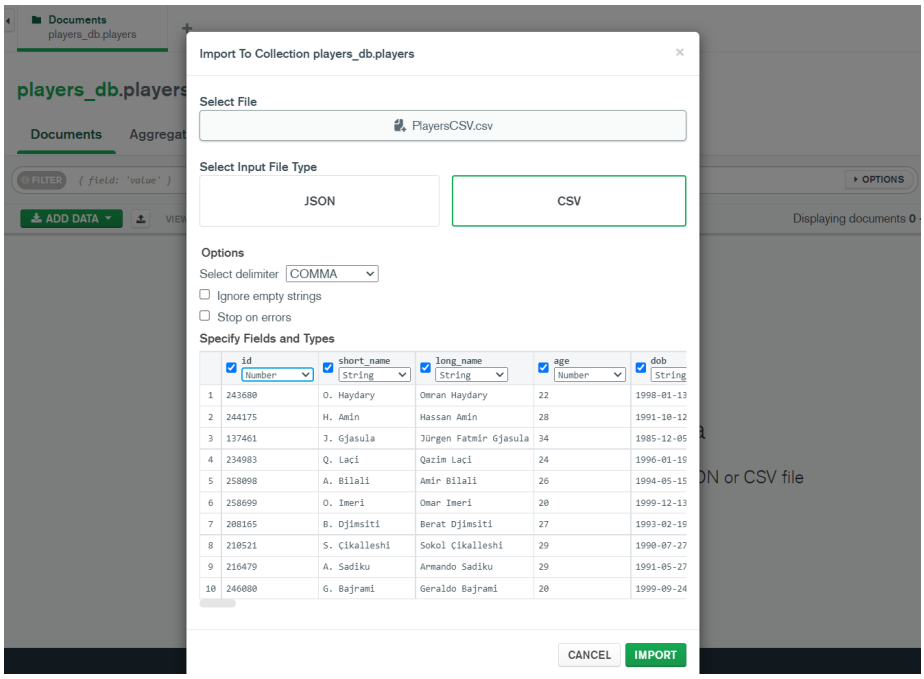


Figura 5.19: Validación de los diferentes campos correspondientes a las dimensiones

Una vez validados todos los datos y teniendo asignado el formato correcto y correspondiente a cada uno de ellos, se realizó la importación de ambas colecciones.

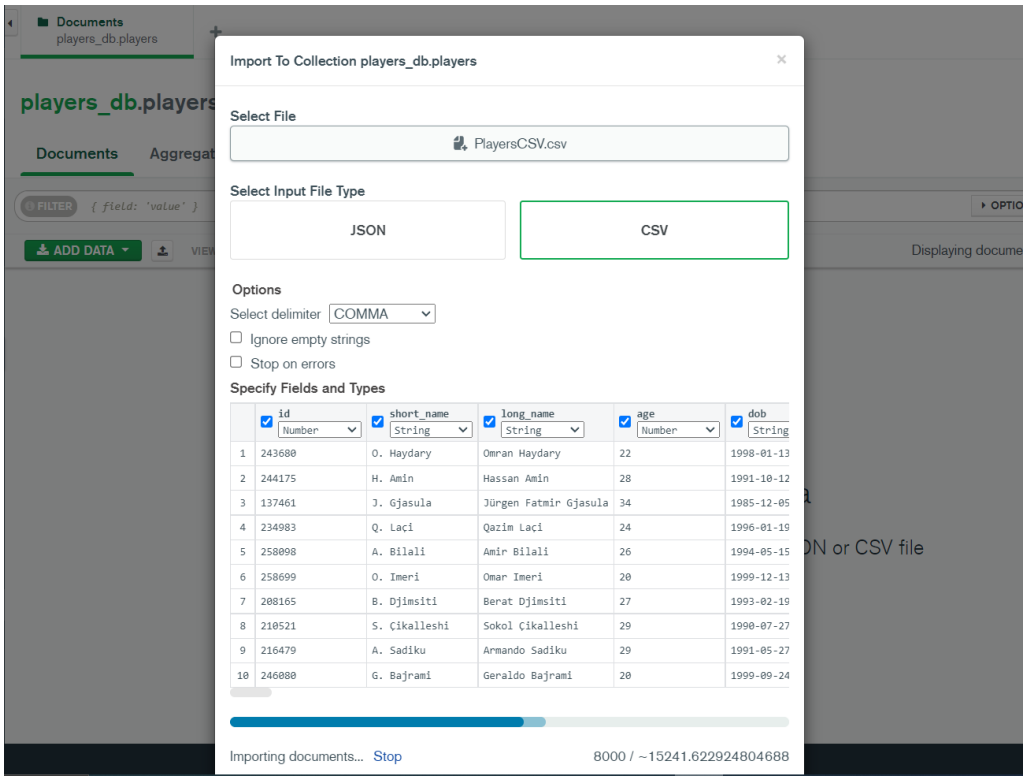


Figura 5.20: Importación de las colecciones de datos desde los ficheros CSV

Una vez realizada la importación, se validó que se hizo de manera correcta y se exportó de nuevo cada colección en formato JSON, obteniendo de nuevo dos ficheros, que serán los que finalmente se utilizarán desde Docker para cargar ambas colecciones en la base de datos cuando se despliegue todo el sistema.

De esta manera, aseguramos que la importación de la información en MongoDB a partir de los datos procesados durante la etapa de ETL, ha sido correcta. Además, al exportar los datos de nuevo desde MongoDB, y utilizar estos ficheros para cargar la base de datos cuando se despliegue el sistema desde Docker, nos aseguramos de que estos ficheros serán compatibles en cuanto a estructura y formato de datos, evitando así posibles errores futuros o problemas técnicos, derivados de un archivo incorrecto o algún dato cuyo formato pueda ocasionar algún tipo de problema.

Índices de la base de datos

A continuación se van a explicar los índices utilizados en las colecciones de datos que se usan en el proyecto para tener un mejor rendimiento de consultas.

Como ya se ha comentado previamente, la colección de datos **teams** contiene un documento por cada equipo, el cuál tiene dos atributos, como son el nombre del equipo y el id asociado al mismo. Para esta colección se ha optado por hacer un índice compuesto por ambos valores. La razón de hacerlo así es que con esta configuración podemos obtener una "consulta cubierta" [27]. Una consulta cubierta es aquella en la que la consulta puede satisfacerse íntegramente mediante un índice, y por tanto, no tener que consultar ningún documento. Un índice cubre una consulta cuando se cumplen las tres siguientes condiciones:

- Todos los campos que se utilizan en la consulta forman parte de un índice.
- Todos los campos devueltos en los resultados están en el mismo índice.
- Ningún campo que se utiliza en la consulta contiene un valor igual a null.

A la hora de obtener los datos de esta colección, siempre se que se haga por el id del equipo o por el nombre, y devuelva el id del equipo, el nombre del equipo o ambos datos, podremos cubrir la consulta de manera completa para que sea más eficiente, acorde a las buenas prácticas de rendimiento recomendadas por MongoDB [26].

Para hacer esto se ha creado un script que se ejecutará en el despliegue del sistema mediante Docker, y que se encargará de asignar los distintos índices para cada colección. Para la colección de **teams** se ha realizado de la siguiente manera:

```
db.teams.ensureIndex({ 'id': 1, 'club_name': 1});
```

Con esto vamos a crear un índice único cuyo primer elemento será el id del equipo y se ordenará de forma ascendente y como segundo elemento tendrá el nombre del equipo. En este caso el orden es indiferente ya que es un valor único y se ordenará con respecto al id del equipo.

Además, se ha tenido en cuenta el identificador propio que le asigna MongoDB a cada colección y que a su vez es índice, y se ha excluido del retorno de la consulta, para así poder tener la consulta completamente cubierta.

En el mismo sentido se han creado los índices par la colección **players**. Para esta colección se ha creado el siguiente índice para tener de nuevo una consulta completamente cubierta:

```
db.players.ensureIndex({'club_id':1 ,  'id': 1, 'short_name':
↪  1});
```

De esta manera, las consultan que se realicen para buscar todos los jugadores de un mismo equipo utilizando el id del equipo, y que devuelvan alguno o todos estos campos utilizados en el índice, serán más eficientes. De igual forma que para el caso anterior, se ha excluido el identificador propio de MongoDB a la hora de devolver los resultados para así poder tener la consulta completamente cubierta.

Para ver la diferencia de rendimiento de estas consultas utilizando índices, se ha utilizado la consola de MongoDB Compass para realizar la misma consulta sin índice y con índice. Para el último caso anterior, los resultados de hacer la consulta sobre la colección sin índice han sido los siguientes:

```
    winningPlan:
     { stage: 'PROJECTION_SIMPLE',
       transformBy: { short_name: 1, id: 1, _id: 0 },
       inputStage:
        { stage: 'COLLSCAN',
          filter: { club_id: { '$eq': 481 } },
          direction: 'forward' } },
    rejectedPlans: [] },
executionStats:
 { executionSuccess: true,
   nReturned: 33,
   executionTimeMillis: 63,
   totalKeysExamined: 0,
   totalDocsExamined: 18944,
```

Figura 5.21: Resultados de la ejecución de la consulta de los jugadores de un equipo sin índice

Vemos que en el apartado de "stage" de "winningPlan" nos indica el modo utilizado para realizar la consulta y en este caso es COLLSCAN, lo que implica que ha recorrido toda la colección para buscar los datos. En el apartado de "executionStats" vemos que la consulta ha devuelto 33 resultados, utilizando un tiempo de 63 milisegundos y ha examinado 18944 elementos, es decir, toda la colección completa.

Una vez creado el índice, repitiendo la misma consulta, los resultados en cuanto al modo de realizarla, son los siguientes:

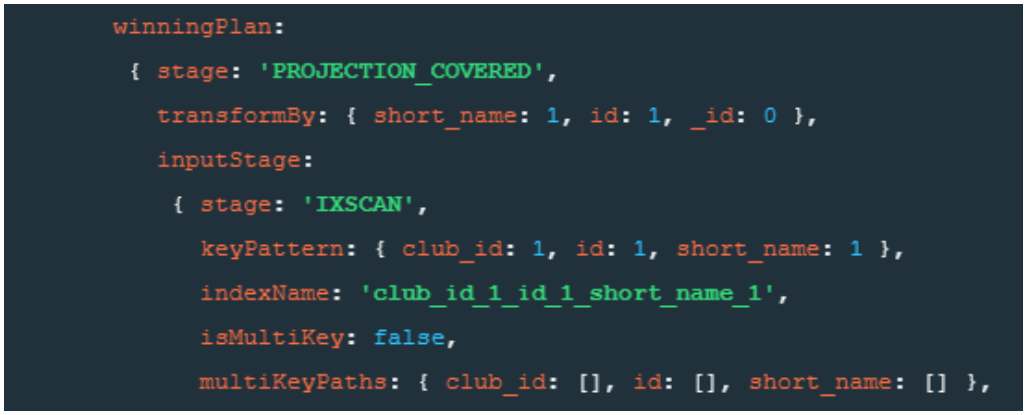


Figura 5.22: Resultados del modo en que se realiza la ejecución de la consulta de los jugadores de un equipo con índice

Vemos que en el apartado "stage" de "winningPlan" indica que la forma de realizar la consulta es IXSCAN. Esto significa que ya no recorre toda la colección, si no que utiliza los índices para encontrar los datos.

En cuanto a las estadísticas de la consulta realizada con los índices obtenemos lo siguientes datos:

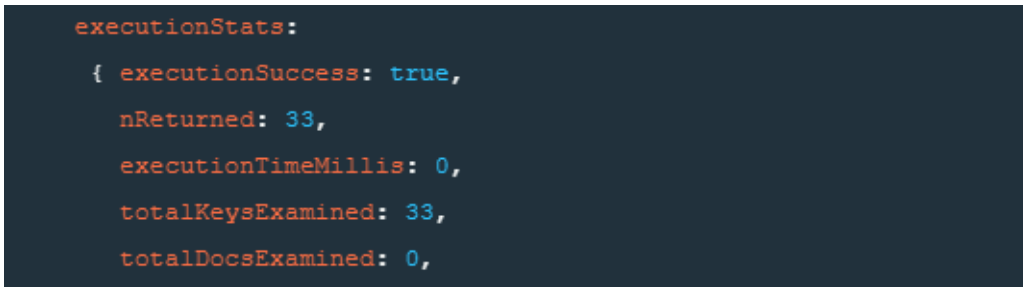


Figura 5.23: Estadísticas la ejecución de la consulta de los jugadores de un equipo con índice

Vemos que ha recorrido 33 claves y ha devuelto 33 resultados, pero en este caso, no ha tenido que recorrer toda la colección completa para obtener los resultados. En cuanto al tiempo empleado, vemos que esta vez ha tardado menos de un milisegundo, mejorando notablemente el rendimiento de la consulta realizada.

5.5. Servicios Web

Una vez almacenados los datos como se ha comentado en el punto anterior, tenemos que poder utilizarlos para calcular la similaridad del jugador que se solicite.

Para realizar esta operativa, se ha optado por realizar una implementación SaaS (software as a service), de manera que la lógica y los datos se mantienen en un servidor y el servicio será consumido mediante peticiones al mismo. Se ha optado por realizar esta operativa mediante servicios webs para que el sistema sea mucho más escalable. De esta forma pueden consumir los resultados distintas aplicaciones que estén desarrolladas a su vez en diferentes tecnologías, tanto aplicaciones web, aplicaciones móviles o cualquier otro tipo de sistema que pueda recibir como input datos en JSON. De esta forma el cálculo de la similaridad es totalmente transparente para el consumidor de los servicios, y de esta forma, hacer que el sistema sea mucho más flexible. También el mantenimiento del mismo será transparente, ya que si hay que modificar cualquier tipo de operación a realizar para calcular la similaridad no afectará a las aplicaciones o clientes que consuman el resultado obtenido.

Para realizar esta operativa se han desarrollado los servicios web indicados utilizando Python y la librería Flask.

Esos servicios web cuentan con tres métodos principales:

- getAllTeams
- getTeamPlayers
- getSimilarity

A continuación se explica cada uno de ellos.

getAllTeams

Este método devuelve en formato JSON todos los equipos de fútbol almacenados en el sistema ordenados alfabéticamente por su nombre. De esta forma podremos mostrar los jugadores para que, por ejemplo, le sea más fácil al usuario poder buscar y localizar el jugador modelo sobre el que quiere partir para realizar la búsqueda de similaridad.

```
@app.route("/getAllTeams")
def getAllTeams():

    db=""
    try:
        db = get_db()
        teamsCollection = db['teams']
        return
        ↪  dumps(teamsCollection.find({},{'_id':False}).sort("club_name"))

    except Exception as e:
        return jsonify({"errorMsg":"Failed: "+ str(e)})
    finally:
        if type(db)==MongoClient:
            db.close()
```

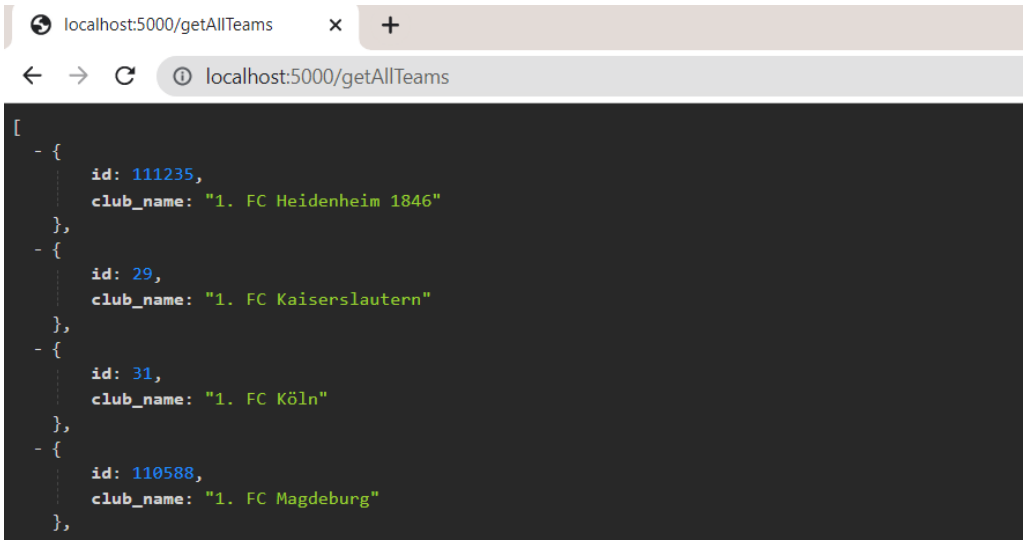


Figura 5.24: Ejemplo de parte de los datos que devuelve el método getAllTeams

getTeamPlayers

Este método devuelve en formato JSON los datos de todos los jugadores del equipo que se recibe como parámetro y devuelve la abreviatura del nombre del jugador y su id:

```
db=""
    try:
        db = get_db()
        playersCollection = db['players']

        return dumps(playersCollection.find({"club_id":idTeam},
        ↪ {"short_name": 1, "id": 1, '_id':False}).sort("short_name"))

    except Exception as e:
        return jsonify({"errorMsg":"Failed: "+ str(e)})
    finally:
        if type(db)==MongoClient:
            db.close()
```

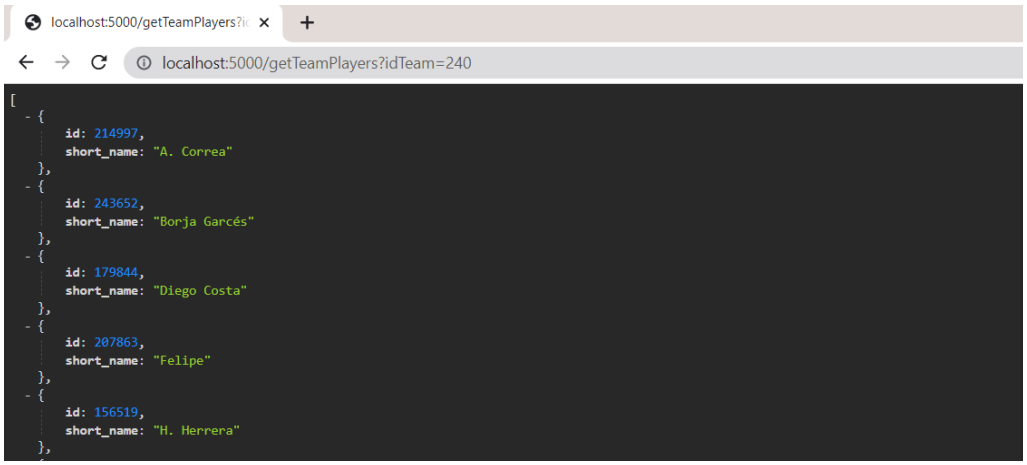


Figura 5.25: Ejemplo de parte de los datos que devuelve el método getTeamPlayers

getSimilarity

Este método recibe como parámetros el id del jugador sobre el que se quiere buscar la similaridad,el número de los jugadores más similares que se tienen que obtener y las métricas de las características deportivas que tienen que tener los jugadores sobre los que realizar la búsqueda de la similariad. Con estos datos, calcula la similaridad del jugador solicitado entre todos los jugadores almacenados en el sistema que cumplan las condiciones recibidas y devuelve en formato JSON los datos de los jugadores más similares encontrados.

En primer lugar se recogen los datos de MongoDB de todos los jugadores del sistema que cumplen con los requisitos recibidos para buscar la similaridad y se almacenan en un DataFrame.

Este DataFrame con los datos se duplica. El motivo es que en el primero guardaremos los datos sin manipular y este segundo DataFrame será utilizado para realizar las distintas operaciones sobre los datos. Como queremos buscar la similaridad en base a su rendimiento y condiciones deportivas, se eliminan del DataFrame los campos que no estén relacionados con el mismo.

```
playersDataFrame2.drop(['id','short_name','long_name','dob','nationality',
'nationality_iso','club_id','club_name','league_name','league_rank','value_eur',
'wage_eur','player_positions','preferred_foot','release_clause_eur',
'contract_valid_until'],axis=1, inplace=True)
```

Una vez realizadas estas operaciones, convertimos este DataFrame a array en formato float32 de numpy.

En este punto ya podemos crear la query de consulta que usará FAISS con todos los datos que necesitamos para calcular la similaridad:

```
    # Creamos una función para crear la query que usaremos
# Los parámetros de entrada son:
# playersOri: Datos originales de los jugadores
# playersNumpy: Datos de los jugadores ya vectorizados
# idsJugadores: Jugadores de los que queremos encontrar elementos
↪  similares
# La función devuelve la query para que se pueda usar en FAISS

def creaQuery (playersOri, playersNumpy,idsJugadores):

    query = []

    for idj in idsJugadores:

        # Buscamos el índice en el que están
        i = playersOri[playersOri['id'] ==
        ↪  int(idj)].index.values.astype(int)[0]

        # Asignamos a la query los datos del jugador a buscar desde
        ↪  los datos vectorizados
        query.append(playersNumpy[i])

    # Hacemos cast y pasamos la lista a array de Numpy
    query = np.asarray(query)

    # Devolvemos la query
    return query
```

Finalmente se ha creado una función que ejecuta la búsqueda de la similaridad a partir de la query creada en el paso anterior:

```
    # Se crea la función buscaJugadoresSimilares
# Como parámetros de entrada tendrá:
# datos: Nuestros datos vectorizados previamente
# Los jugadores origen sobre los que queremos encontrar jugadores
↪  similares
# El número de jugadores similares que se tienen que encontrar para
↪  los jugadores recibido


def buscaJugadoresSimilares(datos, query, numJugadoresSimilares):
```

```
    # Calculamos el número de dimensiones
    dim = datos.shape[1]

    # Creamos un índice que será el tamaño de los vectores
    index = faiss.IndexFlatL2(dim)

    # Comprobamos si está entrenado
    #print(index.is_trained)

    # Añadimos al índice el conjunto de datos con ascontiguosarray
    ↪  para que funcione de manera correcta
    index.add(np.ascontiguousarray(datos))

    # queremos los k elementos más similares
    k = numJugadoresSimilares
    D, I = index.search(query,k)

    return D,I
```

Esta función nos devolverá la distancia de los jugadores respecto al jugador del que partimos, así como los índices de los jugadores más similares encontrados para localizar su posición en el vector.

En este punto ya tenemos las posiciones de los jugadores similares dentro del vector, pero al no tener sólo los datos de las métricas deportivas, necesitamos recuperar los datos originales de los jugadores. Para ello se ha creado una función que realiza esta operación y busca en el DataFrame inicial el índice de los jugadores encontrados, y de esta forma, recuperar la información del jugadores que está en esa posición:

```
    # Creamos una función que nos devolverá más datos (equipo,
    ↪  nombre, posición...) de los jugadores encontrados
# Tenemos el vector con sólo las estadísticas deportivas
# y nos interesa buscar todos los datos completos, como el equipo,
↪  la liga, etc que necesitemos, y esto lo tenemos en los datos
↪  originales

def datosJugadoresEncontrados(playersOri,indices, distancias):

    # Para devolver los datos buscados de los jugadores
    #datosJugadores =
    ↪  pd.DataFrame(columns=playersOri.columns.values)
    datosJugadores = pd.DataFrame
    ↪  (columns=playersDataFrame.columns.values)
```

```
        # Recuperamos de los datos originales la información que
        ↪  necesitamos
        for j in range(0, len(indices)):

            for i in range(0,len(indices[j])):

                # Fila con todos los datos del jugador encontrado
                datosJugador = playersOri.iloc[indices[j][i]]

                # Distancia
                distancia = distancias[j][i]

                #Añadimos la columna distancia con el valor a la row del
                ↪  jugador
                datosJugador['distancia'] = distancia

                datosJugadores = datosJugadores.append(datosJugador)

        return datosJugadores
```

En este punto ya tenemos los datos de los jugadores más similares que se han encontrado acorde a las características solicitadas, además de la distancia entre ellos. Estos datos se devuelven por parte del servicio web en formato JSON para que puedan ser consumidos por el cliente.

5.6. Visualización de los datos

En esta sección se van a detallar los pasos realizados para la visualización de los resultados obtenidos con la búsqueda de la similaridad mediante la aplicación web.

Web scraping

Para obtener las imágenes y los escudos de los equipos relacionados, se ha contactado con la web de Sofifa [1] para solicitar el uso de los assets relacionados con los jugadores y equipos de los que dispone en su web.

Para obtenerlos, se ha generado un script en Python que scrapea las imágenes de los jugadores y nos permite descargarlas a nuestro ordenador. Para ello, se ha partido de las url de las mismas, que vienen referencias en un campo del DataSet inicial de Kaggle desde el que hemos partido inicialmente como origen de los datos.

[1] https://sofifa.com/

```
for i in range (0,df.shape[0]):

    #Cogemos la url
    url = df['photo'].values[i]


    #Guardamos la imagen con el nombre original
    res = requests.get(url, stream = True)

    if res.status_code == 200:
        with
        ↪  open(final_directory+"\\"+str(df['id'].values[i])+'.png','wb')
        ↪  as f:

            # Creamos la imagen
            shutil.copyfileobj(res.raw, f)

            # Agregamos al log
           print('Descarga OK')

    else:
        print('Error en la descarga')
```

Esto nos creará una carpeta llamada **players** en el directorio donde se está ejecutando el script dónde se irán guardando todas las imágenes de los jugadores.

De manera similar se han obtenido las imágenes de los equipos de fútbol, con la diferencia de que en este caso se ha analizado la estructura de la url bajo la que se encuentran en la propia web. A partir de esta estructura, se ha creado un archivo csv con las urls de las imágenes de los distintos equipos que están indentifcados por su id. Este archivo será utilizado por el script de Python para poder obtener las imágenes:

```
for i in range (0,df.shape[0]):

            #Cogemos la url
            url = df['urls'].values[i]

            #Traemos el nombre del directorio a crear que estará entre
            ↪  los dos últimos backslash
            directorio =  url.rsplit('/', 2)[1]

            #Creamos el nuevo directorio dentro de la carpeta teams
            path = "teams\\"+directorio
            access_rights = 0o777
```

```
        os.mkdir(path,access_rights)

        #Guardamos la imagen con el nombre original
    res = requests.get(url, stream = True)

        if res.status_code == 200:
            with open(path+'\\60.png','wb') as f:
                shutil.copyfileobj(res.raw, f)
            print('Imagen descargada OK: ',str(i+1)+")
            ↪  "+path+'\\60.png')
        else:
            print('Error en descarga')
```

Esto nos creará una carpeta llamada **teams** en el directorio donde se está ejecutando el script dónde se irán guardando todas las imágenes de los escudos de los equipos.

En este punto ya tenemos los assets que usaremos posteriormente en la aplicación web para identificar a los jugadores y sus respectivos equipos.

Aplicación Web

Como se ha explicado en los puntos anteriores, ya tenemos los datos almacenados y los servicios web que realizan la búsqueda de la similaridad. Para poder visualizar los resultados obtenidos mediante esta búsqueda, se ha desarrollado una aplicación web. Desde esta aplicación web se consumirán los servicios web y se mostrarán los resultados obtenidos. A través de la misma, el usuario podrá seleccionar el jugador base sobre el que se buscará la similaridad y podrá visualizar los resultados obtenidos.

El diseño de la visualización de la página web se ha realizado siguiendo el patrón F [16], acorde al comportamiento habitual de los usuarios a la hora de leer el contenido de páginas y aplicaciones web.

En la parte izquierda de la aplicación web, se ha añadido un menú lateral desde el que se seleccionará el equipo, el jugador del que partimos para buscar la similaridad y el número de jugadores similares que queremos que nos devuelva. La aplicación mostrará en el primer selector los equipos disponibles en el sistema de manera automática. Una vez el usuario haya seleccionado el equipo, se cargarán en el siguiente selector todos los jugadores que pertenecen a ese equipo y, una vez seleccionado el jugador modelo sobre el que buscar la similaridad, se seleccionará el número de los jugadores más similares a mostrar.

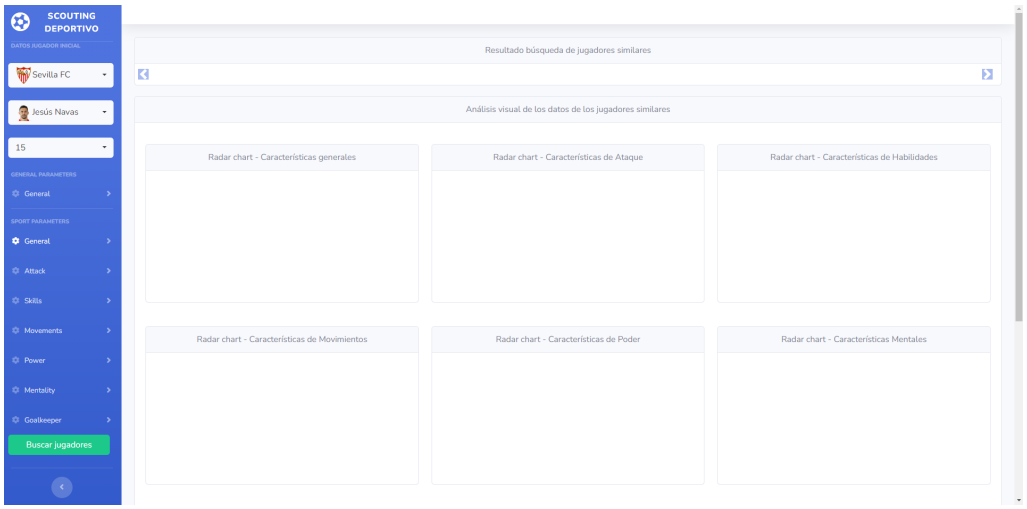


Figura 5.26: Ejemplo selección desde la barra lateral del jugador sobre el que se parte

Tanto a la hora de buscar el equipo como el jugador, se puede utilizar una ayuda de búsqueda a la hora de localizar estos datos, escribiendo parte del nombre del dato a buscar. Esto hará que se filtren los equipos y los jugadores que tengan coincidencia con lo escrito.

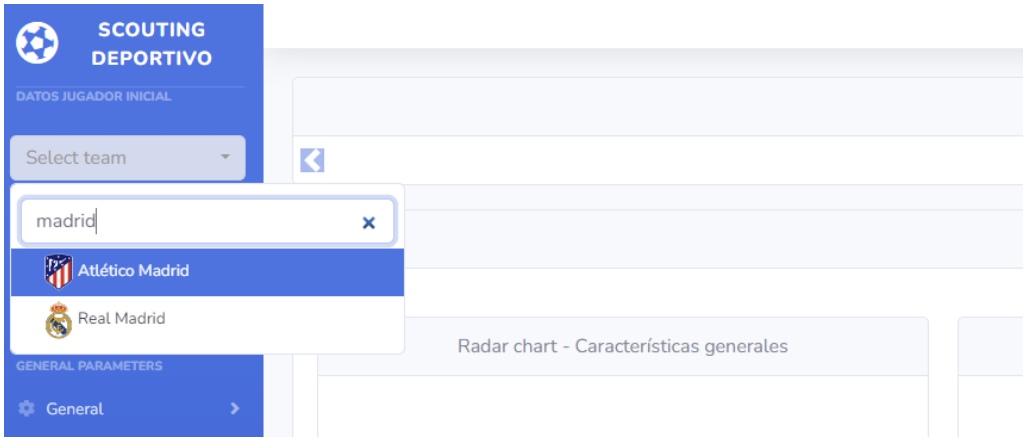


Figura 5.27: Ejemplo ayuda de búsqueda

Además, en el mismo menú lateral, podremos filtrar las características deportivas que tienen que tener los jugadores sobre los que se busque esa

similaridad. De esta forma, podemos personalizar más la búsqueda, ajustarla y adaptarla a nuestros requisitos. Se ha creado un bloque por cada dimensión de características, el cual se puede mostrar y ocultar a gusto del usuario.

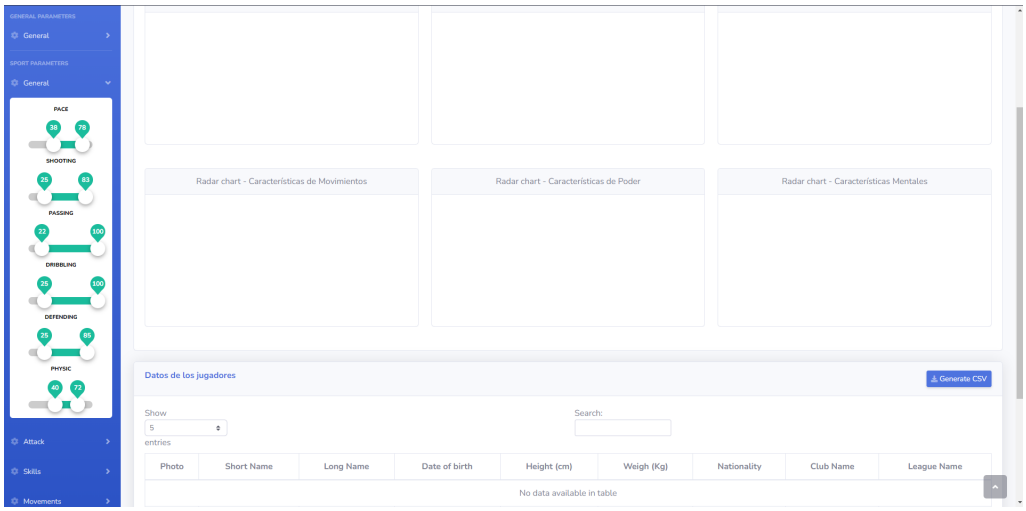


Figura 5.28: Ejemplo selección características deportivas

Esta barra lateral también cuenta con la opción de contraerla, para que por ejemplo, una vez seleccionados los datos, se pueda ganar más campo de visión sobre los mismo. Esto se realiza mediante el botón inferior de la misma.

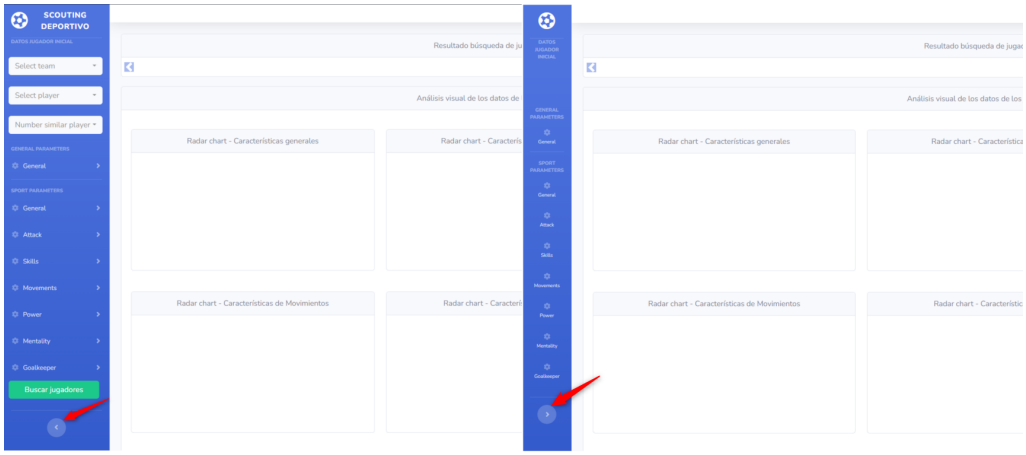


Figura 5.29: Comparativa barra lateral expandida y contraída

Una vez sel,eccionados estos datos, podemos ejecutar la búsqueda y nos devolverá los resultados de los jugadores más similares encontrados acorde a las características y los filtros que hemos indicado.

En cuanto a la visualización de los resultados, una vez se han recibido los resultados de la búsqueda de la similaridad, en la parte superior de la página podremos ver un "carrousel" con el resumen de los datos de los jugadores similares localizados, como la imagen, el equipo, la nacionalidad o el nombre entre otros. Este "carrousel" irá mostrando todos los jugadores de manera automática y también podremos desplazar los datos mediante los bonotes laterales con los que cuenta. Esto servirá para poder tener un resumen visual de un vistazo de los resultados obtenidos.

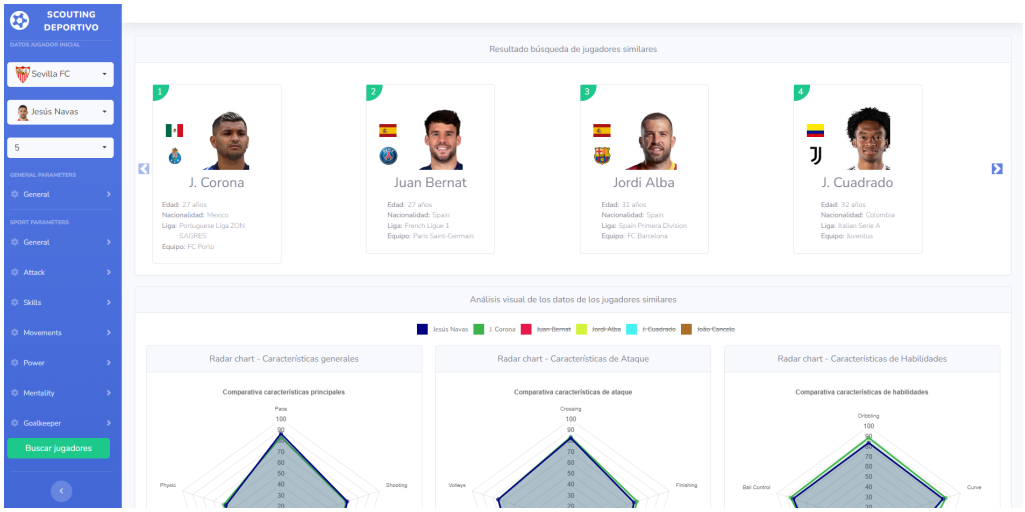


Figura 5.30: Carrousel con los jugadores similares encontrados

En la siguiente parte de la página, hacia abajo, se han añadido seis radar charts para poder comparar la similaridad en cuanto a las características deportivas de los jugadores, de manera visual. Estas gráficas se disponen conformando un panel en cuadrículas aplicando la regla del tercio utilizando un patrón de diseño Z [23]. Las gráficas se presentan separadas por un pequeño espacio en blanco entre ellas para que facilite su legibilidad.

Hay un radar chart por cada familia de características deportivas y son las dimensiones que se filtran desde la barra lateral:

- Dimensiones generales del rendimiento deportivo

- Dimensiones de ataque del rendimiento deportivo
- Dimensiones de habilidades del rendimiento deportivo
- Dimensiones de movimiento del rendimiento deportivo
- Dimensiones de fuerza/resistencia del rendimiento deportivo
- Dimensiones de mentalidad del rendimiento deportivo
- Dimensiones de habilidades de portero del rendimiento deportivo

Para el caso de las dimensiones relacionadas con habilidades de portero, se ha tenido que adaptar la aplicación, ya que los valores de características generales no están informados para los porteros, si no que tienen sus valores personalizados, debido a la peculiaridad de su posición con respecto al resto de jugadores. Por tanto, estas habilidades sólo se mostrarán cuando la posición del jugador seleccionado como modelo sobre el que buscar la similaridad sea un portero, ya que son los únicos casos para los que existe esta información de habilidades. En esos casos, se mostrarán estos datos en la gráfica de características generales.

Todas las gráficas se han creado con la misma escala para mantener una consistencia entre las mismas,de forma que no haya interpretaciones erróneas de los datos y puedan dar lugar a confusión. De igual forma, se mantiene una consistencia cromática entre las distintas gráficas, utilizando el mismo color para un mismo jugador en todas ellas.

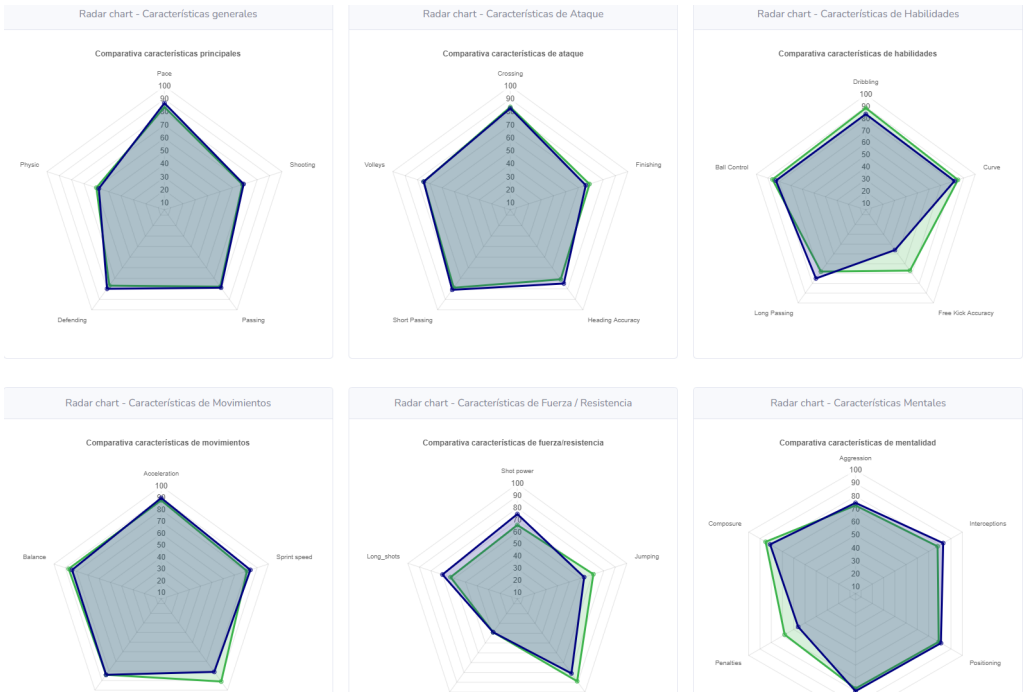


Figura 5.31: Radar charts con los datos de las habilidades de los jugadores

Inicialmente se muestran representados en las gráficas los valores del jugador sobre el que partimos a la hora de buscar la similaridad y del jugador más similar encontrado, para que sea más legible. Se deja en manos del usuario que pueda seleccionar o des seleccionar los jugadores que considere a la hora de hacer la comparación. Se ha optado por elegir colores lo más diferentes posible [34] para que en caso de seleccionar más de un jugador a la vez, ayude a destacar su visualización. Esta selección de los jugadores a mostrar se realiza desde el menú superior de la sección de las gráficas, y aplica para todas las gráficas al mismo tiempo para mantener una consistencia.

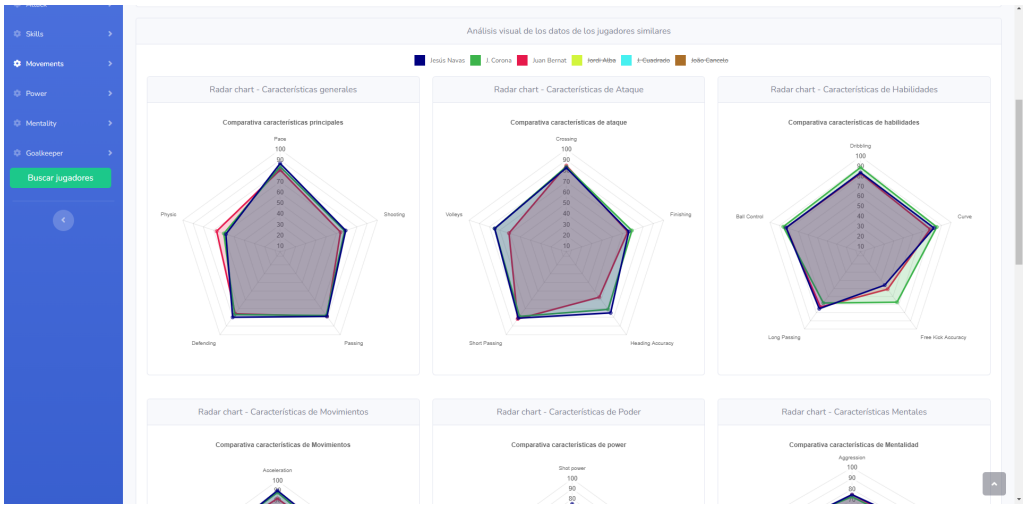


Figura 5.32: Selección de jugadores a mostrar en las gráficas

En la parte inferior de la página aparecerán los datos de los jugadores en formato tabla, complementando la información relacionada con los mismos. Podremos ir navegando por las diferentes partes de la tabla a la hora de visualizar la información, así como por las diferentes páginas de la misma, y además, podremos seleccionar el número de resultados que se visualizan por cada página. También cuenta con una ayuda de búsqueda para poder filtrar por cualquier término de los campos, de manera que nos facilite la visualización de los mismos acorde a nuestras necesidades.

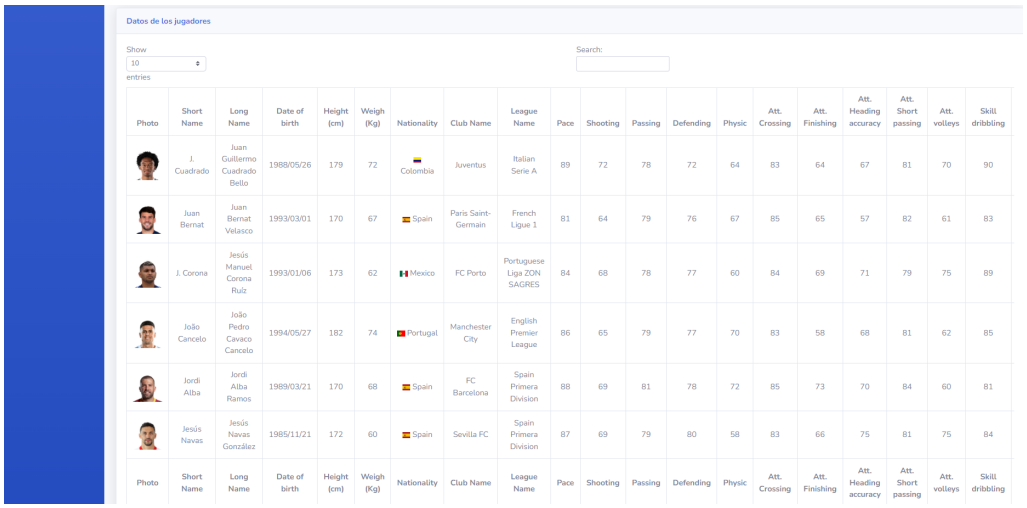


Figura 5.33: Visualización de los datos de los jugadores en formato tabla

5.7. Despliegue del sistema con Docker

Una vez se han desarrollado todos y cada uno de los componentes que serán necesarios para poder utilizar el sistema completo, sólo queda pendiente desplegar los componentes mencionados y tenerlos disponibles para poder usarlos. Para realizar este despliegue del sistema se ha utilizado Docker.

Docker Compose

Docker Compose es una herramienta del entorno de Docker que sirve para definir la infraestructura de los servicios que se quieren desplegar. Esta definición de servicios se realiza mediante un fichero de formato YAML, que será el que lea Docker Compose a la hora de ejecutar el despliegue del sistema. Con este sistema se simplifica la instalación de los diferentes servicios que van a ser necesarios a la hora de iniciar los contenedores de los mismos, ya que todos quedan recogidos en un único archivo. Para este proyecto se despliegan los siguientes servicios:

- Servicios web desarrollados en Python mediante Flask
- Base de datos MongoDB con los datos de los jugadores y equipos
- Servidor web para ejecutar la aplicación web, en este caso es Nginx

El primero de los componentes a tener en cuenta a la hora de desplegar el sistema son los servicios web desarrollados en Python y que devolverán la información de la similaridad. Estos servicios se encuentran definidos en el archivo app.py y se desplegarán en el puerto 5000.

```
app:
    build: .
    command: python -u app.py
    ports:
      - "5000:5000"
    volumes:
      - .:/app
```

Para el almacenamiento de los datos se desplegará MongoDB. En este despliegue se iniciará la base de datos y se cargarán los datos a almacenar a partir de dos archivos JSON, uno para los datos de los jugadores y otro con los datos de los equipo. Este servicio se desplegará en el puerto por defecto 27017.

```
mongoimport:
    image: mongo:latest
    container_name: my-import
    volumes:
      - ./mongodb/collections:/src/data/
    command: sh -c "mongoimport  -u "root" -p "pass"
    ↪ --authenticationDatabase "admin" --host
    ↪  mongodb_players_host --db players_db --collection
    ↪  players --type json --file /src/data/players.json
    ↪ --jsonArray
                  && mongoimport  -u "root" -p "pass"
                  ↪ --authenticationDatabase "admin"
                  ↪ --host mongodb_players_host --db
                  ↪  players_db --collection teams --type
                  ↪  json --file /src/data/teams.json
                  ↪ --jsonArray"
```

Finalmente de desplegará Nginx, el servidor web que permitirá la ejecución de la aplicación web que se ha desarrollado para visualizar los datos. Este servicio se desplegará en el puerto por defecto 8080:80.

```
  web:
        image: nginx:latest
        ports:
            - "8080:80"
        volumes:
          - ./website:/var/www/html
          - ./default.conf:/etc/nginx/conf.d/default.conf
```

Con esto ya tendríamos todos los componentes principales definidos en el archivo que ejecutará DockerCompose a la hora de realizar el despliegue de los mismos.

Dockerfile

Mediante el archivo Dockerfile, ejecutaremos las dependencias de algunas de los componentes definidos en el archivo YAML de DockerCompose. En este caso en concreto, instalaremos las librerías que se necesitan en Python para ejecutar los servicios webs.

```
FROM python:3.6
ADD . /app
WORKDIR /app
RUN pip install -r requirements.txt
```

En este caso las dependencias están definidas en el archivo requirements.txt, que en este caso son las siguientes:

- Flask
- Flask-Cors
- pymongo
- faiss-cpu==1.6.1
- pandas
- numpy

Una vez se ejecute el comando anterior de pip, se instalarán todas las librerías indicadas y los servicios webs podrán ejecutarse mediante Python.

Una vez se han definido estos requisitos, sólo nos queda ejecutar el despliegue del servicio mediante los siguientes siguientes comandos:

```
docker-compose build --no-cache
```

```
docker-compose up
```

En este momento comenzará la creación de las imágenes y la carga de las dependencias por parte de Docker. Una vez finalizado el proceso, ya estarán todos los servicios disponibles y podremos ejecutar la aplicación web para empezar a buscar la similaridad del jugador o jugadores que necesitemos.

5.8. Análisis de los resultados

En esta sección se analizarán algunos de los resultados obtenidos para algunos jugadores, comparándolos con los movimientos entre equipos que hayan realizado los mismos en la vida real.

Jesús Navas

Jesús Navas es un futbolista del Sevilla FC que juega por la banda derecha del equipo. Si ejecutamos la aplicación tomando este jugador como modelo sin utilizar ningún filtro adicional y solicitando los 5 jugadores más similares a él, el resultado obtenido es el que se muestra en la siguiente imagen:

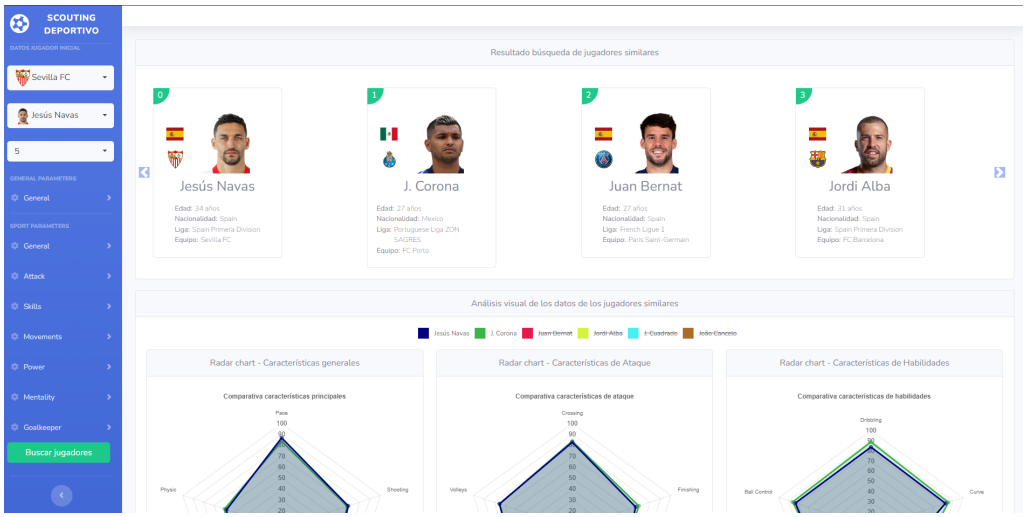


Figura 5.34: Resultado de la búsqueda de la similaridad para Jesús Navas

Como vemos nos devuelve cinco jugadores, que de más a menos similares, son:

Jugador	Edad	Nacionalidad	Equipo
J. Corona	27	Mexico	FC Porto
Juan Bernat	27	Spain	Paris Saint-Germain
Jordi Alba	31	Spain	FC Barcelona
J. Cuadrado	32	Colombia	Juventus
João Cancelo	26	Portugal	Manchester City

Tabla 5.12: Resultado similaridad Jesús Navas

Si analizamos la comparativa de sus habilidades, vemos que son prácticamente idénticos en cuanto a características generales, ataque y habilidades, en la que J.Corona es un poco mejor en cuando a lanzamientos de tiros libres.

Figura 5.35: Comparativa características generales, ataque y habilidades entre Jesús Navas y J.Corona

Si analizamos el resto de habilidades, vemos que también son muy similares, con las mínimas diferencias en agilidad, energía y la mentalidad a la hora de lanzar penaltis.

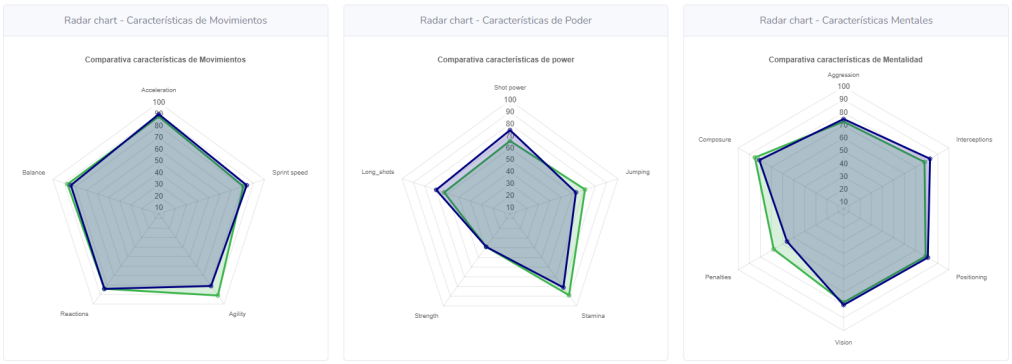


Figura 5.36: Comparativa características de movimientos, fuerza/resistencia y mentales entre Jesús Navas y J.Corona

Como se puede apreciar, en cuanto a rendimiento deportivo en referencia a las habilidades mencionadas, la similaridad es muy alta. Si vemos la trayectoria profesional de J. Corona, vemos que fichó por el Sevilla en enero de 2022 [13], quizá con la idea de encontrar ese jugador que le pueda dar un rendimiento similar a Jesús Navas, por ejemplo, en caso de recambio,

posibilidad de lesión del primero o rotación de jugadores entre a la hora de conformar la alineación en las distintas competiciones.

Figura 5.37: Fichaje J.Corona por el Sevilla [13]

Sergio Asenjo

Sergio Asenjo es un jugador que juega en la demarcación de portero y que estuvo en el Villarreal FC desde el 2014 hasta 2022. Si realizamos la similaridad para este jugador teniendo en cuenta los datos de 2020 obtenemos el siguiente resultado:

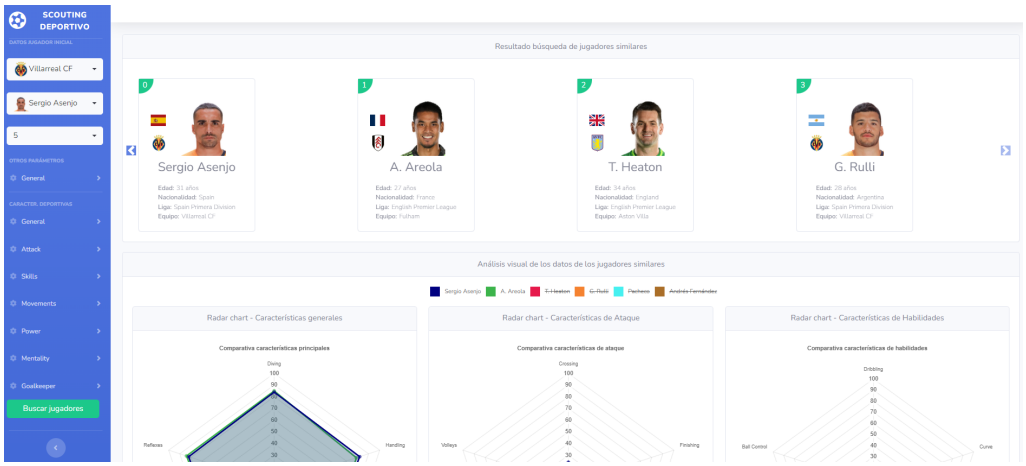


Figura 5.38: Resultado de la búsqueda de la similaridad para Jesús Navas

Como vemos, los cinco jugadores más similares, de más similar a menos similar son los siguientes:

Jugador	Edad	Nacionalidad	Equipo
A. Areola	27	France	Fulham
T. Heaton	34	England	Aston Villa
G. Rulli	28	Argentina	Villarreal CF
Pacheco	28	Spain	Deportivo Alavés
Andrés Fernández	33	Spain	SD Huesca

Tabla 5.13: Resultado similaridad Sergio Asenjo

En este caso vamos a realizar la comparación entre tres de los cinco jugadores más similares obtenidos que son el propio Sergio Asengjo, Gerónimo Rulli y Andrés Fernández. Si comparamos la visualización de sus habilidades generales, vemos que son prácticamente idénticos en cada una de las magnitudes que se miden, teniendo Andrés Fernández un poco menos de puntuación en cuanto al lanzamiento.

En cuanto a las características de ataque, vemos que de nuevo son muy similares, con la pequeña diferencia de los centros al área (Crossing) en la que Sergio Asenjo supera a los otros dos porteros comparados.

En cuanto a las habilidades, vemos que la única diferencia notable es que Gerónimo Rulli tiene peor habilidad en cuanto al pase largo.

Figura 5.39: Comparativa características generales, ataque y habilidades entre Sergio Asenjo, Gerónimo Rulli y Andrés Fernández

En cuanto a las habilidades de movimientos y fuerza/resistencia, los tres porteros están bastante igualados en todas las magnitudes. La mayor diferencia viene en la magnitud de agresividad, correspondiente al grupo de habilidades mentales, en la que Andrés Fernández destaca sobre los otros dos, lo que indica que es un portero que se enfrenta más en el cuerpo a cuerpo con los delanteros, por ejemplo, saliendo de su portería e intentar interceptar antes al jugador rival a la hora de intentar evitar los goles.

Figura 5.40: Comparativa características de movimientos, fuerza/resistencia y mentales entre Sergio Asenjo, Gerónimo Rulli y Andrés Fernández

Si vemos la trayectoria de los tres porteros en cuanto a los cambios de equipo, vemos que Andrés Fernández estuvo en el Villarreal desde el verano del 2017 [6] hasta Agosto de 2020, que fue fichado por el Huesca FC [7].

Figura 5.41: Fichaje de Andrés Fernández por el Huesca CF desde el Villarreal CF [7]

En paralelo, Gerónimo Rulli fichó por el Villarreal FC en ese mismo verano [8]. Por tanto, parece que El Villarreal FC buscaba un reemplazo ante

la salida de Andrés Fernández y que tuviera unas características similares a su portero actual Sergio Asenjo, que a su vez, las compartía con Andrés Fernández.

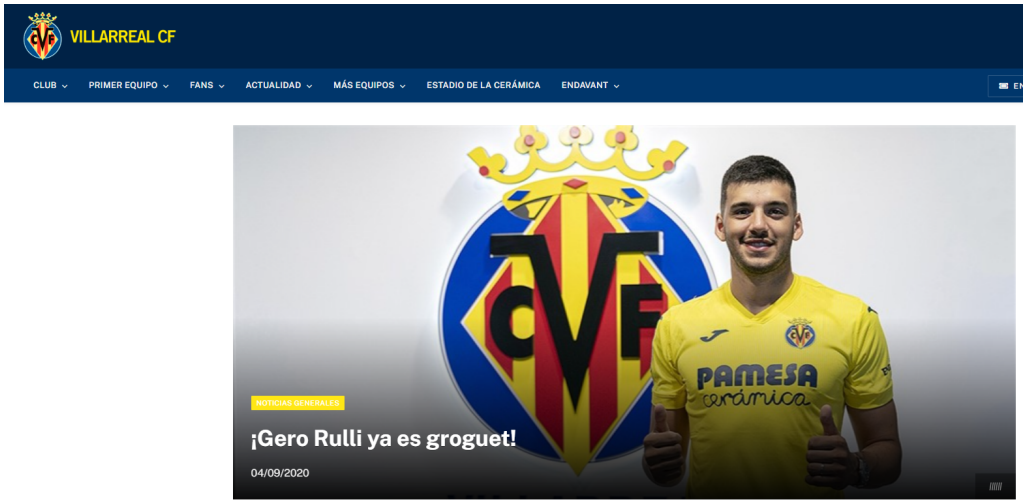


Figura 5.42: Fichaje de Gerónimo Rulli por el Villarreal desde la Real Sociedad [8]

Por tanto, quizá debido al estilo del juego del equipo o de las habilidades que buscara el entrenador para la portería, el Villarreal buscó un perfil similar que en este caso vemos que los tres jugadores compartían, acorde a los resultados obtenidos por el sistema.

João Félix

João Félix es un jugador que fichó por el Atlético de Madrid en el verano de 2019 [9] y su posición es la de delantero. Realizando la búsqueda de similaridad para este jugador con el sistema implementado obtenemos los siguientes datos:

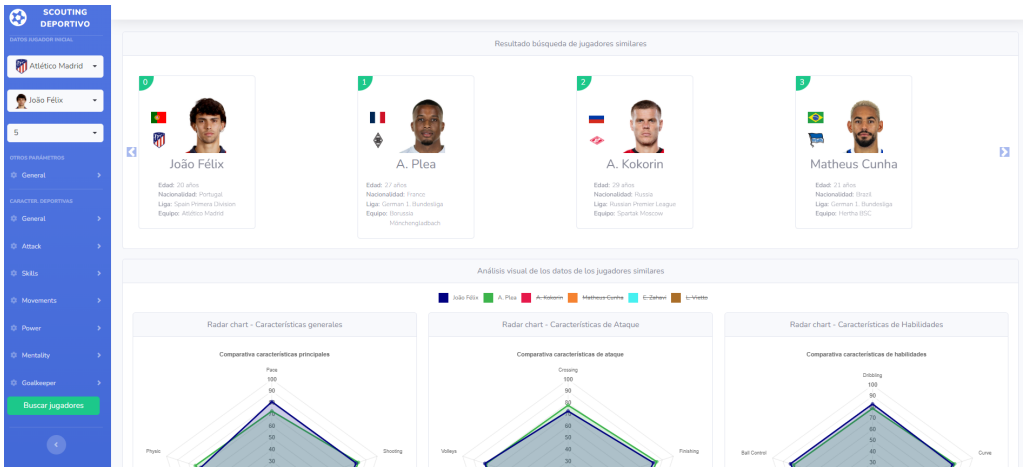


Figura 5.43: Resultado de la búsqueda de la similaridad para João Félix

Los jugadores obtenidos como resultado de esta búsqueda han sido los siguientes:

Jugador	Edad	Nacionalidad	Equipo
A. Plea	27	France	Borussia Mönchengladbach
A. Kokorin	29	Russia	Spartak Moscow
Matheus Cunha	21	Brazil	Hertha BSC
E. Zahavi	32	Israel	PSV
L. Vietto	26	Argentina	Sporting CP

Tabla 5.14: Resultado similaridad João Félix

Vemos que uno de los jugadores más similares encontrados por el sistema ha sido Matheus Cunha. Si nos fijamos en la comparación de las características generales, de ataque y de habilidades de los dos jugadores vemos que prácticamente son las mismas en todas ellas, con la única pequeña diferencia de que la defensa de las características principales es ligeramente superior para João Félix.

Figura 5.44: Comparativa características generales, ataque y habilidades entre João Félix y Matheus Cunha

En cuanto a las características de movimiento, fuerza/resistencia y mentales ocurre lo mismo que para las anteriores, con la única pequeña diferencia de que João Félix tiene una capacidad de interceptar balones mayor que Matheus Cunha.

Figura 5.45: Comparativa características de movimientos, fuerza/resistencia y mentales entre entre João Félix y Matheus Cunha

Si comparamos con la trayectoria de Matheus Cunha, vemos que fichó por el Atlético de Madrid en el verano de 2021 [10].

Figura 5.46: Fichaje de Matheus Cunha por el Atlético de Madrid [10]

Atendiendo a la similaridad que tiene Matheus Cunha con João Félix, este fichaje parece que se realiza con la intención de poder tener un sustituto que aporte un rendimiento similar a João Félix en caso de realizar cambios en un partido, en caso de lesiones o en caso de rotaciones por descanso en las distintas competiciones.

Sergio Ramos

Sergio Ramos era un jugador del Real Madrid FC que fichó por el equipo del París Saint Germain el verano de 2021 [15]. Si realizamos la búsqueda de similaridad sobre Sergio Ramos obtenemos los siguientes resultados:

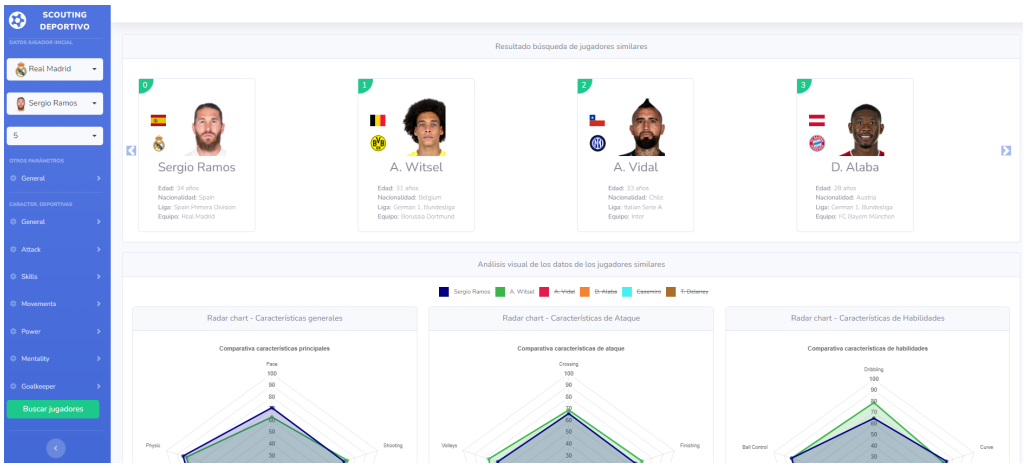


Figura 5.47: Resultado de la búsqueda de la similaridad para Sergio Ramos

Los jugadores más similares obtenidos como resultado de esta búsqueda han sido los siguientes:

Jugador	Edad	Nacionalidad	Equipo
A. Witsel	31	Bélgica	Borussia Dormund
A. Vidal	33	Chile	Inter
D. Alaba	28	Austria	Bayern München
Casemiro	28	Brasil	Real Madrid
T. Delaney	28	Dinamarca	Borussia Dormund

Tabla 5.15: Resultado similaridad Sergio Ramos

Analizando los resultados vemos que uno de los jugadores más similares encontrados es David Álaba. Si tenemos en cuenta la comparación de las características generales, de ataque y de habilidades de ambos, vemos que son muy similares entre sí, destacando respecto a las características de ataque que David Álaba mejora a Sergio Ramos en centros al área rival, y por el contrario, Sergio Ramos tiene una precisión de remate de cabeza mayor que David Álaba. En cuanto a habilidades, David Álaba mejora a Sergio Ramos en lanzamientos de falta, golpeo con efecto del balón y regate.

Figura 5.48: Comparativa características generales, ataque y habilidades entre Sergio Ramos y David Álaba

En cuanto a las características de movimiento, fuerza/resistencia y mentales, vemos que David Álaba mejora a Sergio Ramos en uanto a movimientos, siendo al contrario respecto a las características relacionadas con las características mentales.

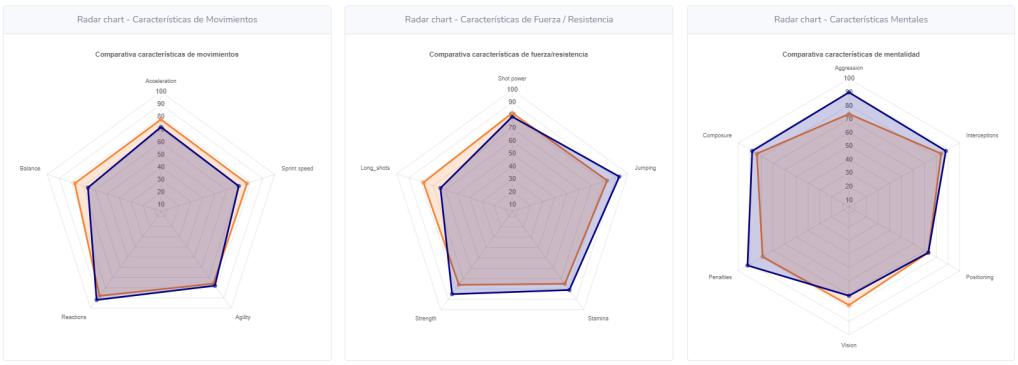


Figura 5.49: Comparativa características de movimientos, fuerza/resistencia y mentales entre entre Sergio Ramos y David Álaba

Si atendemos a la trayectoria deportiva de David Álaba, vemos que fichó por el Real Madrid en mayo de 2021 [5], al mismo tiempo que Sergio Ramos dejaba el club ese mismo verano. Este movimiento parece realizado para sustituir a Sergio Ramos, ante la su venta al equipo del París Saint Germain y poder suplir su baja sin que el equipo pueda perder rendimiento en defensa.

Como se puede comprobar, los resultados obtenidos con la aplicación desarrollada son bastante acertados, teniendo en cuenta los movimientos entre equipos de los distintos jugadores similares encontrados, atendiendo a diferentes motivos como crear un equipo más competitivo, buscar jugadores para tener recambios que aporten el mismo nivel de juego o tener que reemplazar a un jugador por otro por la venta.

Conclusiones y Líneas de trabajo futuras

El desarrollo de este TFM se ha realizado sobre las dos ramas principales sobre las que gira el Máste, como son la Inteligencia de Negocio y el Big Data. Por un lado, se ha desarrollado una herramienta para dar soporte en la toma de decisiones en un ámbito de negocio, en este caso en el mundo del fútbol, a la hora de afrontar el mejor fichaje posible con la idea de que dé el mejor rendimiento esperado dentro de un equipo. Por otro lado, en el desarrollo de este sistema se han pasado por todas las etapas de tratamiento de datos dentro de un proyecto de Big Data, como son los procesos ETL, el almacenamiento, así como la visualización final de los mismos, de manera que la información sea de utilidad y pueda aportar valor.

Además se han tenido en cuenta otras áreas, como la infraestructura para el Big Data a utilizar, el uso de una base de datos NOSQL orientada a grandes volúmenes de datos, así como de aplicar técnicas de aprendizaje no supervisado como es la del concepto de similaridad.

Acorde a los resultados obtenidos, vemos que el cálculo de la similaridad que se realiza con el sistema desarrollado se ve reflejado en los movimientos de mercado de los propios jugadores en la vida real que se han comparado. Algunos de los objetivos personales de la realización de este TFM era el aprendizaje del uso de herramientas como MongoDB o Docker a la hora de poder utilizarlas en un sistema real. Aunque la curva de aprendizaje de estas herramientas no es excesivamente grande, la adaptación a los nuevos conceptos sobre todo del almacenamiento NoSQL ha implicado un hándicap a la hora del desarrollo al no tener conocimientos sólidos previos de estas herramientas.

Como líneas de trabajo futuras, se abren varias vías a seguir, como pueden ser las que se indican a continuación:

- Una vez analizados y comprobados los resultados obtenidos con datos anteriores, se podría utilizar un origen de datos actualizado, por ejemplo, mediante técnicas de scrapping, para tener las estadísticas más recientes de los jugadores siempre actualizadas.
- Ampliar la funcionalidad del propio sistema. Por ejemplo, a la hora de buscar la similaridad, no se han tenido en cuenta en este caso los datos contractuales de los jugadores, ya que es información confidencial, sobre todo, en relación al sueldo. A razón de esto, se podría implementar un sistema que mediante técnicas de aprendizaje supervisado, pudiera estimar el valor del jugador, así como su salario, en base a su rendimiento.
- Siguiendo esta línea, otras opciones que se podrían añadir es valorar la actividad en redes sociales de los jugadores que sean una opción para fichar, por ejemplo mediante análisis de sentimientos en Twitter. De esta manera el club puede tener más información y comprobar, por ejemplo, si suelen estar envueltos en polémicas que puedan dañar la imagen del club o suelen generar conflictos, o por el contrario, pueden reforzar o mejorar la imagen del club si su actividad en las redes es positiva.
- Extender el uso del concepto de búsqueda de similaridad aplicado al deporte a otras áreas, como por ejemplo, baloncesto o rugby entre otros.

Apéndices

Apéndice A

Plan de Proyecto Software

A.1. Introducción

En este anexo se refleja la planificación temporal de las tareas realizadas durante el desarrollo delproyecto, así como un estudio de la viabilidad económica y legal del mismo.

A.2. Planificación temporal

Este trabajo se ha desarrollado dentro del marco del TFM del Máster en Inteligencia de Negocio y Big Data en Entornos Seguros. Acorde a la guía del mismo, el alcance y la duración del TFM se corresponde de 9 créditos ECTS, equivalentes a 225 horas.

El proyecto se inició el 1 de octubre de 2022 y finalizó el 24 de febrero de 2023, llevándose a cabo las diferentes tareas dentro de este intervalo de tiempo. El detalle de las tareas realizadas es el siguiente:

Investigación En esta fase se realizó una investigación de las diferentes posibilidades a la hora de implementar el proyecto, en cuanto a arquitectura, modelos y herramientas a utilizar para llevar a cabo con éxito la realización del mismo. Entre las técnicas y herramientas estudiadas en esta fase estaría el despliegue de la aplicación, la base de datos a utilizar o los servicios web entre otros.

Aprendizaje de herramientas En esta fase, una vez elegida la arquitectura y las herramientas y bibliotecas después del análisis de investigación

realizado en la etapa anterior, se realizó un estudio de las técnicas y herramientas concretas seleccionadas. Entre ellas destacar Docker para el despliegue del sistema, las librerías de Python para poder implementar los servicios web basados en FAISS y la base de datos MongoDB.

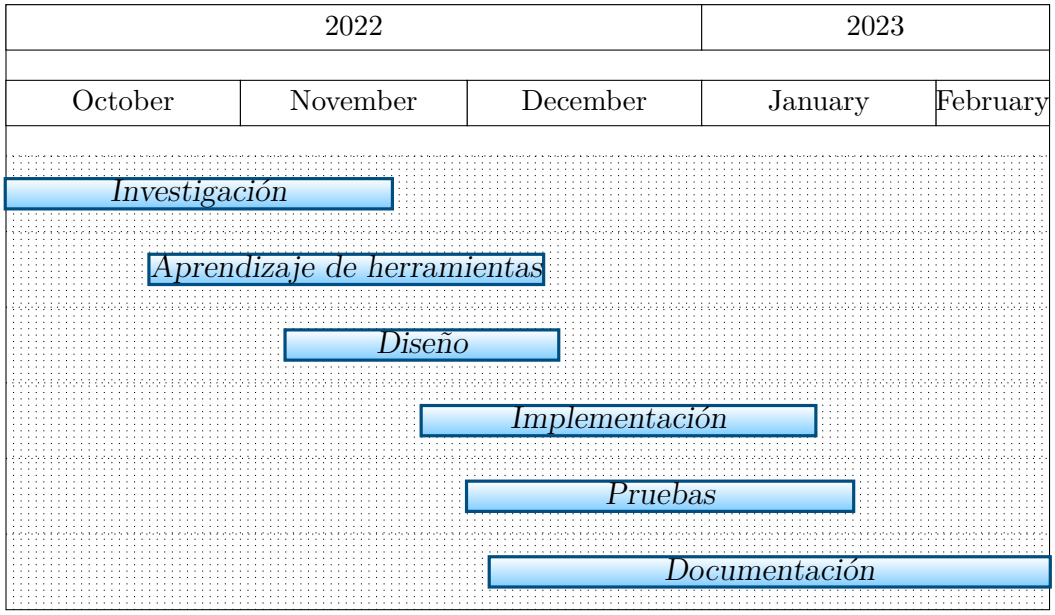
Diseño En esta fase, acorde a las decisiones tomadas en la fase de investigación, se diseñó el despliegue y la estructura del proyecto, teniendo en cuenta los componentes principales del mismo, como serían el almacenamiento de los datos, el despligue de los servicios webs y la aplicación web que consumirá los servicios web y a través de la cuál se visualizarán los resultados. De igual forma, se diseñó la forma de desplegar el sistema.

Implementación En esta fase se realizaron diferentes partes del proyecto, como son la ETL de los datos, el almacenamiento de los mismos, se desarrollaron los servicios web y finalmente la aplicación web. También se configuró y se realizó el despliegue del sistema.

Pruebas En esta fase se realizaron las pruebas para ir verificando el correcto funcionamiento de los diferentes desarrollos, e ir estableciendo puntos de control para confirmar el buen funcionamiento del sistema.

Documentación En esta fase se ha ido documentando tanto la memoria, los anexos y así como la documentación relacionada con el desarrollo del proyecto.

A continuación se muestra el diagrama de Gantt con el flujo temporal de estas fases:



A.3. Estudio de viabilidad

En esta sección se va a detallar la viabilidad económica y legar del desarrollo de este proyecto.

Viabilidad económica

A partir de la planificación anterior y los recursos técnicos utilizados, podemos elaborar una previsión de la viabilidad económica del proyecto desarrollado. Se van a diferenciar para ellos dos tipos de recursos, como son los recursos técnicos y los recursos humanos.

Para el desarrollo del proyecto, tanto a nivel documental como de codificación, se ha llevado a cabo utilizando un ordenador portátil personal con las siguientes características:

- Procesador AMD Ryzen 5 3550H
- Memoria RAM de 8GB DDR4 2400 MHz
- Disco de 512GB SSD M.2 PCIE NVME
- Disco de 1Tb HDD

Consideramos una amortización del ordenador personal de 5 años, siendo el precio total del mismo de 590€. La amortización anual del ordenador sería de 118€, lo que nos deja un coste mensual de 9,83€. Por tanto el coste asociado al tiempo de uso correspondería a 49,15€. En cuanto a gastos técnicos también incluimos el gasto de internet, que tiene un coste mensual de 33,50€. Por tanto, el coste total asociado a la duración del proyecto es de 167,50€.

Además, hay que tener en cuenta los recursos humanos utilizados. En este caso se tendrá en cuenta el rol de Analista Programador, encargado de analizar y desarrollar el código de las distintas implementaciones realizadas en el proyecto. Acorde al "XVII Convenio colectivo estatal de empresas de consultoría y estudios de mercado y de la opinión pública" [1] el sueldo bruto anual sería de 22.993,74€. En este caso se ha compatibilizado el desarrollo del proyecto con el trabajo diario, así que por simplicidad, se va a considerar la mitad de la jornada. El coste mensual a jornada completa sería de 1916,15€ y, por tanto, a media jornada, el coste se corresponde con 958,08€. El coste total de los cinco meses sería de 4790,40€. Finalmente, habría que añadirle el coste de la cotización a la seguridad social, que en este caso, acorde a las bases y tipos de cotización de 2022 [2] sería del 23,60 %, siendo éste un coste de

A continuación se destallan los costes en la siguiente tabla:

Descripción	**Meses**	**Coste mensual**	**Total**
Recursos técnicos			
Ordenador personal	5	€09.83	€49.15
Internet	5	€33.50	€167.50
Recursos humanos			
Sueldo Analista Programador	5	€958.08	€4790.40
Cotización seguridad social	5	€958.08	€1130.53
Coste total			**€6137.58**

Por tanto, el coste total del proyecto sería de 6137,58€.

[1] https://www.boe.es/boe/dias/2018/03/06/pdfs/BOE-A-2018-3156.pdf
[2] https://www.seg-social.es/Trabajadores/CotizacionRecaudacionTrabajadores/36537

Viabilidad legal

En esta sección se va a tratar la viabilidad legal del proyecto desarrollado como Trabajo Fin de Máster.

Teniendo en cuenta los desarrollos realizados, a continuación se detallan las librerías utilizadas junto con sus licencias de uso:

Libería	Licencia
FAISS	MIT License
Flask	BSD-3-Clause license
Flask Corss	MIT License
numpy	MIT License
pymongo	Apache License 2.0
pandas	BSD-3-Clause license

Tabla A.1: Librerías y licencias

Las imágenes de las banderas de los distintos países utilizadas en la aplicación web provienen de "Free Country Flags in SVG" [3], las cuáles se distribuyen bajo licencia MIT License.

Las imágenes de los jugadores y escudos de los equipos de fútbol utilizados en la web provienen de "Sofifa" [4], a la cuál se ha solicitado el uso de los mismos. Los diseños de las imágenes referentes a jugadores y escudos de equipos, así como los distintos nombres asociados a los mismos, pertenecen a sus respectivos propietarios. El uso de los mismos en este proyecto de Trabajo Fin de Máster universitario se realiza únicamente con fines docentes [5].

[3] https://flagicons.lipis.dev/
[4] https://sofifa.com/
[5] https://www.boe.es/buscar/act.php?id=BOE-A-1996-8930

Apéndice B

Especificación de diseño

B.1. Introducción

En esta sección se detallarán los diferentes diseños realizados en el desarrollo del sistema implementado en este proyecto.

B.2. Diseño procedimental

En este apartado se van a detallar los casos de uso y los diagramas de secuencia utilizados para el diseño del sistema.

En cuanto al caso de uso, correspondería con un usuario que realiza la petición para el cálculo de la similaridad a partir de un jugador.

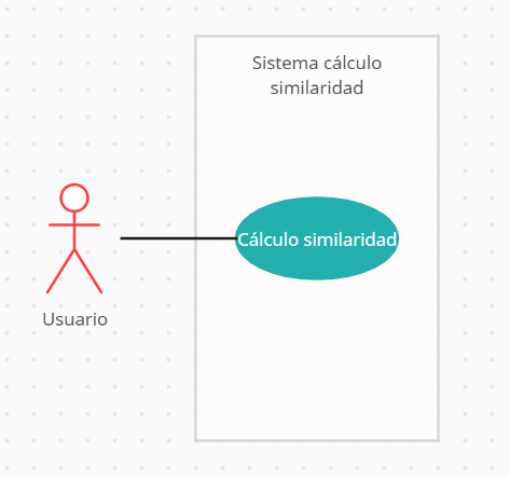


Figura B.1: Caso de uso cálculo de la similaridad

En el caso de uso, el usuario seleccionará el equipo, el jugador, el número de jugadores similares que desea recibir y las características que se tienen que tener en cuenta de esos jugadores. Una vez seleccionados estos parámetros, se solicitará al servicio web que calcule la similaridad y devuelva el resultado para visualizarlo. El diagrama de secuencia del proceso completo sería el siguiente:

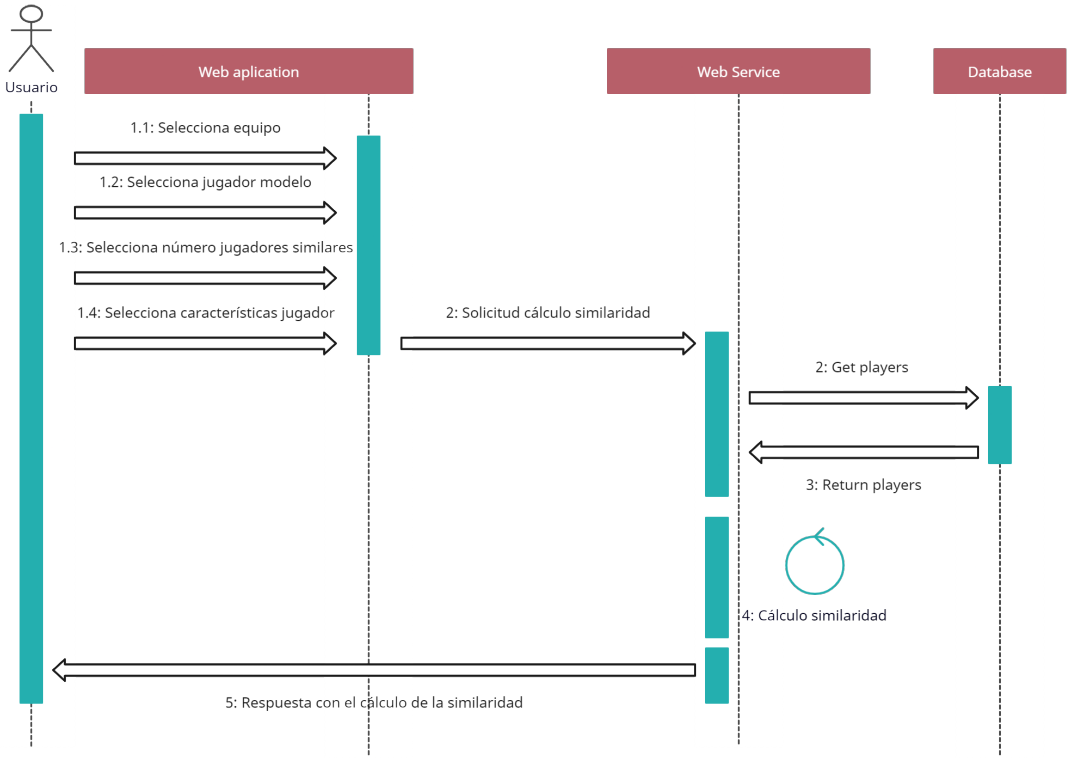


Figura B.2: Diagrama de secuencia cálculo de la similaridad

B.3. Diseño arquitectónico

En cuanto a al arquitectura empleada para el sistema desarrollado, se basa en una arquitectura cliente-servidor combinada con MVC (Modelo Vista Controlador).

Bajo la premisa de cliente-servidor, el servidor estaría compuesto por los servicios web que son los que reciben las solicitudes para calcular el resultado de la similaridad y devolver el mismo y la base de datos donde se almacena la información que utilizarán los servicios web para realizar los cálculos para llegar a ese resultado, mientas que el cliente en este caso, sería la aplicación web, que es quien realiza la solicitud y recibe la repuesta del servidor.

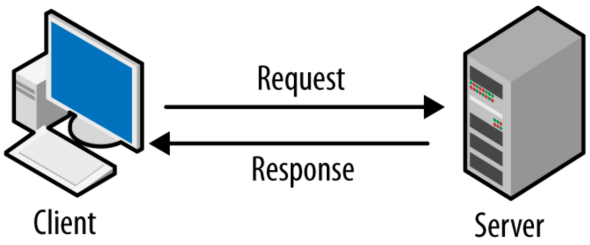


Figura B.3: Ejemplo de arquitectura cliente-servidor

Bajo la premisa del MVC, el modelo y el controlador correspondería a los servicios web, que son los que acceden a la cata de almacenamiento de datos y controlan los eventos de entrada como son las peticiones que llegan desde la vista, que en este caso seria la aplicación web.

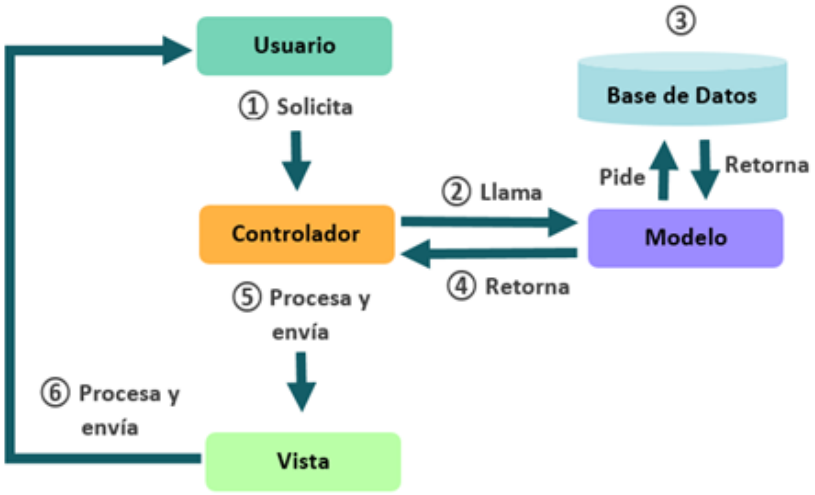


Figura B.4: Ejemplo de arquitectura MVC

Apéndice C

Documentación técnica de programación

C.1. Introducción

En esta sección se van a explicar los diferentes desarrollos realizados para la construcción del sistema que se ha implementado en este proyecto.

C.2. Estructura de directorios

La estructura de directorios del desarrollo de este proyecto es la siguiente:

```
MASTER
├── mongodb
│   └── collections
└── website
    ├── css
    ├── img
    │   ├── countries
    │   ├── players
    │   └── teams
    ├── js
    ├── scss
    └── vendor
```

El contenido de los directorios indicados anteriormente es el siguiente:

MASTER Es el directorio raíz que contiene al resto de directorios además de los ficheros encargados de la configuración para el despliegue de todos los elementos mediante Docker, así como el archivo de los servicios web.

mongodb Este directorio contiene el directorio con las colecciones que se cargarán en la base de datos MongoDB cuando se relice el despliegue mediante Docker.

website Este directorio contiene los ficheros y directorios relacionados con la aplicación web, como pueden ser las imágenes, el código JavaScritp o CSS empleado en la aplicación.

C.3. Manual del programador

En este apartado se van a detallar los desarrollos realizados en los diferentes elementos que componen el sistema total desarrollado.

Servicios Web

Para calcular la similaridad del jugador que se solicite y devolver los resultados. Para realizar esta operativa se han desarrollado unos servicios web utilizando Python y la librería Flask. Este desarrollo se encuentra en el archivo app.py.

Esos servicios web cuentan con tres métodos creados:

- getAllTeams
- getTeamPlayers
- getSimilarity

A continuación se explica cada uno de ellos.

getAllTeams

Este método no recibe ningún parámetro y devuelve en formato JSON todos los equipos de fútbol almacenados en el sistema ordenados alfabéticamente por su nombre.

```
@app.route("/getAllTeams")
def getAllTeams():

    db=""
    try:
        db = get_db()
        teamsCollection = db['teams']
        return
        ↪  dumps(teamsCollection.find({},{'_id':False}).sort("club_name"))

    except Exception as e:
        return jsonify({"errorMsg":"Failed: "+ str(e)})
    finally:
        if type(db)==MongoClient:
            db.close()
```

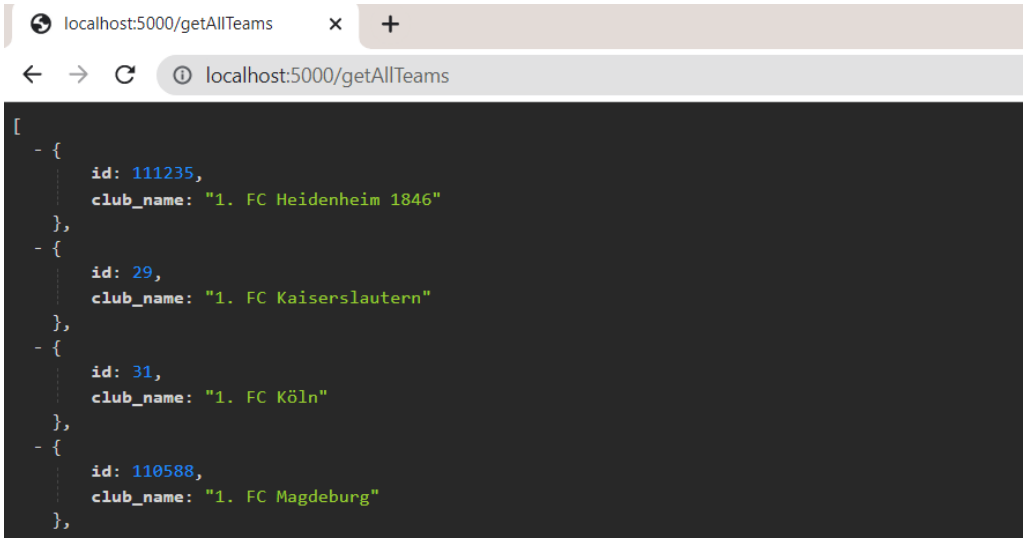


Figura C.1: Ejemplo de parte de los datos que devuelve el método getAllTeams

getTeamPlayers

Este método devuelve en formato JSON los datos de todos los jugadores del equipo que se recibe como parámetro.

```
db=""
    try:
        db = get_db()
```

```
        playersCollection = db['players']

        return dumps(playersCollection.find({"club_id":idTeam},
        ↪  {"short_name": 1, "id": 1, '_id':False}).sort("short_name"))

    except Exception as e:
        return jsonify({"errorMsg":"Failed: "+ str(e)})
    finally:
        if type(db)==MongoClient:
            db.close()
```

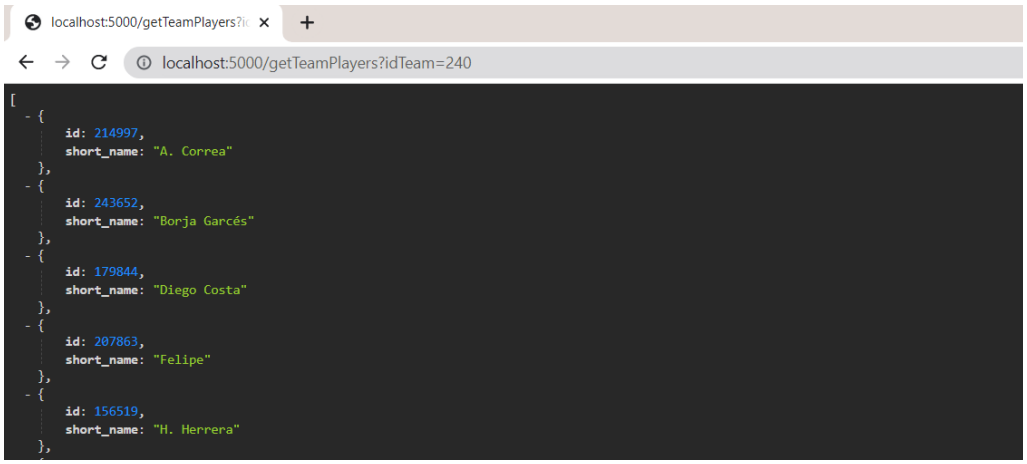


Figura C.2: Ejemplo de parte de los datos que devuelve el método getTeamPlayers

getSimilarity

Este método recibe como parámetros el id del jugador sobre el que se quiere buscar la similaridad,el número de los jugadores más similares que se tienen que obtener y las métricas de las características deportivas que tienen que tener los jugadores sobre los que realizar la búsqueda de la similaridad. Con estos datos, calcula la similaridad del jugador solicitado entre todos los jugadores almacenados en el sistema que cumplan las condiciones recibidas y devuelve en formato JSON los datos de los jugadores más similares encontrados.

En primer lugar se recogen los datos de MongoDB de todos los jugadores del sistema que cumplen con los requisitos recibidos para buscar la similaridad y se almacenan en un DataFrame.

Este DataFrame con los datos se duplica. El motivo es que en el primero guardaremos los datos sin manipular y este segundo DataFrame será utilizado para realizar las distintas operaciones sobre los datos. Como queremos buscar la similaridad en base a su rendimiento y condiciones deportivas, se eliminan del DataFrame los campos que no estén relacionados con el mismo.

```
playersDataFrame2.drop(['id','short_name','long_name','dob','nationality',
'nationality_iso','club_id','club_name','league_name','league_rank','value_eur',
'wage_eur','player_positions','preferred_foot','release_clause_eur',
'contract_valid_until'],axis=1, inplace=True)
```

Una vez realidas estas operaciones, convertimos este DataFrame a array en formato float32 de numpy.

En este punto ya podemos crear la query de consulta que usará FAISS con todos los datos que necesitamos para calcular la similaridad:

```
    # Creamos una función para crear la query que usaremos
# Los parámetros de entrada son:
# playersOri: Datos originales de los jugadores
# playersNumpy: Datos de los jugadores ya vectorizados
# idsJugadores: Jugadores de los que queremos encontrar elementos
↪  similares
# La función devuelve la query para que se pueda usar en FAISS

def creaQuery (playersOri, playersNumpy,idsJugadores):

    query = []

    for idj in idsJugadores:

        # Buscamos el índice en el que están
        i = playersOri[playersOri['id'] ==
        ↪  int(idj)].index.values.astype(int)[0]

        # Asignamos a la query los datos del jugador a buscar desde
        ↪  los datos vectorizados
        query.append(playersNumpy[i])

    # Hacemos cast y pasamos la lista a array de Numpy
    query = np.asarray(query)

    # Devolvemos la query
    return query
```

Finalmente se ha creado una función que ejecuta la búsqueda de la similaridad a partir de la query creada en el paso anterior:

```
    # Se crea la función buscaJugadoresSimilares
# Como parámetros de entrada tendrá:
# datos: Nuestros datos vectorizados previamente
# Los jugadores origen sobre los que queremos encontrar jugadores
↪  similares
# El número de jugadores similares que se tienen que encontrar para
↪  los jugadores recibido


def buscaJugadoresSimilares(datos, query, numJugadoresSimilares):

    # Calculamos el número de dimensiones
    dim = datos.shape[1]

    # Creamos un índice que será el tamaño de los vectores
    index = faiss.IndexFlatL2(dim)

    # Comprobamos si está entrenado
    #print(index.is_trained)

    # Añadimos al índice el conjunto de datos con ascontiguosarray
    ↪  para que funcione de manera correcta
    index.add(np.ascontiguousarray(datos))

    # queremos los k elementos más similares
    k = numJugadoresSimilares
    D, I = index.search(query,k)

    return D,I
```

Esta función nos devolverá la distancia de los jugadores respecto al jugador del que partimos, así como los índices de los jugadores más similares encontrados para localizar su posición en el vector.

En este punto ya tenemos las posiciones de los jugadores similares dentro del vector, pero al no tener sólo los datos de las métricas deportivas, necesitamos recuperar los datos originales de los jugadores. Para ello se ha creado una función que realiza esta operación y busca en el DataFrame inicial el índice de los jugadores encontrados, y de esta forma, recuperar la información del jugadores que está en esa posición:

```
    # Creamos una función que nos devolverá más datos (equipo,
    ↪  nombre, posición...) de los jugadores encontrados
# Tenemos el vector con sólo las estadísticas deportivas
# y nos interesa buscar todos los datos completos, como el equipo,
↪  la liga, etc que necesitemos, y esto lo tenemos en los datos
↪  originales

def datosJugadoresEncontrados(playersOri,indices, distancias):

    # Para devolver los datos buscados de los jugadores
    #datosJugadores =
    ↪  pd.DataFrame(columns=playersOri.columns.values)
    datosJugadores = pd.DataFrame
    ↪  (columns=playersDataFrame.columns.values)

    # Recuperamos de los datos originales la información que
    ↪  necesitamos
    for j in range(0, len(indices)):

        for i in range(0,len(indices[j])):

            # Fila con todos los datos del jugador encontrado
            datosJugador = playersOri.iloc[indices[j][i]]

            # Distancia
            distancia = distancias[j][i]

            #Añadimos la columna distancia con el valor a la row del
            ↪  jugador
            datosJugador['distancia'] = distancia

            datosJugadores = datosJugadores.append(datosJugador)

    return datosJugadores
```

En este punto ya tenemos los datos de los jugadores más similares que se han encontrado acorde a las características solicitadas, además de la distancia entre ellos. Estos datos se devuelven por parte del servicio web en formato JSON para que puedan ser consumidos por el cliente.

Aplicación Web

La aplicación web se ha desarrollado bajo el framework de Boostrap utilizando PHP, JavaScript y CSS para la maquetación de la página. En el directorio website mencionado anteriormente se encuentran los directorios y

archivos que contienen el código de la aplicación. Algunos ficheros a destacar serían los siguientes:

index.php Contiene el código de la página principal desarrollado en PHP. Contiene los elementos necesarios para poder realizar la petición y visualizar el resultado del cálculo de la similaridad.

similardiad.js Contiene el código en JavaScript que sirve de intermediario a la hora de realizar la petición entre la aplicación web y los servicios web ya comentados previamente.

C.4. Compilación, instalación y ejecución del proyecto

Para poder usar la aplicación es necesario tener instalado Docker previamente en el sistema. Por ejemplo, se puede descargar la versión Desktop de Docker desde el siguiente enlace seleccionando el sistema operativo en el que vayamos a utilizar la aplicación y siguiendo los pasos indicados en la propia página, bien descargando un ejecutable o ejecutando las líneas de comando indicadas a través de la consola.

Una vez instalado Docker e iniciado, sólo tenemos que posicionarnos en la carpeta principal del sistema (carpeta MASTER) a través del terminal y ejecutar el comando y ejecutar el siguiente comando:

```
docker-compose build --no-cache

docker-compose up
```

Una vez ejecutado este comando, simplemente tendremos que esperar a que termine la instalación, ya que comenzarán a descargarse de manera automática y se instalarán las dependencias y librerías necesarias para poder utilizar el sistema completo.

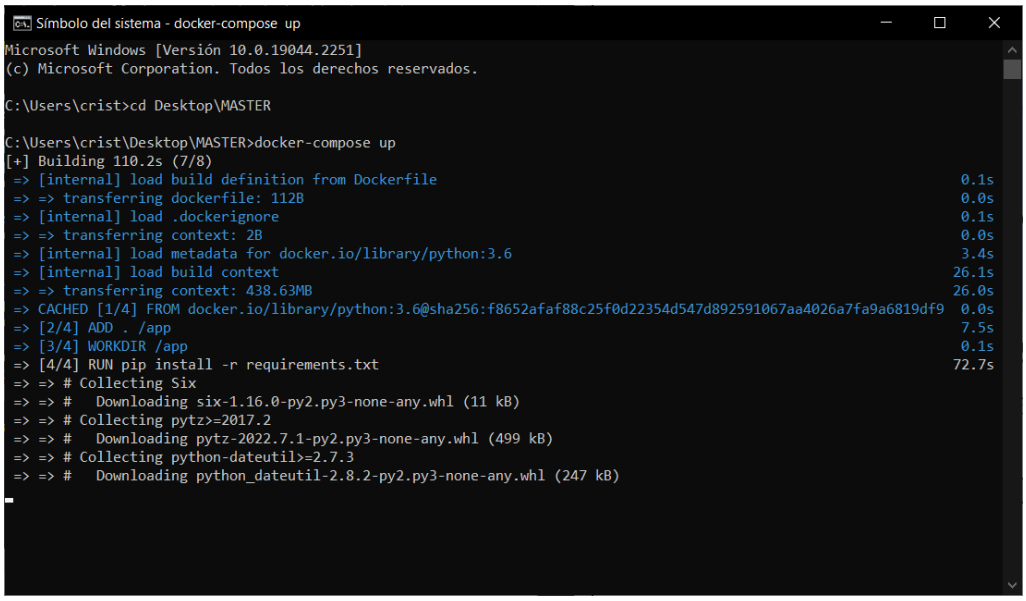


Figura C.3: Despliegue de la aplicación a través de Docker

También se realizará de manera automática la configuración de los distintos elementos que componen el sistema ya mencionados previamente en los diferentes puntos de la memora, como son los servicios web, la base de datos y la aplicación web. Una vez finalizado el despliegue, podremos ver en Docker el nuevo contenedor, que tendrá todos los elementos desplegados y ya disponibles para su uso:

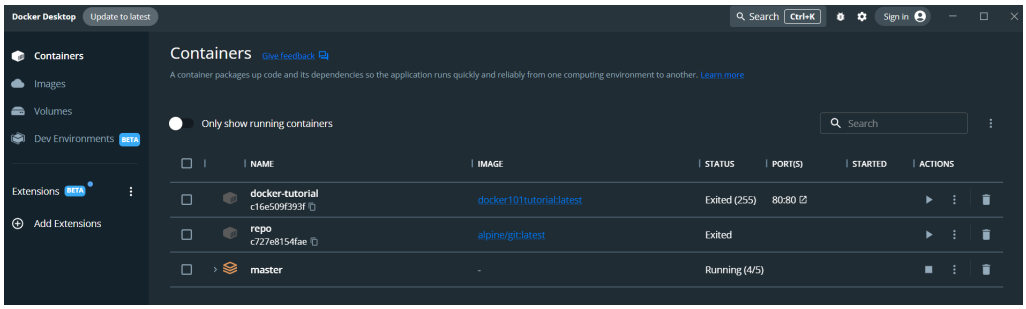


Figura C.4: Contenedor desplegado en Docker

Una vez finalizado el proceso, sólo tenemos que abrir nuestro navegador e ir a la dirección http://localhost:8080/ y ya podremos empezar a utilizar nuestro sistema de búsqueda de similaridad aplicada al scouting deportivo.

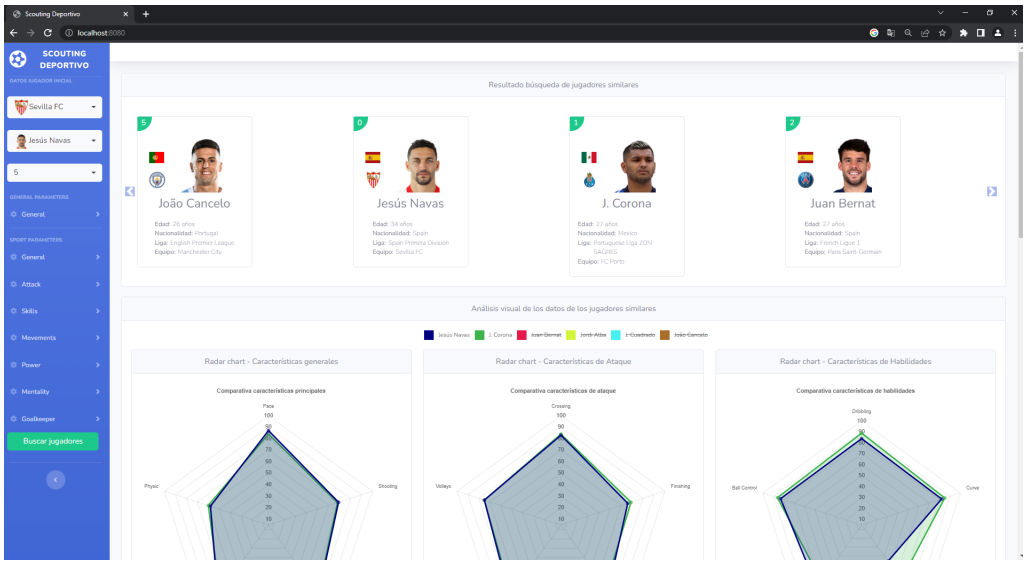


Figura C.5: Página principal de la aplicación web

Apéndice D

Documentación de usuario

D.1. Introducción

En esta sección se va a explicar la instalación y el modo de uso de la herramienta desarrollada en este trabajo fin de máster.

D.2. Instalación

Para poder usar la aplicación es necesario tener instalado Docker previamente en el sistema. Por ejemplo, se puede descargar la versión Desktop de Docker desde su web oficial, seleccionando el sistema operativo en el que vayamos a utilizar la aplicación y siguiendo los pasos indicados en la propia página, bien descargando un ejecutable o ejecutando las líneas de comando indicadas a través de la consola.

También será necesario descargar desde este repositorio, donde se han dejado disponibles los ejecutables necesarios, la carpeta MASTER, que contiene los archivos necesarios para desplegar el sistema.

Una vez instalado Docker e iniciado, sólo tenemos que posicionarnos en la carpeta principal del sistema (carpeta MASTER) a través del terminal y ejecutar el comando y ejecutar el sigiuente comando:

```
docker-compose build --no-cache

docker-compose up
```

Una vez ejecutado este comando, simplemente tendremos que esperar a que termine la instalación, ya que comenzarán a descargarse de manera automática y se instalarán las dependencias y librerías necesarias para poder utilizar el sistema completo.

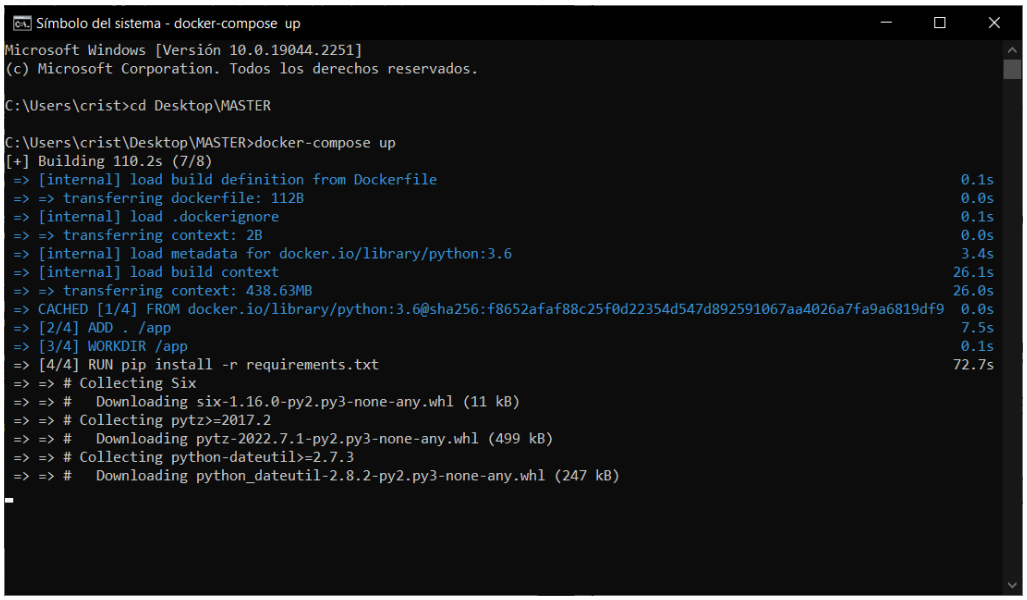


Figura D.1: Despligue de la aplicación a través de Docker

También se realizará de manera automática la configuración de los distintos elementos que componen el sistema ya mencionados previamente en los diferentes puntos de la memora, como son los servicios web, la base de datos y la aplicación web. Una vez finalizado el despliegue, podremos ver en Docker el nuevo contenedor, que tendrá todos los elementos desplegados y ya disponibles para su uso:

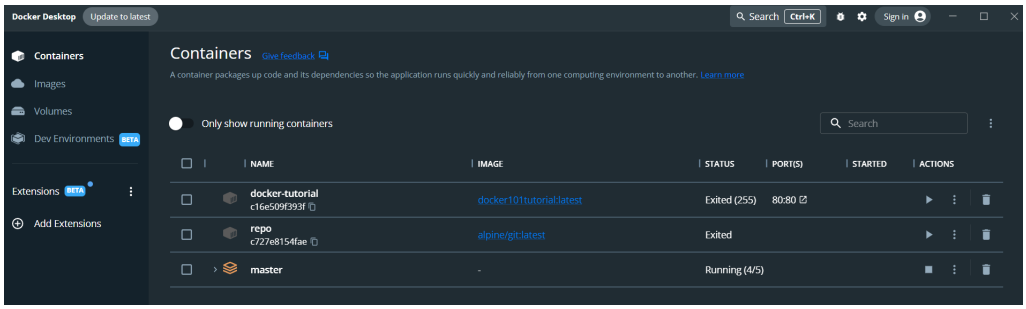


Figura D.2: Contenedor desplegado en Docker

Una vez finalizado el proceso, sólo tenemos que abrir nuestro navegador e ir a la dirección http://localhost:8080/ y ya podremos empezar a utilizar nuestro sistema de búsqueda de similaridad aplicada al scouting deportivo.

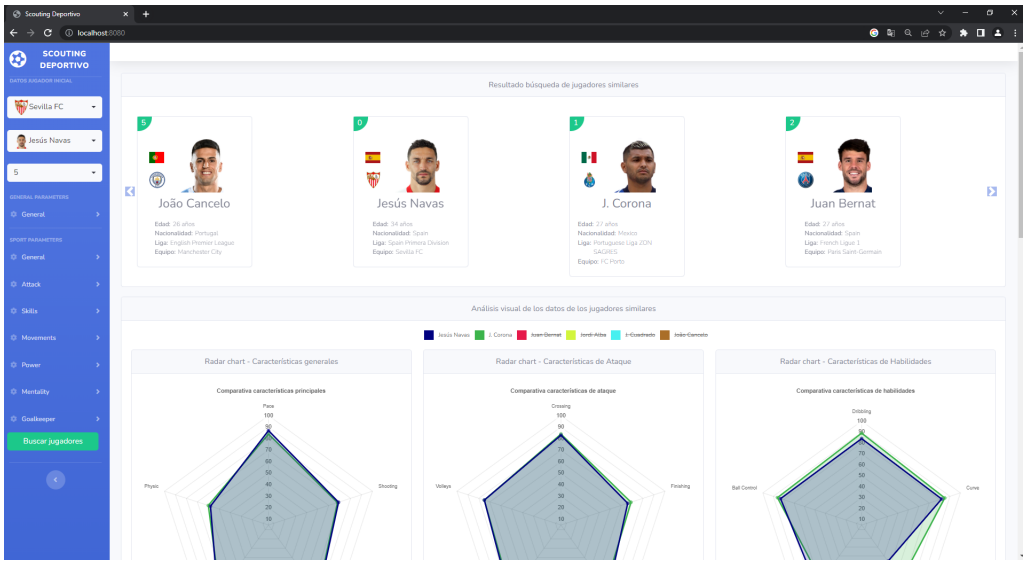


Figura D.3: Página principal de la aplicación web

D.3. Manual del usuario

A continuación se detallará el uso de la aplicación web para buscar la similaridad de un jugador que tendremos como referencia.

En la parte izquierda de la aplicación web, se puede ver un menú lateral desde el que se seleccionará el equipo, el jugador del que partimos para buscar la similaridad y el número de jugadores similares que queremos que nos devuelva. Una vez informado el equipo de fútbol del jugador del que partimos, automáticamente se cargarán todos los jugadores de ese equipo en el siguiente campo de selección. El siguiente paso es elegir el número de jugadores similares que queremos que nos devuelva como resultado. Será obligatorio informar estos tres campos y en caso de que no estén informados y se solicite buscar la similaridad, aparecerá un mensaje informándonos de esto.

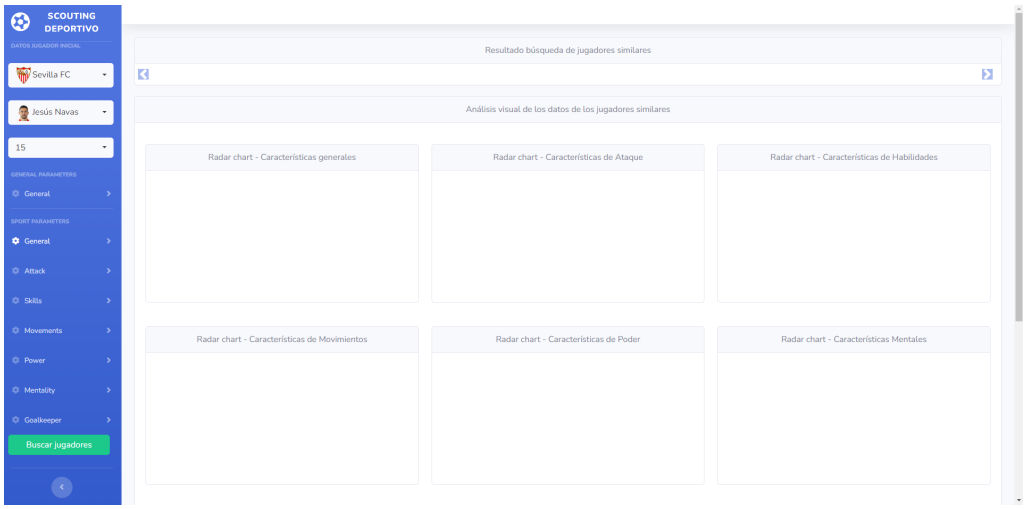


Figura D.4: Ejemplo selección jugador sobre el que se parte

Tanto a la hora de buscar el equipo como el jugador, se puede utilizar una ayuda de búsqueda a la hora de localizar estos datos, escribiendo parte del nombre del dato a buscar. Esto hará que se filtren los equipos y los jugadores que tengan coincidencia con lo escrito.

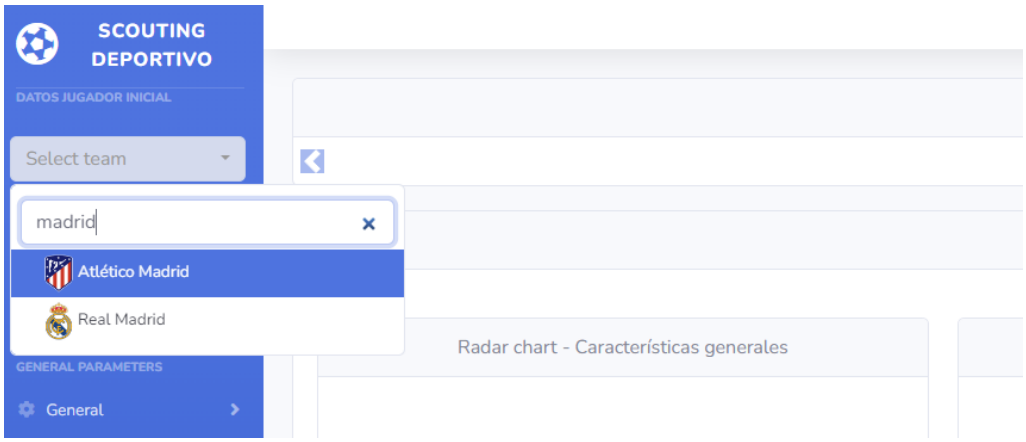


Figura D.5: Ejemplo ayuda de búsqueda

Además, en el mismo menú lateral, podremos filtrar las características deportivas que tienen que tener los jugadores sobre los que se busque esa similaridad. De esta forma, podemos personalizar más la búsqueda, ajustarla y adaptarla a nuestros requisitos.

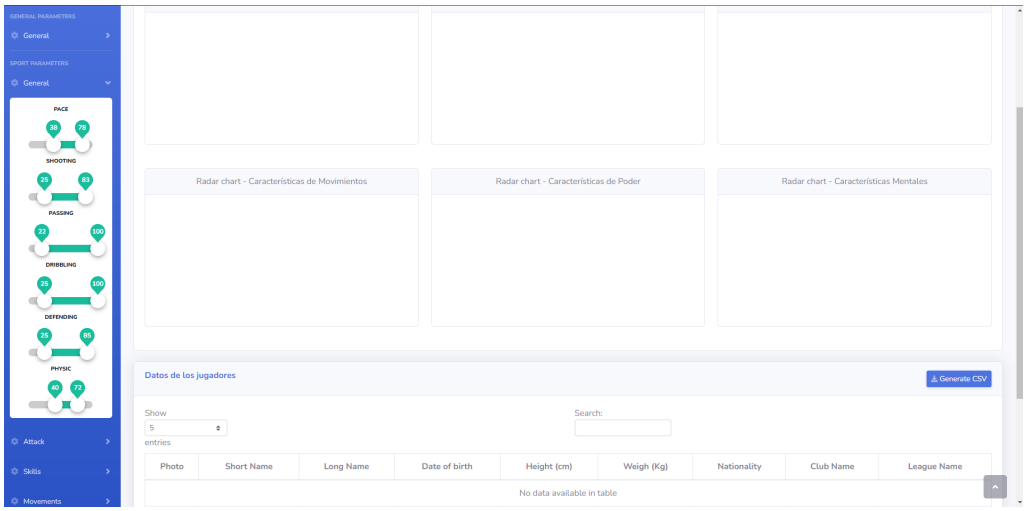


Figura D.6: Ejemplo selección características deportivas

Esta barra lateral también cuenta con la opción de contraerla, para que por ejemplo, una vez seleccionados los datos, se pueda ganar más campo

de visión sobre los mismo. Esto se realiza mediante el botón inferior de la misma.

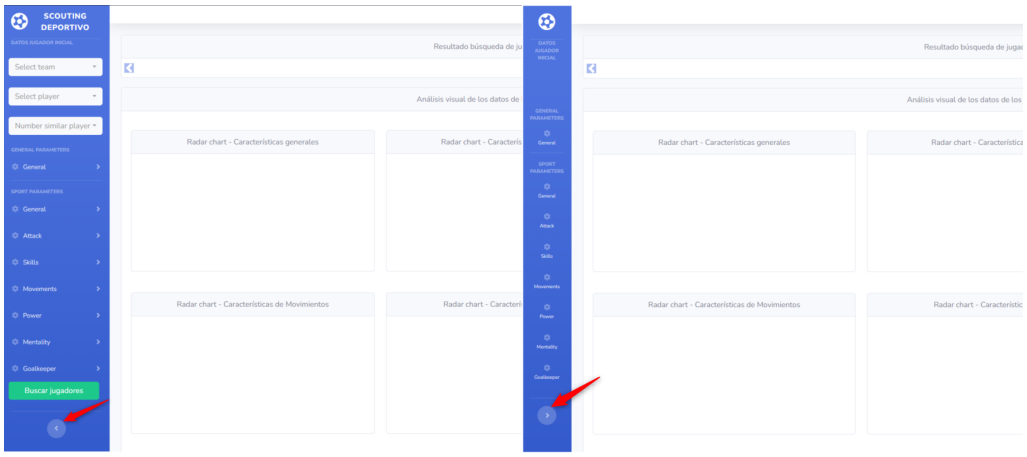


Figura D.7: Comparativa barra lateral expandida y contraída

Una vez seleccionados estos datos podemos ejecutar la búsqueda y nos devolverá los datos de los jugadores similares encontrados acorde a las características que hemos indicado.

En la primera parte de la página podremos ver un carrousel con el resumen de los datos de los jugadores similares localizados.

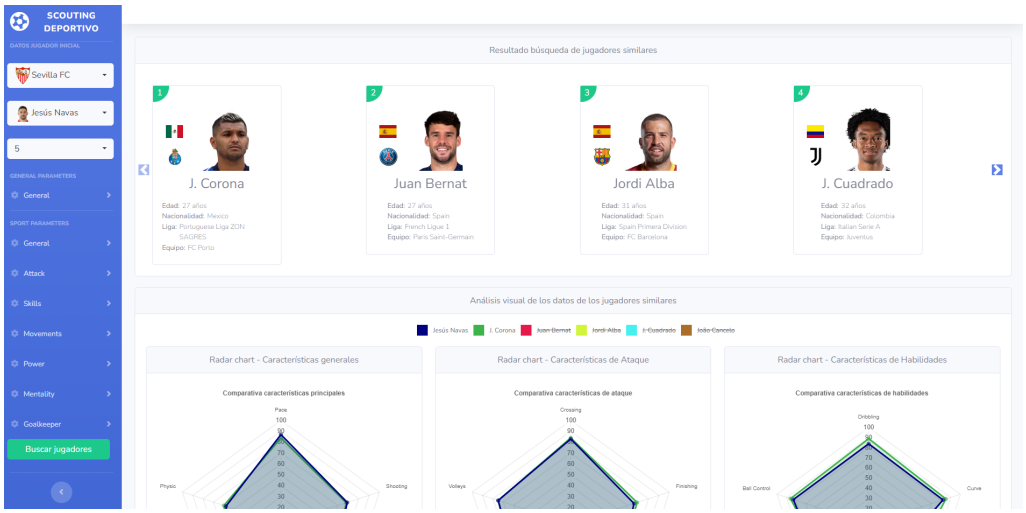


Figura D.8: Carrousel con los jugadores similares encontrados

En la siguiente parte de la página se han añadido seis radar charts para poder comparar las diferencias de similaridad en cuanto a las diferentes características de los jugadores de manera visual.

Hay un radar chart por cada familia de características deportivas:

- Dimensiones generales del rendimiento deportivo
- Dimensiones de ataque del rendimiento deportivo
- Dimensiones de habilidades del rendimiento deportivo
- Dimensiones de movimiento del rendimiento deportivo
- Dimensiones de fuerza/resistencia del rendimiento deportivo
- Dimensiones de mentalidad del rendimiento deportivo
- Dimensiones de habilidades de portero del rendimiento deportivo

Para el caso de las dimensiones relacionadas con habilidades de portero, sólo se mostrarán cuando la posición del jugador seleccionado como modelo sobre el que buscar la similaridad sea un portero, ya que son los únicos para los que existe esta información de habilidades. En esos casos, se mostrarán estos datos en la gráfica de características generales.

Todas las gráficas se han creado con la misma escala para que no haya interpretaciones erróneas de los datos. De igual forma, se mantiene el mismo color para el mismo jugadores en las diferentes gráficas.

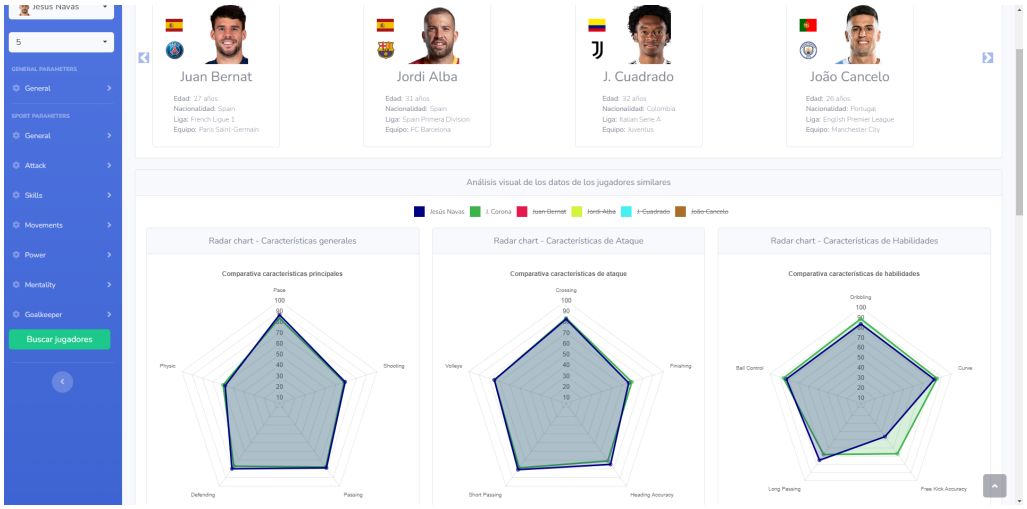


Figura D.9: Radar charts con los datos de los jugadores

Inicialmente se muestran representados en las gráficas los valores del jugador sobre el que partimos a la hora de buscar la similaridad y del jugador más similar encontrado, para que sea más legible. Se deja en manos del usuario que pueda seleccionar o des seleccionar los jugadores que considere a la hora de hacer la comparación. Esta selección de jugadores se realiza desde el menú superior de la sección de las gráficas y aplica para todas las gráficas al mismo tiempo.

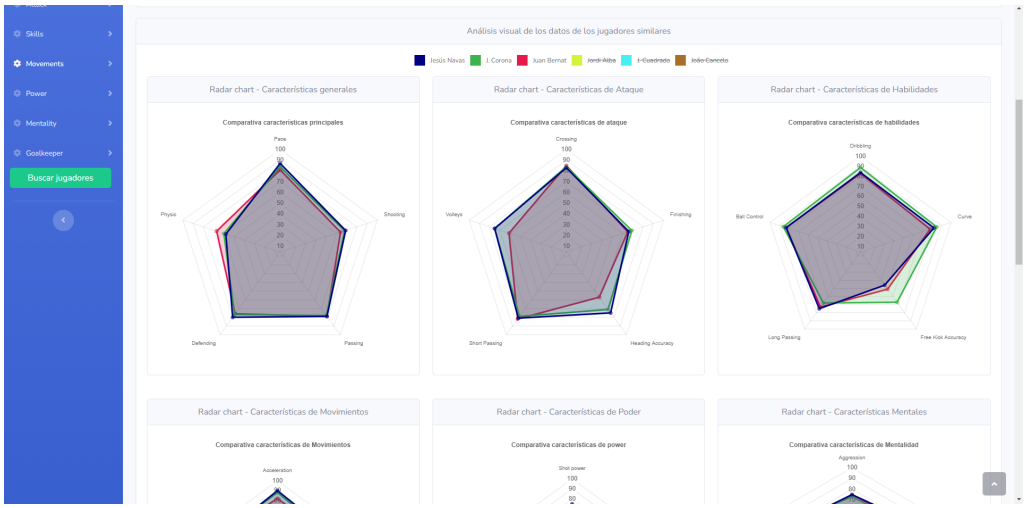


Figura D.10: Selección de jugadores a mostrar en las gráficas

En la parte inferior de la página aparecerán los datos de los jugadores en formato tabla, completando la información relacionada con los mismos. Podremos ir navegando por las diferentes partes de la tabla a la hora de visualizar la información, así como por las diferentes páginas de la misma, y además, podremos seleccionar el número de resultados que se visualizan por cada página.

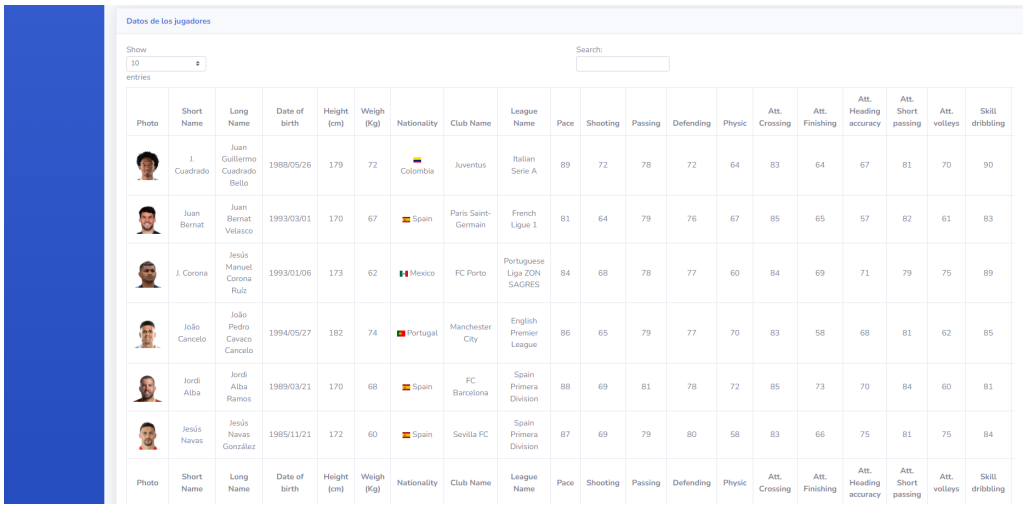


Figura D.11: Visualización de los datos de los jugadores en formato tabla

Bibliografía

[1] Khalid Adam, Mohammed Adam, Ibrahim Fakharaldien, Jasni Mohamad Zain, Mazlina Majid, and Noraziah Majid. Bigdata: Issues, challenges, technologies and methods. 04 2015.

[2] Rajaraman Anand and Ullman Jeffrey David. *Mining of massive datasets*. Cambridge University Press, 2011.

[3] Engineering at Meta. Faiss: A library for efficient similarity search, 2017. [Internet; descargado 5-febrero-2023]. URL: https://engineering.fb.com/2017/03/29/data-infrastructure/faiss-a-library-for-efficient-similarity-search/.

[4] Aníbal Bregón Bregón. *Almacenamiento escalable, Fundamentos de NoSQL*. Universidad de Valladolid, 2022.

[5] Real Madrid CF. Comunicado oficial: Alaba, 2021. [Internet; descargado 6-febrero-2023]. URL: https://www.realmadrid.com/noticias/2021/05/28/comunicado-oficial-alaba.

[6] Villarreal CF. Noticias generales, el club ejerce la opción de compra sobre andrés, 2017. [Internet; descargado 6-febrero-2023]. URL: https://villarrealcf.es/el-club-ejerce-la-opcion-de-compra-sobre-andres/.

[7] Villarreal CF. Noticias generales, el villarreal y el huesca acuerdan el traspaso de andrés, 2020. [Internet; descargado 6-febrero-2023]. URL: https://villarrealcf.es/el-villarreal-y-el-huesca-acuerdan-el-traspaso-de-andres/.

[8] Villarreal CF. Noticias generales, ¡gero rulli ya es groguet!, 2020. [Internet; descargado 6-febrero-2023]. URL: https://villarrealcf.es/gero-rulli-ya-es-groguet/.

[9] Atlético de Madrid. Noticias, ¡bienvenido, joão félix!, 2019. [Internet; descargado 6-febrero-2023]. URL: https://www.atleticodemadrid.com/noticias/bienvenido-jo-o-felix.

[10] Atlético de Madrid. Noticias, ¡bienvenido, cunha!, 2021. [Internet; descargado 6-febrero-2023]. URL: https://www.atleticodemadrid.com/noticias/bienvenido-matheus-cunha.

[11] Inc. Docker. Docker docs, guides, docker overview, 2023. [Internet; descargado 5-febrero-2023]. URL: https://docs.docker.com/get-started/overview/.

[12] Inc. Docker. Use containers to build, share and run your applications, 2023. [Internet; descargado 5-febrero-2023]. URL: https://www.docker.com/resources/what-container/.

[13] Sevilla FC. Noticias, el mexicano jesÚs manuel corona, refuerzo invernal para el sevilla fc, 2022. [Internet; descargado 6-febrero-2023]. URL: https://sevillafc.es/actualidad/noticias/fichaje-jesus-corona-2022.

[14] Fifplay. Fifplay enciclopedia, 2023. [Internet; descargado 6-febrero-2023]. URL: https://www.fifplay.com/encyclopedia/#number.

[15] Paris Saint Germain. Sergio ramos firma con el paris saint-germain hasta 2023, 2021. [Internet; descargado 6-febrero-2023]. URL: https://es.psg.fr/equipos/primer-equipo/content/sergio-ramos-firma-con-el-paris-saint-germain-hasta-2023.

[16] Nielsen Norman Group. F-shaped pattern for reading web content (original study), 2006. [Internet; descargado 8-febrero-2023]. URL: https://www.nngroup.com/articles/f-shaped-pattern-reading-web-content-discovered/.

[17] IPython. Built-in magic commands, 2023. [Internet; descargado 8-febrero-2023]. URL: https://ipython.readthedocs.io/en/stable/interactive/magics.html.

[18] Itop. Pentaho, 2023. [Internet; descargado 5-febrero-2023]. URL: https://www.itop.es/pentaho.html.

[19] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2019.

[20] Jupyter. Jupyter notebooks, 2023. [Internet; descargado 8-febrero-2023]. URL: https://jupyter.org/.

[21] Stefano Leone. Fifa 21 complete player dataset, 2021. [Internet; descargado 5-febrero-2023]. URL: https://www.kaggle.com/datasets/stefanoleone992/fifa-21-complete-player-dataset?select=players_21.csv.

[22] Wilson Mar. Graph databases, 2021. [Internet; descargado 5-febrero-2023]. URL: https://wilsonmar.github.io/graph-databases/.

[23] Medium. Visual hierarchy, gutenberg diagram, f & z pattern, 2019. [Internet; descargado 8-febrero-2023]. URL: https://lineindesign.medium.com/be-a-designer-who-can-also-help-with-writing-copy-2f4ea02a5646.

[24] Microsoft. Non-relational data and nosql, 2023. [Internet; descargado 5-febrero-2023]. URL: https://learn.microsoft.com/en-us/azure/architecture/data-guide/big-data/non-relational-data.

[25] Inc. MongoDB. Compass. the gui for mongodb, 2023. [Internet; descargado 6-febrero-2023]. URL: https://www.mongodb.com/products/compass.

[26] Inc MongoDB. Performance best practices: Indexing, 2023. [Internet; descargado 7-febrero-2023]. URL: https://www.mongodb.com/blog/post/performance-best-practices-indexing.

[27] Inc MongoDB. Query optimization, covered query, 2023. [Internet; descargado 7-febrero-2023]. URL: https://www.mongodb.com/docs/manual/core/query-optimization/#std-label-read-operations-covered-query.

[28] Inc. MongoDB. Understanding the different types of nosql databases, 2023. [Internet; descargado 5-febrero-2023]. URL: https://www.mongodb.com/scale/types-of-nosql-databases.

[29] Redis. What is a key-value database?, 2023. [Internet; descargado 5-febrero-2023]. URL: https://redis.com/nosql/key-value-databases/.

[30] Toby Segaran. *Programming collective intelligence: building smart web 2.0 applications*. .º'Reilly Media, Inc.", 2007.

[31] Krypton Solid. Tipos de bases de datos nosql explicados: almacén de valores clave, 2023. [Internet; descargado 5-febrero-2023]. URL: https://kryptonsolid.com/tipos-de-bases-de-datos-nosql-explicados-almacen-de-valores-clave/.

[32] Todoultimateteam. Guia de atributos – fifa ultimate team, 2017. [Internet; descargado 6-febrero-2023]. URL: https://www.todoultimateteam.com/guia-de-atributos-fifa-ultimate-team/.

[33] Google Trends. Tendencia de diferentes bases de datos nosql, 2023. [Internet; descargado 5-febrero-2023]. URL: https://trends.google.es/trends/explore?date=all&q=%2Fm%2F04f32m3,%2Fm%2F05z_r2n,%2Fm%2F0h_bxxk,%2Fm%2F0b76t3s,%2Fm%2F03c4_2p.

[34] Sasha Trubetskoy. List of 20 simple, distinct colors, 2017. [Internet; descargado 8-febrero-2023]. URL: https://sashamaps.net/docs/resources/20-colors/.

[35] Hitachi Vantara. Data integration and analytics powered by pentaho., 2023. [Internet; descargado 5-febrero-2023]. URL: https://www.hitachivantara.com/en-us/products/dataops-software/data-integration-analytics.html.

[36] Wikipedia. Desigualdad triangular — wikipedia, la enciclopedia libre, 2022. [Internet; descargado 5-febrero-2023]. URL: https://es.wikipedia.org/w/index.php?title=Desigualdad_triangular&oldid=144876994.

[37] Wikipedia. Espacio euclídeo — wikipedia, la enciclopedia libre, 2022. [Internet; descargado 5-febrero-2023]. URL: https://es.wikipedia.org/w/index.php?title=Espacio_eucl%C3%ADdeo&oldid=148086313.

[38] Wikipedia. Iso 3166-1 alfa-2 — wikipedia, la enciclopedia libre, 2022. [Internet; descargado 5-febrero-2023]. URL: https://es.wikipedia.org/w/index.php?title=ISO_3166-1_alfa-2&oldid=148111720.