UNIVERSITAT POLITÈCNICA DE VALÈNCIA

_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería de Telecomunicación

Optimización del Rendimiento Futbolístico: Análisis Integral con Statsbomb, SQL y Power BI para un caso práctico real de un partido del mundial de 2018

Trabajo Fin de Grado

Grado en Tecnología Digital y Multimedia

AUTOR/A: Rivilla Rodriguez, Jaume

Tutor/a: Guerola Navarro, Vicente

CURSO ACADÉMICO: 2023/2024

Escuela Técnica Superior de Ingeniería de Telecomunicación
Universitat Politècnica de València
Edificio 4D. Camino de Vera, s/n, 46022 Valencia
Tel. +34 96 387 71 90, ext. 77190
www.etsit.upv.es

Resumen

En este proyecto, se aborda la optimización del rendimiento en el fútbol mediante un enfoque integral de análisis de datos. Se utilizará la API Statsbomb de Python para recopilar datos detallados sobre jugadores, equipos y eventos de partidos. La justificación radica en la necesidad creciente de tomar decisiones informadas en el ámbito deportivo, donde el análisis de datos puede proporcionar insights valiosos para mejorar el rendimiento individual y colectivo. Se emplearán técnicas de análisis de datos avanzadas y métricas de rendimiento, para extraer información relevante. Además, se implementará una base de datos en SQL para almacenar y gestionar los datos de los jugadores seleccionados. La utilización de Power BI permitirá la creación de visualizaciones interactivas y atractivas que faciliten la interpretación de los resultados. Esta metodología integrada brindará a entrenadores, jugadores y directivos una visión más profunda y objetiva de los factores que influyen en el desempeño en el fútbol profesional. En resumen, el proyecto combina tecnologías de vanguardia y metodologías analíticas para mejorar la toma de decisiones en el mundo del fútbol más concretamente en un caso práctico real de un partido del mundial de 2018.

Resum

This project addresses football performance optimization through a comprehensive data analysis approach. The Python Statsbomb API will be utilized to gather detailed data on players, teams, and match events. The justification lies in the growing need for informed decision-making in sports, where data analysis can provide valuable insights for improving both individual and collective performance. Advanced data analysis techniques and performance metrics will be employed to extract relevant information. Additionally, an SQL database will be implemented to store and manage data for selected players. The use of Power BI will enable the creation of interactive and engaging visualizations to facilitate result interpretation. This integrated methodology will provide coaches, players, and managers with a deeper and more objective understanding of the factors influencing performance in professional football. In summary, the project combines cutting-edge technologies and analytical methodologies to enhance decision-making in the football world, specifically focusing on a real practical case of a 2018 World Cup match.

Abstract

En aquest projecte, s'aborda l'optimització del rendiment en el futbol mitjançant un enfocament integral d'anàlisi de dades. Es farà servir l'API Statsbomb de Python per a recopilar dades detallades sobre jugadors, equips i esdeveniments de partits. La justificació rau en la necessitat creixent de prendre decisions informades en l'àmbit esportiu, on l'anàlisi de dades pot proporcionar idees valuoses per a millorar el rendiment individual i col·lectiu. Empraran tècniques d'anàlisi de dades avançades i mètriques de rendiment, per a extreure informació rellevant. A més, s'implementarà una base de dades en SQL per a emmagatzemar i gestionar les dades dels jugadors seleccionats. L'ús de Power BI permetrà la creació de visualitzacions interactives i atractives que faciliten la interpretació dels resultats. Aquesta metodologia integrada brindarà a entrenadors, jugadors i directius una visió més profunda i objectiva dels factors que influeixen en el rendiment en el futbol professional. En resum, el projecte combina tecnologies de punta i metodologies analítiques per a millorar la presa de decisions en el món del futbol, específicament en un cas pràctic real d'un partit del mundial de 2018.

RESUMEN EJECUTIVO

La memoria del TFG del Grado en Tecnología Digital y Multimedia debe desarrollar en el texto los siguientes conceptos, debidamente justificados y discutidos, centrados en el ámbito de la tecnologías digitales y multimedia

CONCEPT (ABET)	**CONCEPTO (traducción)**	**¿Cumple? (S/N)**	**¿Dónde? (páginas)**
1. IDENTIFY:	1. IDENTIFICAR:		
1.1. Problem statement and opportunity	1.1. Planteamiento del problema y oportunidad	S	1
1.2. Constraints (standards, codes, needs, requirements & specifications)	1.2. Toma en consideración de los condicionantes (normas técnicas y regulación, necesidades, requisitos y especificaciones)	S	1 - 39
1.3. Setting of goals	1.3. Establecimiento de objetivos	S	1
2. FORMULATE:	2. FORMULAR:		
2.1. Creative solution generation (analysis)	2.1. Generación de soluciones creativas (análisis)	S	7 - 36
2.2. Evaluation of multiple solutions and decision-making (synthesis)	2.2. Evaluación de múltiples soluciones y toma de decisiones (síntesis)	S	7 - 36
3. SOLVE:	3. RESOLVER:		
3.1. Fulfilment of goals	3.1. Evaluación del cumplimiento de objetivos	S	37
3.2. Overall impact and significance (contributions and practical recommendations)	3.2. Evaluación del impacto global y alcance (contribuciones y recomendaciones prácticas)	S	37 - 38

A mi familia, por darme todo el apoyo y herramientas para mi formación

A mi tutor Vicente, por la dedicación y pasión por su trabajo

Índice

Índice de Figuras

1. Introducción

El análisis de datos en el fútbol es una herramienta esencial para mejorar el rendimiento deportivo. En este proyecto, se utilizará la librería de Statsbombpy con Python para recopilar datos detallados sobre jugadores, equipos y eventos de partidos. También, se recopilarán datos y se almacenar en una base de datos MySQL.

Además, se utiliza Power BI como interfaz visual para que esos datos recopilados sean más visuales, tengan más sentido y faciliten la interpretación de los resultados. Esta metodología integrada ofrecerá a entrenadores, jugadores y directivos una mejor visión y más objetiva del desempeño en el fútbol profesional.

El proyecto aplicará a un caso real de un partido del Mundial de 2018 en concreto de Portugal vs España, demostrando cómo el análisis de datos puede transformar la toma de decisiones en el fútbol.

1.1 Objetivos del proyecto

El objetivo principal de este trabajo es analizar datos de fútbol utilizando la librería Statsbombpy con Python y SQL para posteriormente visualizarlos de manera efectiva en Power BI.

Se realizan los siguientes pasos:

- Comprender la librería Statsbombpy: Entender su funcionamiento para poder utilizarlo con Jupyter Notebook utilizando Python, recopilar datos sobre partidos, jugadores y equipos y poder exportarlos para que posteriormente puedan ser analizados en Power BI.
- Crear una base de datos en MySQL: Almacenar datos de los jugadores de la web de Sofascore para poder utilizar los datos en Power BI.
- Visualizar los resultados: Utilizar Power BI para crear dash (Matplotlib, 2024)boards interactivos que faciliten la interpretación de los datos tanto para entrenadores, directivos, jugadores y para personas interesadas en el mundo del fútbol.

1.2 Contexto y justificación

En todos los deportes la competitividad y el nivel de exigencia a jugadores, entrenadores y equipos son cada vez mayores y cada vez se dejan menos cosas al azar. Con el avance tecnológico que estamos viendo, un deporte como el fútbol que genera tanto dinero y que tiene tanta presión, cada vez son más los equipos profesionales y amateurs que dedican una parte de su presupuesto anual para crear un equipo enfocado a analizar datos tanto de su propio equipo como el de los equipos rivales.

A nivel personal en este Trabajo Final de Grado quería combinar todo lo que he ido aprendiendo durante estos años, tanto en la universidad como fuera de ella, y mi *hobbie* desde pequeño, que es el fútbol. He disfrutado de este deporte tanto practicándolo como viéndolo y ahora espero seguir disfrutando, analizando los datos.

2. Fundamentos teóricos

Antes de profundizar en los detalles del desarrollo del proyecto, es fundamental entender y comprender varios conceptos clave.

En primer lugar, se definirá el análisis de datos en el ámbito del fútbol. Además, se explorarán librerías utilizadas en el desarrollo del proyecto y más en detalle la librería de Statsbombpy, explicando su funcionamiento y cómo facilita la recopilación de datos detallados sobre competiciones, jugadores, equipos y eventos de los partidos.

Por último, se dedicará un apartado exclusivo a la implementación de la base de datos en SQL y al uso de Power BI. Se proporcionará una introducción a estas herramientas y se explicarán las diferentes funcionalidades que se utilizarán en el proyecto, para que se entienda mejor la base del análisis y la visualización de los datos.

2.1 Análisis de datos en el ámbito del fútbol

La tecnología de análisis de datos en el ámbito del fútbol se refiere al uso de herramientas y técnicas para recopilar, procesar y analizar grandes volúmenes de datos generados tanto en partidos como en los entrenamientos. En estos datos se incluyen estadísticas sobre jugadores, equipos y lo más importante, eventos que ocurren durante el partido. Los eventos de un partido se refieren a las distintas acciones que ocurren durante el partido o entrenamiento, estos eventos son recopilados para posteriormente poder hacer el análisis. Algunos ejemplos de medidas de estos eventos son:

- Gráficos de pases
- Gráficos de tiros
- Gráficos de calor
- Goles
- Porcentaje de posesión
- Número de pérdidas de balón
- Éxito en el regate
- Ubicación de los despejes
- Etc

2.2 Python para el Análisis de Datos

Python es un lenguaje de programación de alto nivel, entre otras cosas es muy utilizado para el análisis de datos debido a su sencillez y la cantidad de librerías especializadas disponible para este ámbito (Python, 2024). A continuación, voy a explicar el entorno que he utilizado para el análisis y las librerías que he utilizado en este proyecto:

2.2.1 Jupyter Notebook

Jupyter Notebook es un entorno informático interactivo basado en la web para crear documentos de Jupyter notebook. Jupyter Notebook permite crear documentos que contienen código, ecuaciones, visualizaciones y texto narrativo. Es una herramienta especialmente popular entre los analistas de datos ya que puedes ejecutar fragmentos de código de manera interactiva facilitando el proceso de encontrar posibles errores en el código. También soporta la visualización de gráficos, este punto es muy importante para analizar la información. Soporta múltiples lenguajes como R, Julia, Scala, aunque el más popular es Python. (Jupyter Notebook, 2024)

2.2.2 Statsbompy

Antes de hablar de la librería me gustaría empezar hablando de que es Statsbomb.

Statsbomb es una empresa creada por analistas de fútbol que se encarga de recopilar y analizar los datos deportivos. Empezó siendo un blog de análisis de fútbol en 2013 y pronto se convirtió en un lugar de referencia en el que encontrar contenido basado en datos que utiliza todo el mundo (Statsbomb, 2023).

Esta empresa tiene un apartado en el que se puede acceder a datos gratuitos. Cuando accedes a ese apartado te lleva a un repositorio de GitHub. (Github statsbomb, 2024) En este repositorio te explica cómo puedes acceder a sus datos de manera gratuita. En mi caso he utilizado Statsbomby, en el que los datos se proporcionan como archivos JSON exportados desde la API de datos de Statsbomb. En el mismo repositorio te explica como instalarlo:

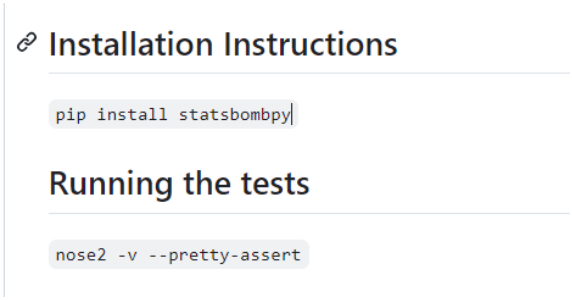
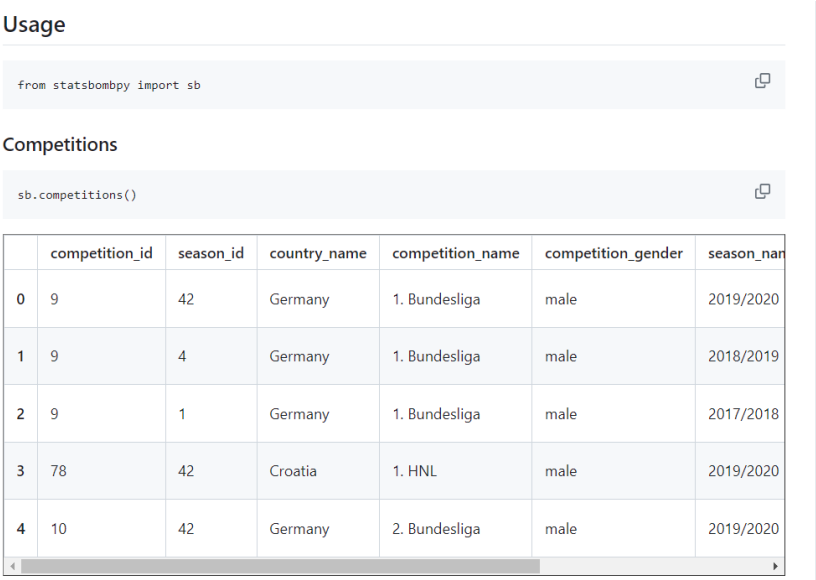


Figura 1. Instalación de librería Stastbomby

Y también explica como empezar a utilizarlo:



	competition_id	season_id	country_name	competition_name	competition_gender	season_nan
0	9	42	Germany	1. Bundesliga	male	2019/2020
1	9	4	Germany	1. Bundesliga	male	2018/2019
2	9	1	Germany	1. Bundesliga	male	2017/2018
3	78	42	Croatia	1. HNL	male	2019/2020
4	10	42	Germany	2. Bundesliga	male	2019/2020

Figura 2. Ejemplo de uso librería Stasbombpy

Getting Started

The data is provided as JSON files exported from the StatsBomb Data API, in the following structure:

- Competition and seasons stored in `competitions.json`.
- Matches for each competition and season, stored in `matches`. Each folder within is named for a competition ID, each file is named for a season ID within that competition.
- Events and lineups for each match, stored in `events` and `lineups` respectively. Each file is named for a match ID.
- StatsBomb 360 data for selected matches, stored in `three-sixty`. Each file is named for a match ID.

Some documentation about the meaning of different events and the format of the JSON can be found in the `doc` directory.

Figura 3. 2º Ejemplo de uso librería Stasbombpy

Figura 4. Logo de Stastbomb

2.2.3 Pandas

Pandas es una librería de Python especializada en la manipulación y el análisis de datos. Alguna de sus funcionalidades son:

- Realizar operaciones de limpieza, transformación, filtrado y análisis de datos.
- Permite cargar datos desde varios formatos (CSV, Excel, SQL, etc)
- Permite manejar datos nulos, fusión y concatenación de tablas.

Pandas implementa dos conceptos:

- **Series**: estructura de datos unidimensional. Cada elemento en una Serie tiene una etiqueta (índice) que permite acceder y manipular datos.
- **DataFrames**: estructura bidimensional, similar a una tabla o una hoja de cálculos. Consiste en una colección de Series que comparten el mismo índice, lo que facilita la manipulación de datos tabulares. (Pandas, 2024)

Figura 5. Logo de Pandas

2.2.4 Matplotlib

Matplotlib es una librería de visualización de datos en Python y permite crear gráficos y diagramas estáticos. Permite generar gráficos de líneas, gráficos de dispersión, gráficos de barras, histogramas, etc.

Esta librería es permite personalizar y dar estilo a los gráficos. Proporciona una amplia variedad de opciones para modificar colores, marcadores, estilos de línea, etiquetas, ejes, cuadriculas y leyendas, entre otros.

Matplotlib permite funcionar con otras librerías de Python, como Pandas, lo que facilita el análisis y manipulación de datos. También se integra perfectamente con Jupyter Notebook y otros entornos para la programación con Python.

Con Matplotlib también se puede guardar gráficos en diferentes formatos, como PNG, PDF, SVG, HTML o GIFs animados, esta funcionalidad la utilizaremos para exportar los gráficos que generemos, como los gráficos de pases o de tiros. (Matplotlib, 2024)

Figura 6. Logo de Matplotlib

2.2.5 NumPy

NumPy es una librería para la computación científica en Python. Proporciona soporte para grandes arreglos y matrices multidimensionales, junto a una colección de funciones matemáticas de alto nivel para operar con estos datos. Es muy utilizado en el análisis de datos y la ciencia de datos.

Sus principales funcionalidades son operaciones matemáticas y estadísticas, *arrays* multidimensionales, manipulación de datos, álgebra lineal, generación de datos, entre otras.

Su capacidad de manejar grandes volúmenes de datos y realizar operaciones complejas con facilidad la convierte en una librería indispensable para analistas de datos, científicos y desarrolladores. (Numpy, 2024)

Figura 7. Logo de NumPy

2.2.6 Mplsoccer

Mplsoccer es una librería de Python diseñada para la visualización de datos en el fútbol. Aprovecha las funcionalidades de Matplotlib, ya mencionada anteriormente, y agrega funcionalidades y herramientas para trabajar con datos de fútbol.

Esta librería facilita a los analistas la creación de gráficos informativos y atractivos relacionados con los eventos y la estadística del juego.

Sus principales funcionalidades son la visualización de pases y tiros, los mapas de calor y diagramas de posición, visualización de eventos y la más importante para mi proyecto es que tiene compatibilidad con la librería de Statsbombpy. (mplsoccer, 2021)

Figura 8. Logo de Mplsoccer

2.2.7 Os

Os es una librería de Python que proporciona una manera de usar funcionalidades dependientes del sistema operativo. Permite a los programas interactuar con el sistema operativo de manera eficiente, realizando tareas como la manipulación de archivos y directorios, la ejecución de comandos del sistema y la obtención de información sobre el entorno del sistema.

Sus principales funcionalidades son la manipulación de archivos y directorios, lectura y escritura, navegación, información del entorno, manipulación de rutas, entre otras. (os, 2023)

3. Desarrollo y resultados del trabajo.

A la hora de explicar el desarrollo y resultados del trabajo, he diferenciado 3 puntos importantes:

- Limpieza y extracción de datos con Python y Jupyter Notebook.
- Creación de la base de datos en el sistema gestión de base de datos relacional MySQL.
- Representación de datos con Power BI.

3.1 Limpieza y extracción de datos con Python y Jupyter Notebook

A continuación, voy a explicar cómo he hecho la limpieza y extracción de los datos mediante el código en Python. Voy a explicar que he hecho en cada código para extraer los datos que quería para realizar el análisis y poder visualizarlo posteriormente en Power BI.

3.1.1 Extraer los archivos .xlsx de la Alineación de España, de Portugal y los partidos de España en el Mundial del 2018

El primer paso es importar las librerías necesarias para poder realizar este análisis.

```
import pandas as pd
from statsbombpy import sb
import matplotlib.pyplot as plt
import numpy as np
from mplsoccer import VerticalPitch
from pandas import json_normalize
```

```
pd.set_option('display.max_rows', None)  # Mostrar todas las filas
pd.set_option('display.max_columns', None)  # Mostrar todas las columnas
```

Figura 9. Importación librerías

Obtenemos información sobre las competiciones disponibles en la librería de Statsbombpy. Este código nos devuelve un DataFrame que contiene los detalles sobre el ID de la competición, el año de la competición, etc. He subrayado de entre muchas de las competiciones, cual es la que voy a utilizar para el análisis.

```
sb.competitions()
```

25	37	4	England	FA Women's Super League	female	False
26	1470	274	International	FIFA U20 World Cup	male	False
27	43	106	International	FIFA World Cup	male	False
28	43	3	International	FIFA World Cup	male	False

Figura 10. Información de las competiciones

Visualizamos DataFrames de los partidos de la competición que quiero analizar.

```
df_partidos = sb.matches(competition_id=43, season_id=3)
df_partidos = pd.DataFrame(df_partidos)
df_partidos
```

	match_id	match_date	kick_off	competition	season	home_team	away_team	home_score	away_score
0	7585	2018-07-03	20:00:00.000	International - FIFA World Cup	2018	Colombia	England	1	1
1	7570	2018-06-28	20:00:00.000	International - FIFA World Cup	2018	England	Belgium	0	1
2	7586	2018-07-03	16:00:00.000	International - FIFA World Cup	2018	Sweden	Switzerland	1	0
3	7557	2018-06-25	20:00:00.000	International - FIFA World Cup	2018	Iran	Portugal	1	1
4	7542	2018-06-20	14:00:00.000	International - FIFA World Cup	2018	Portugal	Morocco	1	0

Figura 11. Información de los partidos

Filtro solo los partidos en los que España haya jugado, para ello filtramos los partidos en los que la columna *"home_team"* o *"away_team"* sean igual a *"Spain"*.

```
partidos_españa = df_partidos[(df_partidos["home_team"] == "Spain") | (df_partidos["away_team"] == "Spain")]
partidos_españa
```

	match_id	match_date	kick_off	competition	season	home_team	away_team	home_score	away_score	match_status	match_
17	7576	2018-06-15	20:00:00.000	International - FIFA World Cup	2018	Portugal	Spain	3	3	available	
42	7582	2018-07-01	16:00:00.000	International - FIFA World Cup	2018	Spain	Russia	1	1	available	
45	7560	2018-06-25	20:00:00.000	International - FIFA World Cup	2018	Spain	Morocco	2	2	available	
51	7543	2018-06-20	20:00:00.000	International - FIFA World Cup	2018	Iran	Spain	0	1	available	

Figura 12. Filtrado de partidos de España

Obtengo datos de la alineación del partido de Portugal vs España.

```
# Obtener los datos de la alineación del partido
lineups_espport = sb.lineups(match_id=7576)
lineups_espport
```

```
{'Portugal':     player_id                                player_name    pla
 0        2947                              Anthony Lopes             None
 1        3168      João Filipe Iria Santos Moutinho      João Moutinho
 2        3193  Bernardo Mota Veiga de Carvalho e Silva    Bernardo Silva
 3        3329           Cédric Ricardo Alves Soares      Cédric Soares
 4        3707      Adrien Sebastian Perruchet Silva       Adrien Silva
 5        3960            José Miguel da Rocha Fonte         José Fonte
```

Figura 13. Alineación del partido

Obtengo datos de la alineación de España.

```
lineups_esp = lineups_espport['Spain']
lineups_esp = pd.DataFrame(lineups_esp)
lineups_esp
```

	player_id	player_name	player_nickname	jersey_number	country
0	3064	David Josué Jiménez Silva	David Silva	21	Spain
1	3333	David de Gea Quintana	David de Gea	1	Spain
2	3498	Ignacio Monreal Eraso	Nacho Monreal	16	Spain

Figura 14. Alineación de España

Ahora limpio las columnas "*cards*" y "*positions*". Estas columnas contienen listas de diccionarios. La función lambda toma un argumento "x", que se espera sea una lista de diccionarios, y utiliza una lista de comprensión y el método .**get()** para recorrer cada diccionario en "cards" y extraer el valor correspondiente a la clave "card_type", y en "positions" para extraer el valor de la clave "position". Así, obtengo para cada jugador las tarjetas que ha recibido en el partido y las posiciones en las que puede jugar.

```
# Para las columnas 'cards' y 'positions', extraemos solo la clave que necesitamos
lineups_esp['cards'] = lineups_esp['cards'].apply(lambda x: [item.get('card_type') for item in x])
lineups_esp['positions'] = lineups_esp['positions'].apply(lambda x: [item.get('position') for item in x])

# Mostrar el DataFrame resultante
lineups_esp
```

	player_id	player_name	player_nickname	jersey_number	country	cards	positions
0	3064	David Josué Jiménez Silva	David Silva	21	Spain	[]	[Right Wing]
1	3333	David de Gea Quintana	David de Gea	1	Spain	[]	[Goalkeeper]
2	3498	Ignacio Monreal Eraso	Nacho Monreal	16	Spain	[]	[]
3	3957	César Azpilicueta Tanco	César Azpilicueta	14	Spain	[]	[]
4	4926	Francisco Román Alarcón Suárez	Isco	22	Spain	[]	[Left Wing]
5	5198	Diego da Silva Costa	Diego Costa	19	Spain	[]	[Center Forward]

Figura 15. Limpiar datos de columna "cards" y "positions España"

Hago el mismo proceso para la alineación de Portugal.

```
lineups_port = lineups_espport['Portugal']
lineups_port = pd.DataFrame(lineups_port)
```

```
# Para las columnas 'cards' y 'positions', extraemos solo la clave que necesitamos
lineups_port['cards'] = lineups_port['cards'].apply(lambda x: [item.get('card_type') for item in x])
lineups_port['positions'] = lineups_port['positions'].apply(lambda x: [item.get('position') for item in x])

# Mostrar el DataFrame resultante
lineups_port
```

	player_id	player_name	player_nickname	jersey_number	country	cards	positions
0	2947	Anthony Lopes	None	12	Portugal	[]	[]
1	3168	João Filipe Iria Santos Moutinho	João Moutinho	8	Portugal	[]	[Right Center Midfield]
2	3193	Bernardo Mota Veiga de Carvalho e Silva	Bernardo Silva	11	Portugal	[]	[Right Wing]

Figura 16. Limpiar datos de columna "cards" y "positions Portugal"

Por último, exportamos los DataFrames sin índice en formato .xlsx y como no le he indicado ruta se guarda en la misma ruta donde está el código de Jupyter Notebook.

```
lineups_port.to_excel('Alineacion-Portugal.xlsx', index=False)
lineups_esp.to_excel('Alineacion-España.xlsx', index=False)
partidos_españa.to_excel('Partidos-España.xlsx', index=False)
```

Figura 17. Exportar DataFrames

3.1.2 Extracción de las estadísticas principales del partido

Importo las librerías necesarias para la manipulación de los datos y obtengo los eventos del partido.

```
import pandas as pd
from statsbombpy import sb
import matplotlib.pyplot as plt
import numpy as np
from mplsoccer import VerticalPitch
from pandas import json_normalize
```

```
# Obtener datos de eventos del partido
match_id = 7576
df_eventos = sb.events(match_id=match_id)
print(df_eventos.head())
print(df_eventos.info())
```

Figura 18. Importar librería y obtener datos de los eventos del partido

Filtro los eventos del partido por cada equipo (Portugal y España) y calculo las estadísticas principales del partido.

Primero, obtengo los partidos únicos que han participado en el partido y creo una lista para guardar la información. Luego, con un bucle **for** obtengo para cada equipo los eventos para calcular el porcentaje de posesión del balón, calculo el número de goles, tiros totales, pases realizados, faltas, fueras de juego y saques de esquina.

Estas estadísticas se guardan en un diccionario y con la librería pandas la convierto en un DataFrame.

```
# Filtrar eventos por equipo y calcular estadísticas
equipos = df_eventos['possession_team'].unique()
datos_generales = []

for equipo in equipos:
    df_equipo = df_eventos[df_eventos['possession_team'] == equipo]
    goles = len(df_equipo[(df_equipo['type'] == 'Shot') & (df_equipo['shot_outcome'] == 'Goal')])
    tiros = len(df_equipo[df_equipo['type'] == 'Shot'])
    tiros_puerta = len(df_equipo[(df_equipo['type'] == 'Shot') &
                                 (df_equipo['shot_outcome'] != 'Goal') &
                                 (df_equipo['shot_outcome'] != 'Off T')]) # Filtrar los tiros que no van fuera
    pases = len(df_equipo[df_equipo['type'] == 'Pass'])
    total_pases = len(df_eventos[df_eventos['type'] == 'Pass'])
    posesion = (pases / total_pases) * 100
    faltas = len(df_equipo[df_equipo['type'] == 'Foul Committed'])
    fueras_juego = len(df_equipo[df_equipo['type'] == 'Offside'])
    corners = len(df_equipo[df_equipo['pass_type'] == 'Corner'])

    datos_generales.append({
        'Equipo': equipo,
        'Goles': goles,
        'Tiros': tiros,
        'Tiros a puerta': tiros_puerta,
        'Posesión': posesion,
        'Pases': pases,
        'Faltas': faltas,
        'Fueras de juego': fueras_juego,
        'Saques de esquina': corners
    })
# Crear DataFrame
df_partido_stats = pd.DataFrame(datos_generales)
# Mostrar el DataFrame
df_general = df_partido_stats
df_general
```

Figura 19. DataFrame estadísticas principales del partido

	Equipo	Goles	Tiros	Tiros a puerta	Posesión	Pases	Faltas	Fueras de juego	Saques de esquina
0	Portugal	3	9	4	32.962329	385	13	0	3
1	Spain	3	13	5	67.037671	783	11	1	5

Figura 20. Resultado DataFrame estadísticas principales del partido

Sumo las tarjetas amarillas y tarjetas rojas para cada equipo iterando sobre las filas y verificando si algún jugador ha obtenido alguna tarjeta amarilla durante el partido y lo guardo en un DataFrame para las tarjetas amarillas y otro para las rojas para posteriormente añadirlo al DataFrame de estadísticas generales que ya había creado anteriormente.

```
# Obtener alineaciones del partido para ambos equipos
lineups = sb.lineups(match_id=7576)
lineups_portugal = pd.DataFrame(lineups['Portugal'])
lineups_spain = pd.DataFrame(lineups['Spain'])

# Contar el número de tarjetas amarillas para Portugal
num_tarjetas_amarillas_portugal = sum(len(row['cards']) for _,
                                      row in lineups_portugal.iterrows() if any(card['card_type'] == 'Yellow Card'
                                                                                for card in row['cards']))
# Contar el número de tarjetas amarillas para España
num_tarjetas_amarillas_spain = sum(len(row['cards']) for _,
                                   row in lineups_spain.iterrows() if any(card['card_type'] == 'Yellow Card'
                                                                          for card in row['cards']))
# Crear un DataFrame con el número de tarjetas amarillas para ambos equipos
df_tarjetas_amarillas = pd.DataFrame({'Equipo': ['Portugal', 'Spain'],
                                      'Tarjetas Amarillas': [num_tarjetas_amarillas_portugal, num_tarjetas_amarillas_spain]})
```

Figura 21. Tarjetas Amarillas

```
# Contar el número de tarjetas rojas para Portugal
num_tarjetas_rojas_portugal = sum(len(row['cards']) for _,
                                  row in lineups_portugal.iterrows() if any(card['card_type'] == 'Red Card'
                                                                            for card in row['cards']))

# Contar el número de tarjetas rojas para España
num_tarjetas_rojas_spain = sum(len(row['cards']) for _,
                               row in lineups_spain.iterrows() if any(card['card_type'] == 'Red Card'
                                                                      for card in row['cards']))

# Crear un DataFrame con el número de tarjetas rojas para ambos equipos
df_tarjetas_rojas= pd.DataFrame({'Equipo': ['Portugal', 'Spain'],
                                 'Tarjetas Rojas': [num_tarjetas_rojas_portugal, num_tarjetas_rojas_spain]})
```

Figura 22. Tarjetas Rojas

Ahora fusiono los DataFrames de tarjetas amarillas (df_tarjetas_amarillas) y de tarjetas rojas (df_tarjetas_rojas) con el DataFrame de estadísticas generales (df_general) creado anteriormente.

```
df_general = df_general.merge(df_tarjetasAmarilla)
df_general = df_general.merge(df_tarjetas_rojas)
df_general
```

Figura 23. Fusionar DataFrame general con DataFrame de las tarjetas

	Equipo	Goles	Tiros	Tiros a puerta	Posesión	Pases	Faltas	Fueras de juego	Saques de esquina	Tarjetas Amarillas	Tarjetas Rojas
0	Portugal	3	9	4	32.962329	385	13	0	3	1	0
1	Spain	3	13	5	67.037671	783	11	1	5	1	0

Figura 24. Resultado Dataframe general final

Por último, exporto el DataFrame a un archivo .xlsx sin el índice.

```
df_general.to_excel('Estadisticas_Partido_Esp_Port.xlsx', index=False)
```

Figura 25. Exportar a formato .xlsx el DataFrame general

3.1.3 Mapa de tiro

Importo las librerías necesarias para visualizar los eventos sobre los tiros en el partido de Portugal vs España.

También obtengo los eventos del partido utilizando la función **events** del módulo de Statsbombpy y lo guardo en la variable "partido".

```
from statsbombpy import sb
import pandas as pd
import matplotlib.pyplot as plt
from mplsoccer import VerticalPitch, Pitch
import numpy as np
import os
```

```
pd.set_option('display.max_rows', None)  # Mostrar todas las filas
pd.set_option('display.max_columns', None)  # Mostrar todas las columnas
```

```
partido = sb.events(match_id=7576)
```

```
partido.columns
```

Figura 26. Librerías y eventos de Mapa de tiro

Creo DataFrame llamado "tiros" y luego filtro de todos los eventos solo los de tipo "Shot". Estos datos se agregan al DataFrame "tiros" para posteriormente extraer las coordenadas x e y de la columna "location". Como se muestra en la imagen hay 22 filas que son el total de tiros en el partido.

Finalmente muestro un DataFrame con las coordenadas "x", "y" y "location" de cada tiro en el partido.

```
tiros = pd.DataFrame()
partido = partido[partido['type'] == "Shot"]
tiros = tiros.append(partido)
tiros.shape
```

```
(22, 81)
```

```
tiros[['x','y']] = tiros.location.apply(pd.Series)
tiros[['x','y','location']]
```

	x	y	location
4018	109.0	41.0	[109.0, 41.0]
4019	106.0	38.0	[106.0, 38.0]
4020	96.0	53.0	[96.0, 53.0]
4021	99.0	63.0	[99.0, 63.0]

Figura 27. Coordenada "x" "y" de cada tiro del partido

Agrego una nueva columna llamada “shot_team” al DataFrame ya creado “tiros”. La columna es España cuando la posesión la tiene España y Portugal cuando la tiene Portugal. Luego cuento cuantos tiros realizó cada equipo en el partido y se guardan en dos DataFrames.

```
tiros['shot_team'] = np.where(tiros['possession_team'] == 'Spain','España','Portugal')
tiros.shot_team.value_counts()
```

```
España      13
Portugal     9
Name: shot_team, dtype: int64
```

```
tiros_Esp = tiros[tiros['shot_team'] == 'España']
tiros_Port = tiros[tiros['shot_team'] == 'Portugal']
```

Figura 28. Tiros España y Portugal en el partido

Creo un gráfico de dispersión en un campo de fútbol utilizando la librería **mplsoccer**. Configuro el tamaño y defino el campo para que se muestre en vertical. Agrego puntos de dispersión para representar los tiros de España. El tamaño depende de la probabilidad esperada de gol en cada tiro.

```
fig, ax = plt.subplots(figsize=(16,9))
pitch = VerticalPitch(
    pitch_type='statsbomb',
    goal_type='box',
    half=True
)

pitch.draw(ax=ax)
plt.ylim(70,125)

pitch.scatter(tiros_Esp.x, tiros_Esp.y,ax=ax, s= tiros_Esp.shot_statsbomb_xg*700, ec = 'black', label = 'Tiros')
plt.legend()
plt.title('Tiros España')
```

Figura 29. Código Primera versión gráfico de tiros de España

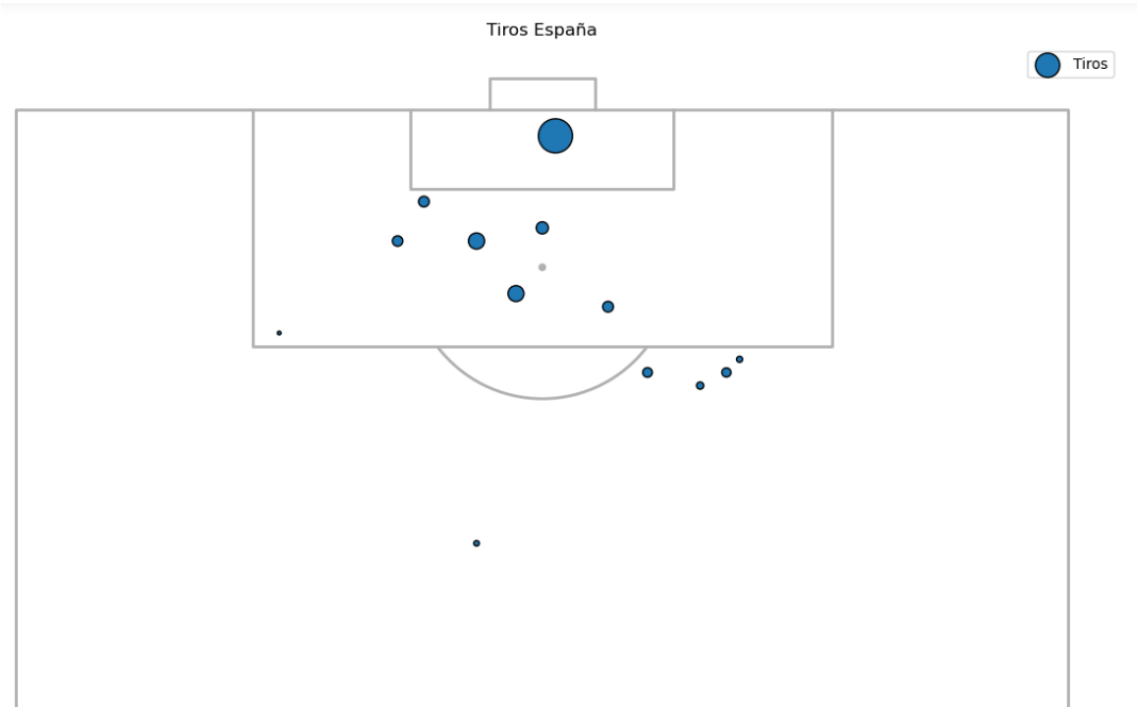


Figura 30. Primera versión gráfico de tiros de España

Ahora diferencio los tiros que son goles de los que no lo son.

```
goles_Esp = tiros_Esp[tiros_Esp.shot_outcome == 'Goal']
goles_Port = tiros_Port[tiros_Port.shot_outcome == 'Goal']
No_goles_Esp = tiros_Esp[tiros_Esp.shot_outcome != 'Goal']
No_goles_Port = tiros_Port[tiros_Port.shot_outcome != 'Goal']
```

Figura 31. Goles y No goles

Mejoro el grafico creado anteriormente. Pongo el campo en negro y también diferencio los tiros que son goles, de los tiros que no lo son. Los que son los pinta en verde y los que no lo son en rojo. Además, añado a la derecha del gráfico el equipo que ataca, los tiros y los goles. Y guardo el archivo en una carpeta creada con la librería **os**. Hago el mismo proceso para el grafico de tiros de Portugal.

```
fig, ax = plt.subplots(figsize=(16,9))
pitch = VerticalPitch(
    pitch_type='statsbomb',
    pitch_color='black',  # Cambiar color del campo a negro
    line_color='white',   # Cambiar color de las líneas a blanco
    goal_type='box',
    half=True
)

pitch.draw(ax=ax)
#plt.ylim(70,125)
plt.xlim(-5,100)

pitch.scatter(goles_Esp.x, goles_Esp.y,ax=ax, s= goles_Esp.shot_statsbomb_xg*1000, alpha= .9,
              ec = 'white', label = 'Goles', color= 'green')
pitch.scatter(No_goles_Esp.x, No_goles_Esp.y,ax=ax, s= No_goles_Esp.shot_statsbomb_xg*1000,alpha= .8,
              ec = 'white', label = 'Tiros', color = 'red')
plt.legend(ncol= 2, loc = "upper left", prop = {'size': 16},shadow = True, edgecolor = 'black')
ax.text(90,105, "Ataque\nEspaña",va = "center", ha= "center", fontsize = 25, color = 'white')
ax.text(90,95,f'{tiros_Esp.shape[0]} tiros',va = "center", ha= "center", fontsize = 20, color = 'white')
ax.text(90,90,f'{goles_Esp.shape[0]} goles',va = "center", ha= "center", fontsize = 20, color = 'white')

carpeta_salida_esp = 'equipo/graficos_tiros/'
if not os.path.exists(carpeta_salida_esp):
    os.makedirs(carpeta_salida_esp)

nombre_archivo_esp = f'{carpeta_salida_esp} tiros España.png'
plt.savefig(nombre_archivo_esp, dpi=300)  # dpi ajusta la resolución del archivo
```

Figura 32. Código definitivo grafico de tiros España

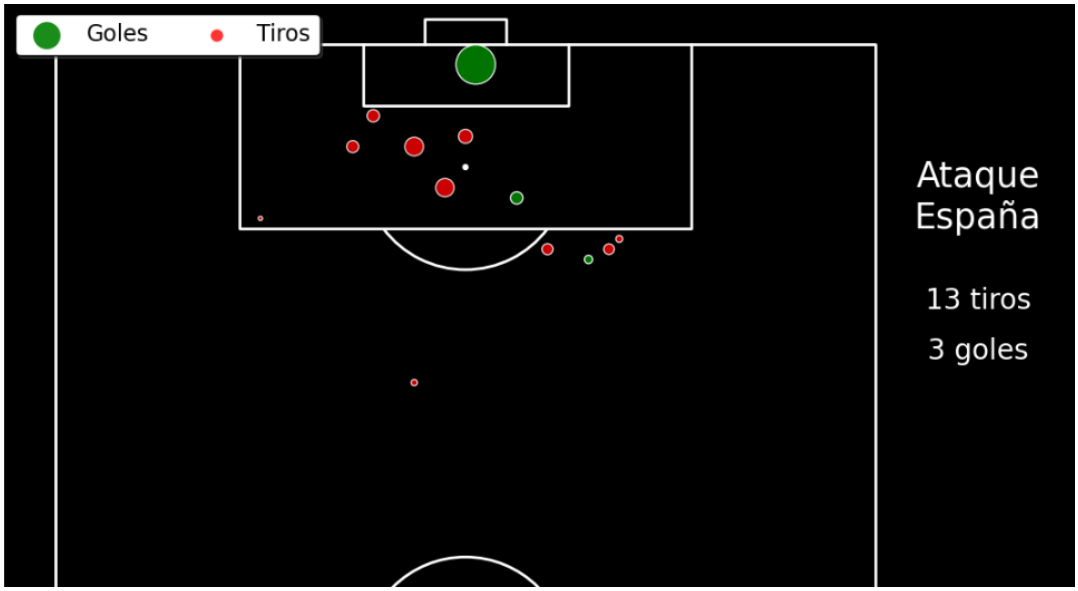


Figura 33. Gráfico definitivo de tiros España

3.1.4 Mapa de calor

Importo las librerías necesarias para hacer los mapas de calor de cada jugador en el partido Portugal vs España.

También establezco una carpeta de salida donde se guardarán los archivos PNG generados de cada jugador.

```
import pandas as pd
from mplsoccer import VerticalPitch, Pitch
from statsbombpy import sb
import matplotlib.pyplot as plt
import os
import numpy as np
import warnings
warnings.filterwarnings('ignore')

# Carpeta donde se guardarán los archivos PNG
carpeta_salida = 'jugador/bueno/graficos_calor/'

# Verificar si la carpeta de salida existe, si no, crearla
if not os.path.exists(carpeta_salida):
    os.makedirs(carpeta_salida)
```

Figura 34. Importación librerías Mapa de Calor

Cargo los datos de los eventos y alineaciones del partido, separándolo entre España y Portugal.

```
partido = sb.events(match_id=7576)
lineups_espport = sb.lineups(match_id=7576)
lineups_esp = lineups_espport['Spain']
lineups_por = lineups_espport['Portugal']
```

Figura 35. Alineaciones y eventos Mapa de Calor

Selecciono las columnas “player_id” y “player_nickname” de las alineaciones de España y Portugal y las concateno en un DataFrame (df_combined).

Luego, creo una lista de apodos (jugadores_con_apodos) que son los nombres que conoce la mayoría, filtrando sus apodos con sus identificadores únicos.

Finalmente, la lista creada anteriormente la convierto en un array para mas tarde utilizar esos apodos como los títulos de los gráficos de cada jugador.

```
# Seleccionar solo las columnas 'player_id' y 'player_nickname' de cada DataFrame
df_esp_subset = lineups_esp[['player_id', 'player_nickname']]
df_por_subset = lineups_por[['player_id', 'player_nickname']]

partido = sb.events(match_id=7576)
# Obtener solo los player_id únicos del DataFrame partido
player_ids = partido['player_id'].unique()


# Concatenar los DataFrames
df_combined = pd.concat([df_esp_subset, df_por_subset], ignore_index=True)

# Inicializar una lista para almacenar los apodos de los jugadores
jugadores_con_apodos = [df_combined.loc[df_combined['player_id'] == player_id, 'player_nickname'].iloc[0]
                        for player_id in player_ids if player_id in df_combined['player_id'].values]

# Convertir la lista de jugadores con apodos en un array de NumPy
jugadores_con_apodos = np.array(jugadores_con_apodos)
```

Figura 36. Apodos de los jugadores

Primero obtenemos la lista de jugadores únicos que han participado en el partido y filtramos aquellos que el jugador es NaN.

Luego itero por cada jugador los eventos del partido, contando con las coordenadas (“x” e “y”). Con esos eventos para cada jugadore se genera un gráfico de un campo de futbol utilizando la librería “**mplsoccer**” y se genera un mapa de calor a través de los pases realizados en el partido. Cada grafico tiene el titulo de “Mapa de calor de (apodo del jugador) en el partido” y se guarda como PNG en la carpeta creada anteriormente.

```
# Obtener la lista de jugadores únicos en el partido, excluyendo "Nan"
jugadores = partido.loc[partido['player'].notna(), 'player'].unique()
for jugador_id, apodo in zip(jugadores, jugadores_con_apodos):
    df_jugador = partido[partido['player'] == jugador_id]
    df_jugador[['x','y']] = df_jugador.location.apply(pd.Series)
    #df_jugador[['end_x','end_y']] = df_jugador.pass_end_location.apply(pd.Series)
    df_jugador[['x','y','location']]
    fig, ax = plt.subplots(figsize=(16,9))
    pitch = VerticalPitch(pitch_type='statsbomb', line_color='black', pitch_color='white')
    pitch.draw(ax=ax)
    pitch.kdeplot(
        df_jugador[df_jugador['type'] == 'Pass']['x'],
        df_jugador[df_jugador['type'] == 'Pass']['y'],
        ax=ax,
        levels=100,
        shade=True,
        zorder=-1,
        shade_lowest=True,
        cmap='OrRd'
    )
    ax.set_title(f'Mapa de calor de {apodo} en el partido')
    # Guardar el gráfico como un archivo PNG en la carpeta de salida
    nombre_archivo = f'{carpeta_salida}mapa_calor_{apodo}.png'
    plt.savefig(nombre_archivo, dpi=300)  # dpi ajusta la resolución del archivo
```

Figura 37. Código creación del Mapa de Calor

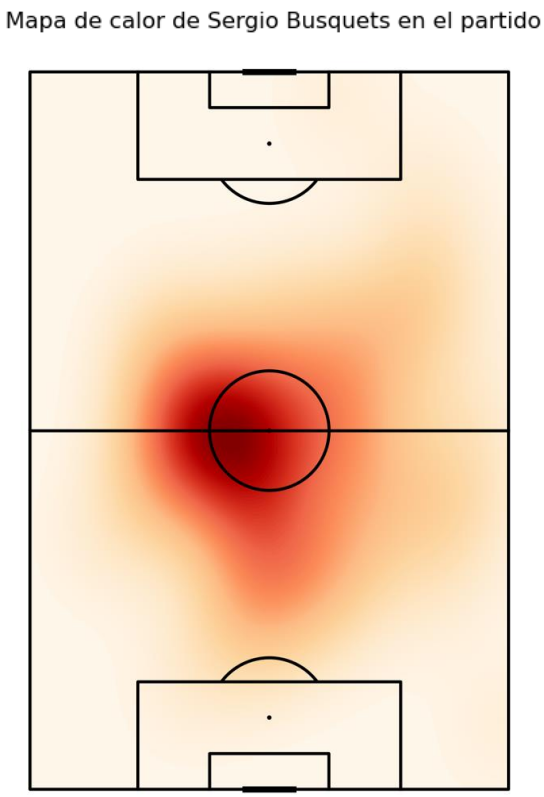


Figura 38. Ejemplo Mapa de Calor de un jugador

3.1.5 Mapa de pases

Importamos las librerías necesarias y mostramos todas las filas y columnas a la hora de visualizar los DataFrames si fuera necesario.

```
import pandas as pd
from statsbombpy import sb
import matplotlib.pyplot as plt
from mplsoccer import VerticalPitch, Pitch
import os
import numpy as np

# Establecer opciones de visualización de pandas
pd.set_option('display.max_rows', None)  # Mostrar todas las filas
pd.set_option('display.max_columns', None)  # Mostrar todas las columnas
```

Figura 39. Importación de librerías Mapa de pases

Defino donde se guardarán los archivos PNG con ayuda de la librería **os**. Verifica si existe la carpeta y si no, la crea en la ruta indicada.

```
# Carpeta donde se guardarán los archivos PNG
carpeta_salida = 'jugador/bueno/graficos_pases/'

# Verificar si la carpeta de salida existe, si no, crearla
if not os.path.exists(carpeta_salida):
    os.makedirs(carpeta_salida)
```

Figura 40. Carpeta de salida Mapa de pases

Obtengo las alineaciones y los eventos del partido entre Portugal y España con las columnas “player_id” y “player_nickname”, que es donde se guarda el apodo que voy a utilizar para poner título a los gráficos de cada jugador del mapa de pases y guardo los eventos del partido en la variable “partido”.

```
lineups_espport = sb.lineups(match_id=7576)
lineups_esp = lineups_espport['Spain']
lineups_por = lineups_espport['Portugal']

# Seleccionar solo las columnas 'player_id' y 'player_nickname' de cada DataFrame
df_esp_subset = lineups_esp[['player_id', 'player_nickname']]
df_por_subset = lineups_por[['player_id', 'player_nickname']]

partido = sb.events(match_id=7576)
```

Figura 41. Apodos y eventos Mapa de pases

Hago que los identificadores únicos de cada jugador (“player_id”) concatene el DataFrame de alineaciones y asigno los apodos (“player_nickname”) en una lista que posteriormente se convierte en un array para que puedan ser procesados y recorridos para hacer el mapa de pases por jugadore y así poner cada titulo en cada mapa de pases de cada jugador.

```
# Obtener solo los player_id únicos del DataFrame partido
player_ids = partido['player_id'].unique()


# Concatenar los DataFrames
df_combined = pd.concat([df_esp_subset, df_por_subset], ignore_index=True)

# Inicializar una lista para almacenar los apodos de los jugadores
jugadores_con_apodos = [df_combined.loc[df_combined['player_id'] == player_id, 'player_nickname'].iloc[0]
                        for player_id in player_ids if player_id in df_combined['player_id'].values]

# Convertir la lista de jugadores con apodos en un array de NumPy
jugadores_con_apodos = np.array(jugadores_con_apodos)
```

Figura 42. Apodos procesados Mapa de pases

Finalmente hago un bucle para crear un gráfico de mapa de pases por cada jugador. Para ello diferencio los pases completos de los incompletos. Cuento los pases y genero un gráfico de un campo de fútbol completo en vertical en el que los pases completos los pinto de verde y los pases incompletos se pintan en rojo. También se genera una leyenda en la esquina derecha del gráfico y a la derecha del gráfico indica cuantos pases completos e incompletos ha realizado.

Finalmente guardo cada gráfico como un archivo PNG en la carpeta especificada anteriormente.

```
# Obtener la lista de jugadores únicos en el partido, excluyendo "Nan"
jugadores = partido.loc[partido['player'].notna(), 'player'].unique()
# Crear un gráfico separado para cada jugador
for jugador_id, apodo in zip(jugadores, jugadores_con_apodos):
    # Filtrar los eventos para el jugador específico
    eventos_jugador = partido[partido['player'] == jugador_id]

    # Filtrar eventos de pases para el jugador
    pases_jugador = eventos_jugador[eventos_jugador['type'] == 'Pass']

    # Filtrar pases completos e incompletos
    pases_comp = pases_jugador[~pases_jugador['pass_outcome'].isin(['Incomplete', 'Pass Offside'])]
    pases_incomp = pases_jugador[pases_jugador['pass_outcome'].isin(['Incomplete', 'Pass Offside'])]

    # Conteo de pases completos e incompletos
    conteo_completos = len(pases_comp)
    conteo_incompletos = len(pases_incomp)

    # Crear la figura y el campo de fútbol
    fig, ax = plt.subplots(figsize=(16, 9))
    pitch = VerticalPitch(
        pitch_color='black',  # Cambiar color del campo a negro
        line_color='white',   # Cambiar color de las líneas a blanco
        pitch_type='statsbomb',
        goal_type='box',
    )
    pitch.draw(ax=ax)
```

Figura 43. Generación Mapa de pases (1)

```
# Dibujar flechas para pases completos e incompletos solo si hay pases disponibles
if not pases_comp.empty:
    pitch.arrows(pases_comp['location'].apply(lambda x: x[0]), pases_comp['location'].apply(lambda x: x[1]),
                 pases_comp['pass_end_location'].apply(lambda x: x[0]),
                 pases_comp['pass_end_location'].apply(lambda x: x[1]),
                 ax=ax, color='green', alpha=.8, width=1.5, label='Pases completos')
if not pases_incomp.empty:
    pitch.arrows(pases_incomp['location'].apply(lambda x: x[0]), pases_incomp['location'].apply(lambda x: x[1]),
                 pases_incomp['pass_end_location'].apply(lambda x: x[0]),
                 pases_incomp['pass_end_location'].apply(lambda x: x[1]),
                 ax=ax, color='red', width=1.5, label='Pases incompletos')

# Agregar el texto con el conteo de pases completos e incompletos
ax.text(105, 5, f'Pases completos: {conteo_completos}\nPases incompletos: {conteo_incompletos}', size=12, color='black')

ax.set_title(f'Mapa de pases de {apodo} en el partido')  # Cambiar el título con el apodo

# Mostrar la leyenda abajo a la derecha del gráfico
ax.legend(loc='lower right')

# Guardar el gráfico como un archivo PNG en la carpeta de salida
nombre_archivo = f'{carpeta_salida}mapa_pases_{apodo}.png'
plt.savefig(nombre_archivo, dpi=300)  # dpi ajusta la resolución del archivo

# Mostrar el gráfico
plt.show()
```

Figura 44. Generación Mapa de pases (2)

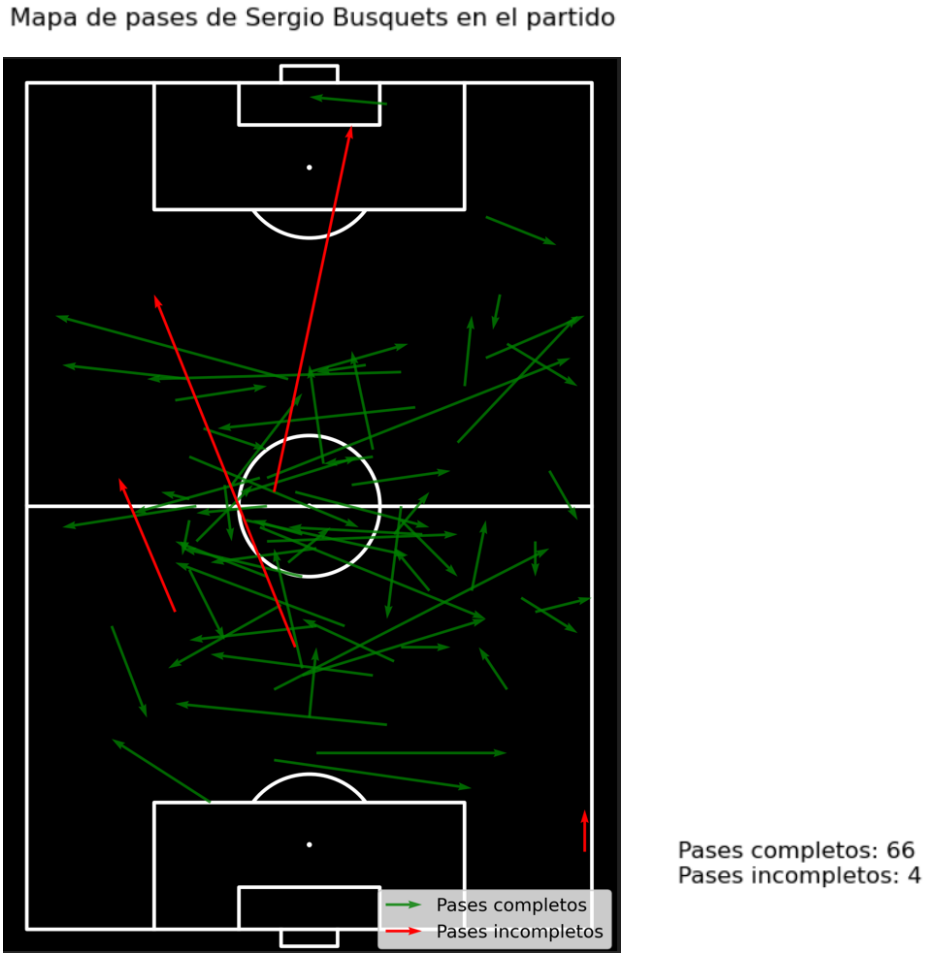


Figura 45. Ejemplo Mapa de pases

3.2 Creación de la base de datos

Creo una base de datos para almacenar informaciones generales de los jugadores en la temporada 2017 / 2018. Decido utilizar MySQL como sistema de gestión de base de datos por varias razones:

- **Escalabilidad y flexibilidad**: MySQL es escalable y puede manejar tanto pequeña cantidad de datos como volúmenes de datos grandes, así que si quiero mejorar el proyecto analizando más partidos y más jugadores no tendría ningún problema con MySQL.
- **Rendimiento y velocidad**: Las consultas y la creación de tablas o actualización de los datos son de un rendimiento alto.
- **Facilidad de uso y mantenimiento**: Con un poco de conocimiento en el lenguaje SQL es posible hacer consultas sencillas o crear y actualizar tablas, lo que me ha facilitado el trabajo.
- **Integración y compatibilidad**: Es compatible con otras herramientas, como Power BI. Desde Power BI se puede acceder a los datos relativamente fácil.
- **Coste**: MySQL es de código abierto y gratuito así que es la opción perfecta para proyectos de personales, aunque también ofrece versiones comerciales.

He creado tres tablas divididas entre delanteros, centrocampistas y defensas, ya que decidí que en cada posición quería medir una serie de métricas, porque para los delanteros quería obtener métricas ofensivas, para los centrocampistas métricas de pases y duelos y para los defensas métricas defensivas.

La información para crear cada tabla la he obtenido de Sofascore, que es una página web y una aplicación móvil que ofrece estadísticas detalladas de equipos y jugadores entre otras cosas (Wikipedia-SofasCore, 2023).

3.2.1 Creación de la base de datos

Creo la base de datos llamada “Estadísticas-SofaScore” con la siguiente configuración.

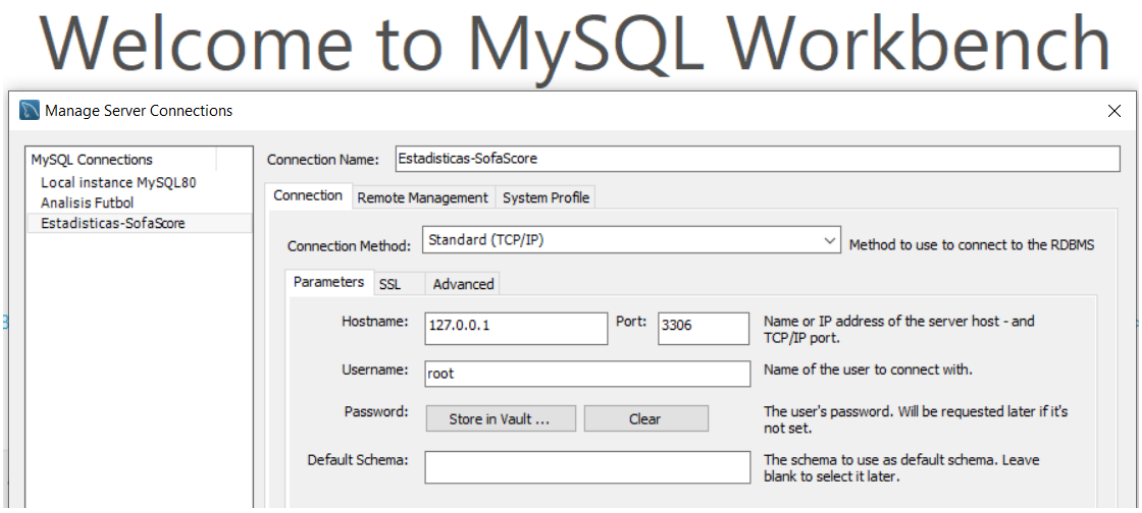


Figura 46. Configuración de la base de datos

3.2.2 Creación de las tablas

Creo las tablas con sus campos y el tipo de datos que va a utilizar ese campo.

Relleno el campo ID con el ID que he obtenido del análisis en Python.

- **Tabla atacantes**:

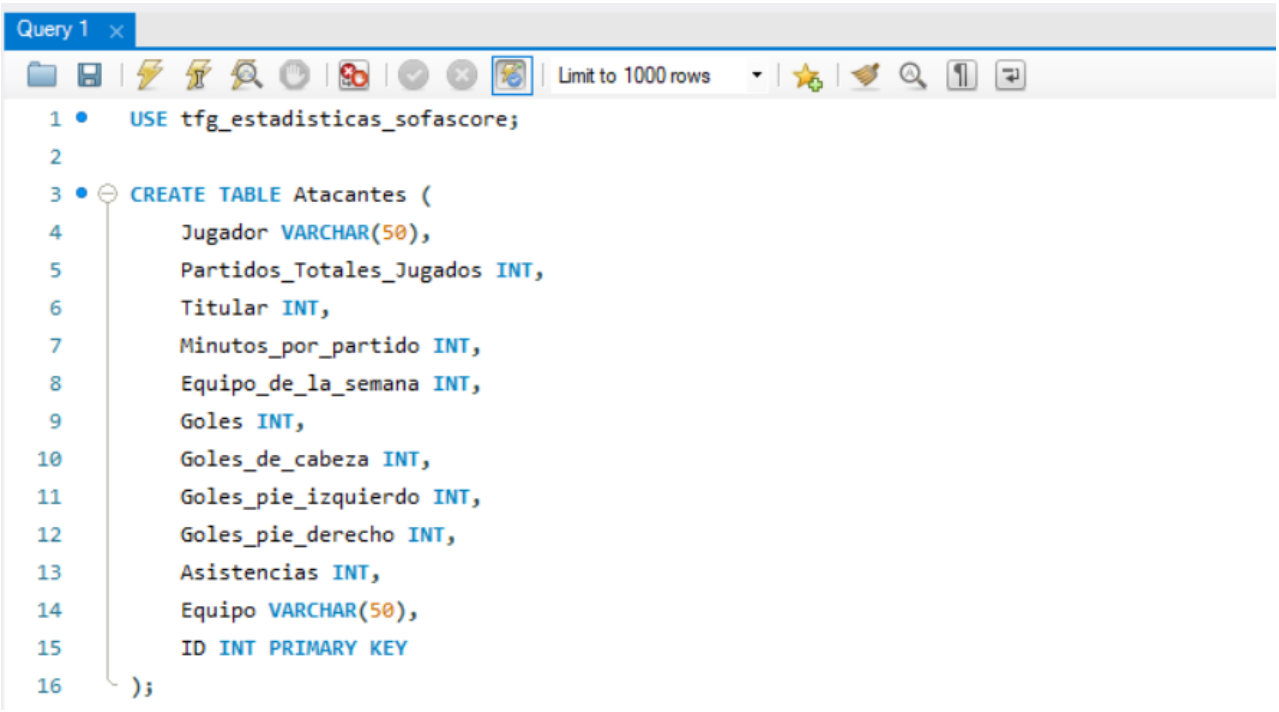


Figura 47. Creación tabla atacantes

```
INSERT INTO Atacantes (Jugador, Partidos_Totales_Jugados, Titular, Minutos_por_partido,
 Equipo_de_la_semana, Goles, Goles_de_cabeza, Goles_pie_izquierdo,
 Goles_pie_derecho, Asistencias, Equipo, ID)
VALUES
('Cristiano Ronaldo', 27, 27, 85, 11, 26, 5, 7, 14, 5, 'Real Madrid CF', 5207),
('Andre Silva', 24, 7, 38, 0, 2, 1, 0, 1, 2, 'A.C Milán', 5218),
('Gonçalo Guedes', 33, 27, 74, 4, 5, 0, 0, 5, 9, 'Valencia C.F', 4367),
('Diego Costa', 15, 13, 70, 0, 3, 0, 0, 3, 3, 'Atlético de Madrid', 5198),
('Iago Aspas', 34, 33, 86, 9, 22, 1, 13, 6, 5, 'Celta de Vigo', 5217);
```

Figura 48. Rellenar datos en la tabla atacantes de la información de SofasCore

```sql
CREATE TABLE Mediocentros (
    Jugador VARCHAR(50),
    Partidos_Totales_Jugados INT,
    Titular INT,
    Minutos_por_partido INT,
    Equipo_de_la_semana INT,
    Goles INT,
    Asistencias INT,
    Grandes_ocasiones_creadas INT,
    Pases_completados_por_partido DECIMAL(4,1),
    Duelos_totales_ganados DECIMAL(4,1),
    Equipo VARCHAR(50),
    ID INT PRIMARY KEY
);
```

Figura 49. Creación tabla mediocentros

```sql
INSERT INTO Mediocentros (Jugador, Partidos_Totales_Jugados, Titular, Minutos_por_partido,
 Equipo_de_la_semana, Goles, Asistencias, Grandes_ocasiones_creadas,
 Pases_completados_por_partido, Duelos_totales_ganados, Equipo, ID)
VALUES
('João Moutinho', 33, 31, 85, 4, 1, 6, 10, 49.5, 4.2, 'AS Mónaco', 3168),
('Bernardo Silva', 35, 15, 43, 2, 6, 4, 3, 23.2, 2.5, 'Manchester City', 3193),
('William Carvalho', 24, 24, 89, 3, 1, 1, 4, 56.8, 6.8, 'Sporting Club de Portugal', 5214),
('Bruno Fernandes', 33, 32, 85, 9, 11, 8, 11, 35.0, 5.9, 'Sporting Club de Portugal', 5204),
('Ricardo Quaresma', 26, 23, 77, 10, 4, 7, 18, 21.4, 5.1, 'Beşiktaş JK', 5215),
('João Mário', 13, 12, 79, 0, 2, 1, 1, 27.8, 4.5, 'West Ham United', 4272),
('Andrés Iniesta', 30, 25, 61, 2, 1, 2, 3, 44.7, 4.4, 'F.C. Barcelona', 5216),
('Koke', 35, 33, 79, 3, 4, 4, 10, 48.4, 4.5, 'Atlético de Madrid', 5199),
('Isco', 30, 21, 59, 3, 7, 7, 10, 45.0, 4.7, 'Real Madrid C.F.', 4926),
('Sergio Busquets', 31, 30, 84, 1, 1, 4, 2, 65.5, 6.9, 'F.C. Barcelona', 5203),
('David Silva', 29, 28, 84, 9, 9, 11, 14, 74.5, 4.1, 'Manchester City', 3064),
('Thiago Alcantara', 19, 12, 65, 4, 2, 2, 2, 62.0, 5.5, 'Bayern de Múnich', 5208);
```

Figura 50. Rellenar datos en la tabla mediocentros de la información de SofasCore

```sql
CREATE TABLE Defensas (
    Jugador VARCHAR(50),
    Partidos_Totales_Jugados INT,
    Titular INT,
    Minutos_por_partido INT,
    Equipo_de_la_semana INT,
    Porteria_a_cero INT,
    Despejes_por_partido DECIMAL(4,1),
    Duelos_totales_ganados DECIMAL(4,1),
    Duelos_en_el_suelo_ganados DECIMAL(4,1),
    Duelos_aereos_ganados DECIMAL(4,1),
    Equipo VARCHAR(50),
    ID INT PRIMARY KEY
);
```

Figura 51. Creación tabla defensas

```sql
INSERT INTO Defensas (Jugador, Partidos_Totales_Jugados, Titular, Minutos_por_partido,
Equipo_de_la_semana, Porteria_a_cero, Despejes_por_partido, Duelos_totales_ganados,
Duelos_en_el_suelo_ganados, Duelos_aereos_ganados, Equipo, ID)

VALUES
('Cédric Soares', 32, 32, 87, 0, 7, 2.6, 3.4, 2.4, 1.0, 'Southampton', 3329),
('Pepe', 23, 23, 89, 2, 7, 3.8, 6.2, 2.3, 3.9, 'Beşiktaş JK', 20016),
('José Fonte', 8, 8, 86, 1, 3, 1.0, 4.8, 2.1, 2.6, 'West Ham United', 3960),
('Raphaël Guerreiro', 9, 6, 61, 0, 0, 0.9, 3.2, 2.9, 0.3, 'Borussia Dortmund', 5209),
('Jordi Alba', 33, 30, 83, 5, 15, 1.5, 3.7, 3.2, 0.5, 'F.C. Barcelona', 5211),
('Sergio Ramos', 26, 26, 88, 4, 3, 3.2, 5.9, 2.6, 3.3, 'Real Madrid C.F', 5201),
('Gerard Pique', 30, 29, 88, 4, 14, 4.1, 4.0, 1.8, 2.2, 'F.C. Barcelona', 5213),
('Nacho Fernández', 27, 22, 77, 4, 7, 1.8, 4.6, 3.1, 1.4, 'Real Madrid C.F', 5202),
('Lucas Vázquez', 33, 16, 51, 2, 3, 0.2, 4.0, 3.8, 0.2, 'Real Madrid C.F', 5200);
```

Figura 52. Rellenar datos en la tabla defensas de la información de SofasCore

3.3 Representación de datos con Power BI

En la era digital actual, los datos se han convertido en un recurso muy importante. Permiten tomar mejores decisiones y de manera más eficiente,

Pero pienso también que la manera de mostrar esos datos o esa información que nos proporcionan los datos es igual de importante. Para que la visualización sea atractiva y cumpla con los requisitos de mostrar la información que quiero he elegido Power BI.

Power BI es una herramienta de visualización de datos desarrollada por Microsoft. Permite convertir datos en información visual y comprensible mediante gráficos, *dashboards*, y reportes interactivos. Además, permite crear gráficos poder filtrar la información de manera interactiva, puede conectarse a una amplia variedad de fuentes de datos, como en este trabajo que nos conectamos a Excel y a Bases de datos MySQL, permite actualizar la información en tiempo real, es relativamente fácil de usar y tiene una opción de uso gratuita (Power BI, 2024).

3.3.1 Conexión desde Power BI a las fuentes de datos creadas anteriormente

Como he comentado anteriormente en este proyecto nos conectamos a dos fuentes de datos: Excel y Bases de datos MySQL.

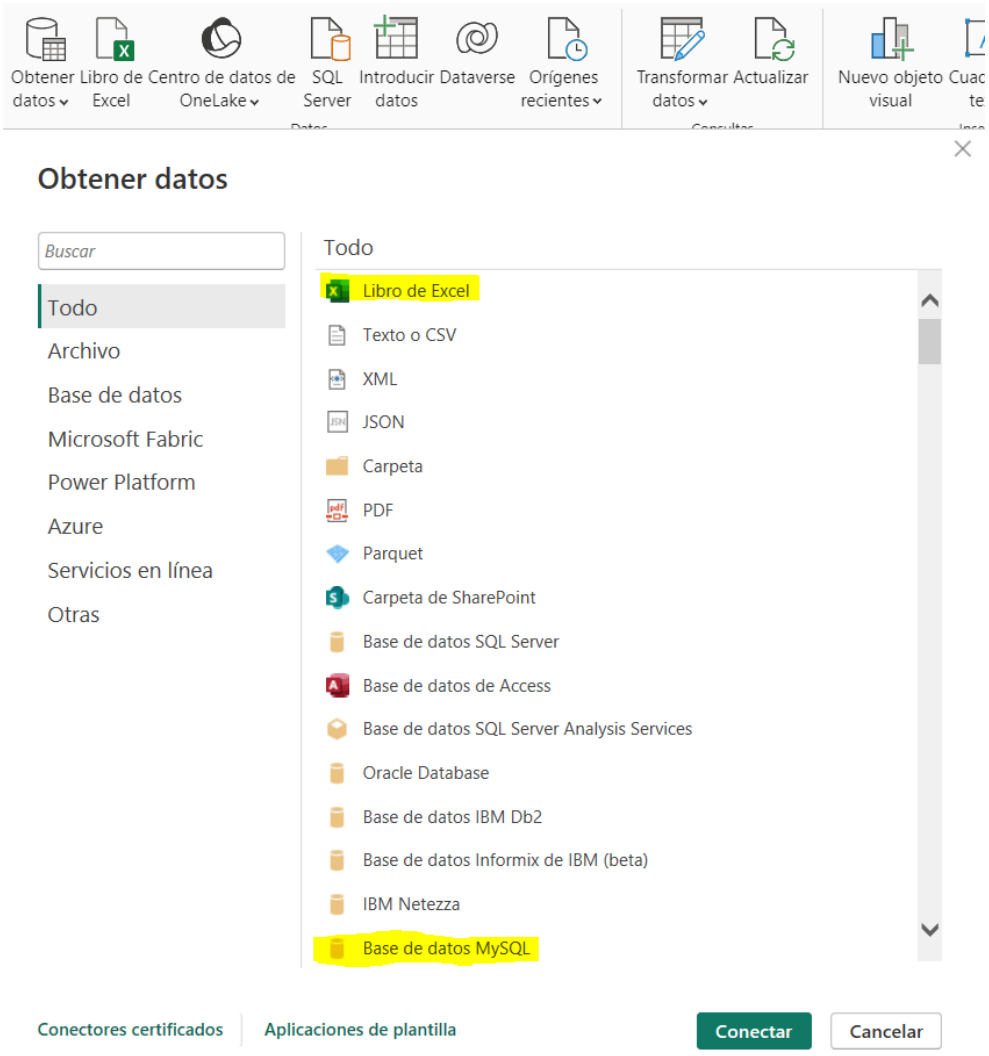


Figura 53. Conectores utilizados en Power BI

Para la importación de datos desde Excel no hay ningún problema. Buscas el Excel en la ruta donde los tengas guardados, que en este caso es la ruta que hemos indicado desde el análisis de datos con Python, pero para la importación de datos con la base de datos MySQL se necesitan conectores adicionales instalados en tu equipo, ya que Power BI no tiene nativamente incorporado el protocolo de comunicación de MySQL. Estos conectores permiten que Power BI se comunique correctamente utilizando los protocolos y formatos que MySQL entiende.

En mi caso me instalé dos conectores:

- mysql-connector-odbc-8.0.28-winx64
- mysql-connector-net-8.0.28

Es importante que los dos conectores tengan la misma versión ya que comprobé que con la versión más actualizada o con dos versiones diferentes puede dar problemas a la hora de conectarse.

Una vez instalados solo hay que ir a al conector de Power BI, Base de datos MySQL y poner el servidor y la base de datos.

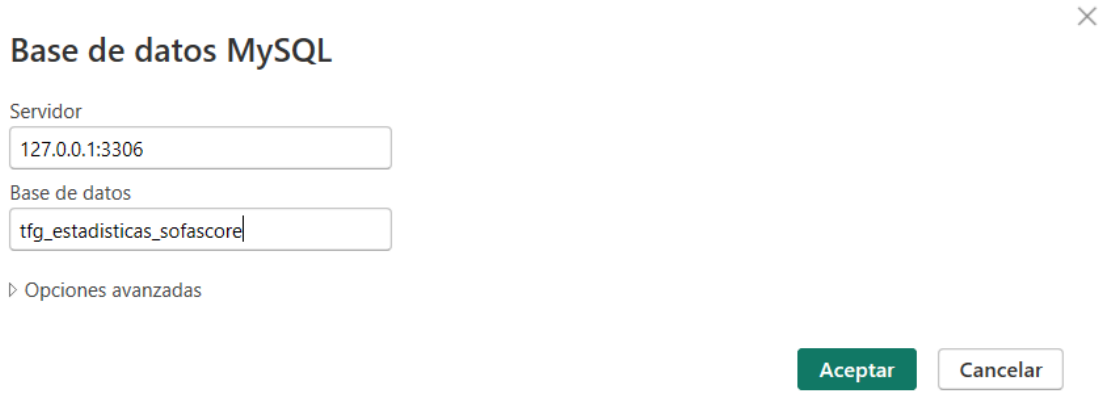
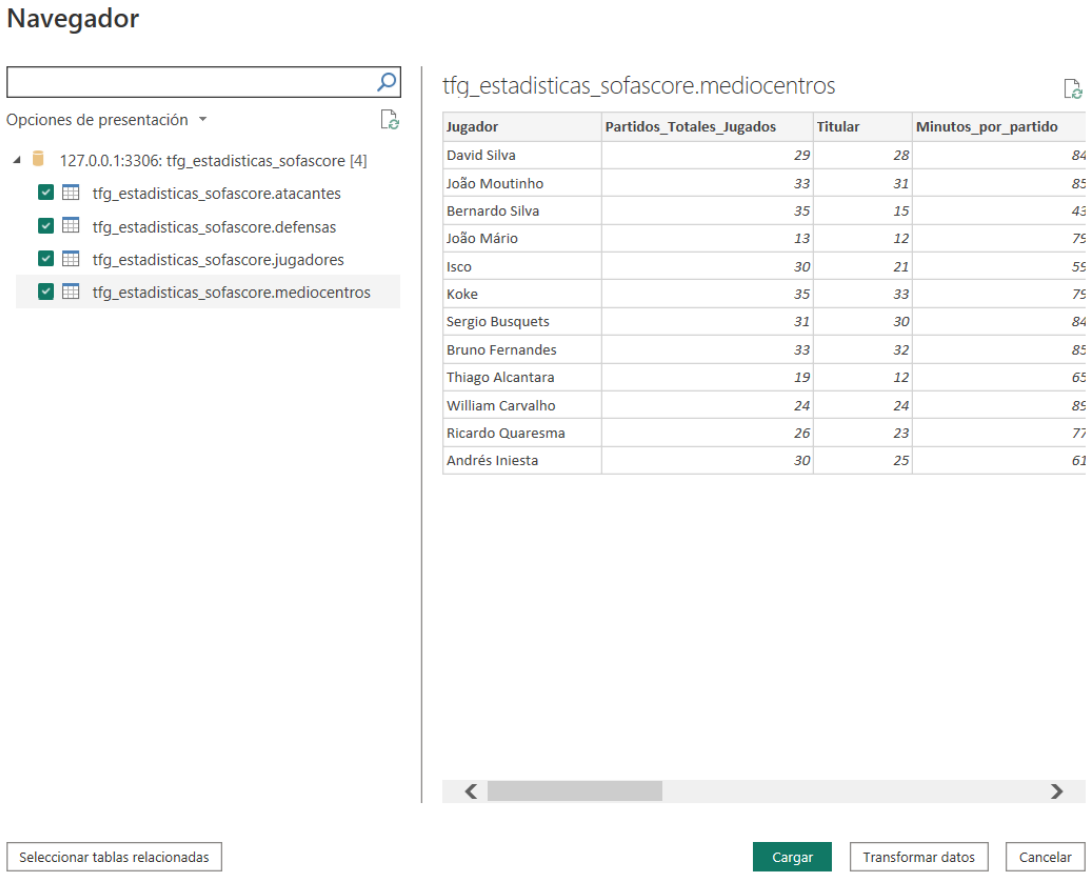


Figura 54. Conexión desde Power BI a MySQL

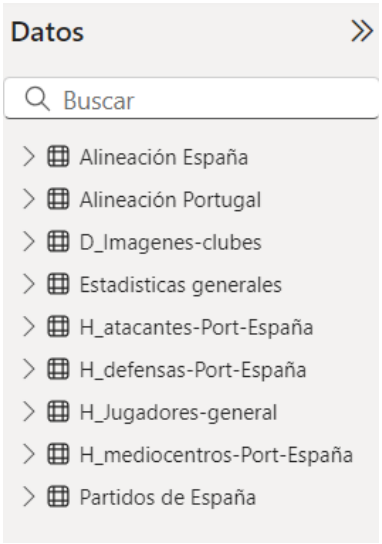
Cuando aceptas te lleva a la siguiente pestaña, donde puedes ver las tablas que tienes en tu base de datos y si seleccionas las tablas puedes ver una vista previa de los datos de la tabla.



Jugador	Partidos_Totales_Jugados	Titular	Minutos_por_partido
David Silva	29	28	84
João Moutinho	33	31	85
Bernardo Silva	35	15	43
João Mário	13	12	75
Isco	30	21	55
Koke	35	33	75
Sergio Busquets	31	30	84
Bruno Fernandes	33	32	85
Thiago Alcantara	19	12	65
William Carvalho	24	24	85
Ricardo Quaresma	26	23	77
Andrés Iniesta	30	25	61

Figura 55. Conexión desde Power BI a MySQL (2)

Una vez le damos a cargar, ya podemos utilizar los datos para hacer nuestro informe, ya que ya tendríamos cargados los datos que teníamos en formato de Excel y los de la base de datos MySQL.



3.3.2 Imágenes de los clubes

A la hora de realizar el informe tenía una pequeña complicación, ya que quería que, mediante un filtro de cada jugador, en cada uno de ellos les apareciera el escudo del equipo donde estuvieron en la temporada en la que les tenían que convocar en la selección.

Para resolver este problema y que Power BI entendiera como utilizar estas imágenes hice una serie de pasos.

Primero me descargue las imágenes en formato PNG de los clubs que iba a necesitar y los guarde en una ruta en local de mi ordenador.

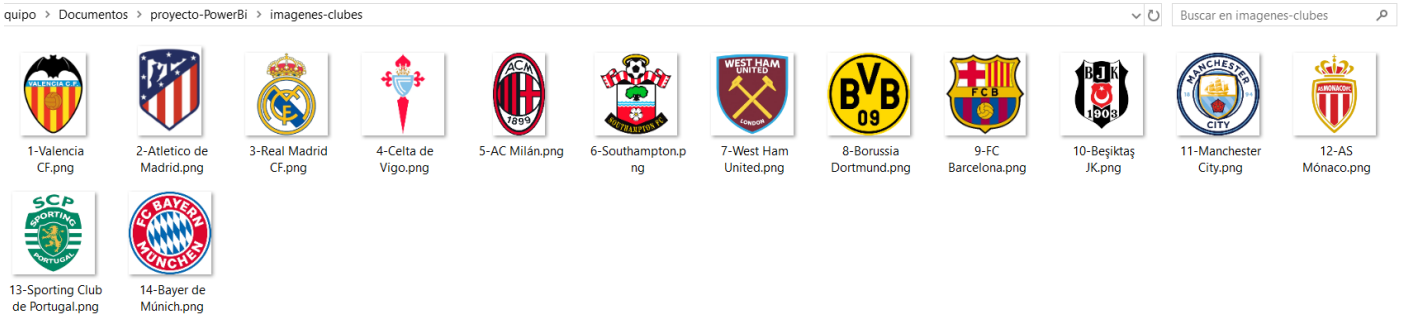


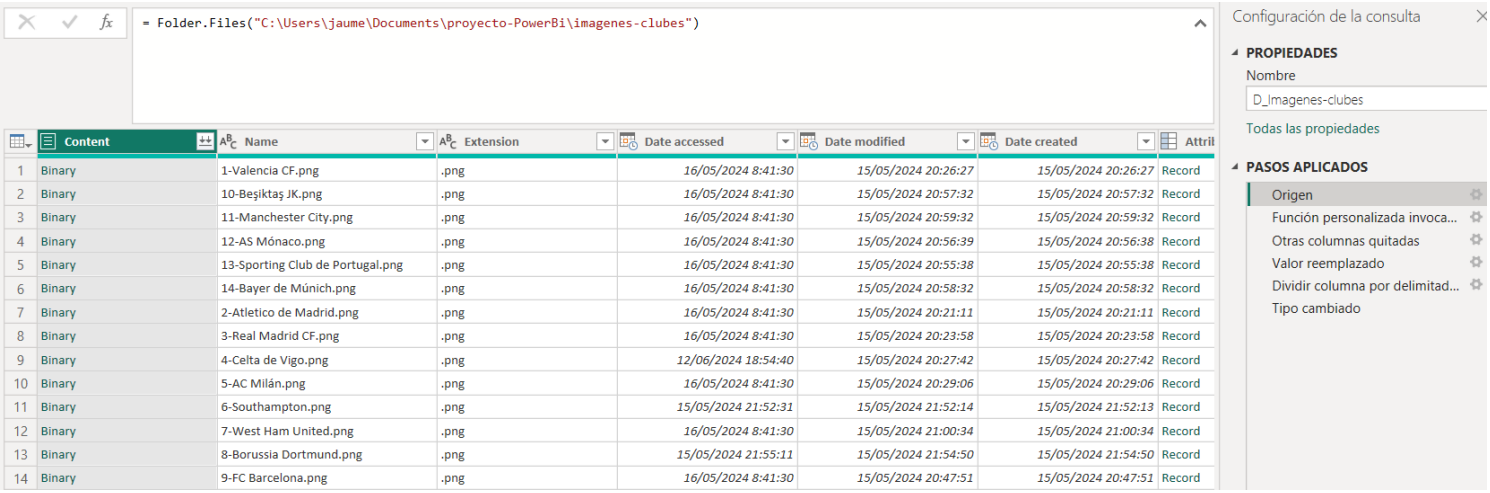
Figura 56. Descarga de imágenes de los clubes

Una vez tenía las imágenes de los clubes descargadas tenía que cargarlas en Power BI y en el apartado de **Transformar Datos (Power Query)** hacer transformaciones para que detecte las imágenes y puedan ser utilizadas. A la hora de guardar las imágenes les coloque un número, que es el mismo ID que también cree en cada tabla (atacantes, mediocentros, defensas) para que puedan ser relacionadas en Power BI.

Tenía que convertir contenido Binario a una cadena de texto codificada en Base64 para poder utilizarla como una dirección URL de la imagen. Para ello, Power Query tiene una opción para poder crear funciones y poder aplicarlas en algún paso de Power Query.

Figura 57. Función de Binario a texto en Base64 en Power Query

Ahora tenía que cargar mi carpeta con las imágenes en Power BI y en Power Query aplicar la función que había creado previamente. Para ello y con lenguaje M (que es el utilizado en Power Query) hice mis transformaciones para que el resultado final fuera una tabla con 3 columnas (ID Equipo, Nombre Equipo, Imágenes-club-bin-imágenes), esta última columna es donde están guardadas cada imagen de cada club en formato de texto en Base64 para poder utilizarlas como URL de imagen. Voy a indicar en imágenes como quedo el proceso de transformación.



	Content	Name	Extension	Date accessed	Date modified	Date created	Attril
1	Binary	1-Valencia CF.png	.png	16/05/2024 8:41:30	15/05/2024 20:26:27	15/05/2024 20:26:27	Record
2	Binary	10-Beşiktaş JK.png	.png	16/05/2024 8:41:30	15/05/2024 20:57:32	15/05/2024 20:57:32	Record
3	Binary	11-Manchester City.png	.png	16/05/2024 8:41:30	15/05/2024 20:59:32	15/05/2024 20:59:32	Record
4	Binary	12-AS Mónaco.png	.png	16/05/2024 8:41:30	15/05/2024 20:56:39	15/05/2024 20:56:38	Record
5	Binary	13-Sporting Club de Portugal.png	.png	16/05/2024 8:41:30	15/05/2024 20:55:38	15/05/2024 20:55:38	Record
6	Binary	14-Bayer de Múnich.png	.png	16/05/2024 8:41:30	15/05/2024 20:58:32	15/05/2024 20:58:32	Record
7	Binary	2-Atletico de Madrid.png	.png	16/05/2024 8:41:30	15/05/2024 20:21:11	15/05/2024 20:21:11	Record
8	Binary	3-Real Madrid CF.png	.png	16/05/2024 8:41:30	15/05/2024 20:23:58	15/05/2024 20:23:58	Record
9	Binary	4-Celta de Vigo.png	.png	12/06/2024 18:54:40	15/05/2024 20:27:42	15/05/2024 20:27:42	Record
10	Binary	5-AC Milán.png	.png	16/05/2024 8:41:30	15/05/2024 20:29:06	15/05/2024 20:29:06	Record
11	Binary	6-Southampton.png	.png	15/05/2024 21:52:31	15/05/2024 21:52:14	15/05/2024 21:52:13	Record
12	Binary	7-West Ham United.png	.png	16/05/2024 8:41:30	15/05/2024 21:00:34	15/05/2024 21:00:34	Record
13	Binary	8-Borussia Dortmund.png	.png	15/05/2024 21:55:11	15/05/2024 21:54:50	15/05/2024 21:54:50	Record
14	Binary	9-FC Barcelona.png	.png	16/05/2024 8:41:30	15/05/2024 20:47:51	15/05/2024 20:47:51	Record

Figura 58. Carga inicial de la carpeta donde se encuentran las imágenes en Power Query

Así llega la información de la carpeta, ahora voy a poner el lenguaje en M para hacer las transformaciones mencionadas anteriormente, subrayando el paso donde invoco la función que he creado en el paso anterior.

En la otra parte del código hago filtrado de la información de la tabla, como quitar las columnas que no utilizo, renombrar columnas, separar en dos columnas el ID del equipo del nombre del equipo, entre otras.

Figura 59. Transformación en Power Query de la tabla de las imágenes

Con estos pasos consigo el resultado final esperado con las tres columnas mencionadas anteriormente.

	ID Equipo	Nombre Equipo	Imagenes-clubes-bin-img
1	1	Valencia CF	iVBORw0KGgoAAAANSUhEUgAAAE8AAABkCAYAAADHX21zAAAAAX...
2	10	Beşiktaş JK	iVBORw0KGgoAAAANSUhEUgAAAGQAAABkCAYAAABw4pVUAAAAA...
3	11	Manchester City	iVBORw0KGgoAAAANSUhEUgAAAGQAAABkCAYAAABw4pVUAAAAA...
4	12	AS Mónaco	iVBORw0KGgoAAAANSUhEUgAAAGQAAABkCAYAAABw4pVUAAAAA...
5	13	Sporting Club de Portugal	iVBORw0KGgoAAAANSUhEUgAAAGQAAABkCAYAAABw4pVUAAAAA...
6	14	Bayer de Múnich	iVBORw0KGgoAAAANSUhEUgAAAGQAAABkCAYAAABw4pVUAAAAA...
7	2	Atletico de Madrid	iVBORw0KGgoAAAANSUhEUgAAAIEAAAC0CAYAAACzF09OAAAAAXN...
8	3	Real Madrid CF	iVBORw0KGgoAAAANSUhEUgAAAGQAAABkCAYAAABw4pVUAAAAA...
9	4	Celta de Vigo	iVBORw0KGgoAAAANSUhEUgAAAGQAAABkCAYAAABw4pVUAAAAA...
10	5	AC Milán	iVBORw0KGgoAAAANSUhEUgAAAD8AAABkCAYAAAAxFujrAAAAAXN...
11	6	Southampton	iVBORw0KGgoAAAANSUhEUgAAAGQAAABkCAYAAABw4pVUAAAAA...
12	7	West Ham United	iVBORw0KGgoAAAANSUhEUgAAAGQAAABkCAYAAABw4pVUAAAAA...
13	8	Borussia Dortmund	iVBORw0KGgoAAAANSUhEUgAAAGQAAABkCAYAAABw4pVUAAAAA...
14	9	FC Barcelona	iVBORw0KGgoAAAANSUhEUgAAAGIAAABkCAYAAAB9/OUTAAAAAX...

Todas las propiedades

PASOS APLICADOS

- Origen
- Función personalizada invoca...
- Otras columnas quitadas
- Valor reemplazado
- Dividir columna por delimitad...
- Tipo cambiado

Figura 60. Tabla final de imágenes después de las transformaciones

Por último, en la columna de Imágenes-clubes-bin-img hay que indicarle en categoría de datos que se utilice como Dirección URL de la imagen.

3.3.3 Relaciones de las tablas

En este modelo de datos las relaciones son bastante sencillas, ya que solo quiero que se relacionen las tablas de atacantes, mediocentros y defensas con la de las imágenes de los clubes. Ya que quiero que cuando selecciones a un jugador muestre el club en el que ha estado. Las otras tablas las utilizo en el informe, pero no necesito que se relacionen entre sí, ya que solo quiero que muestre la información de cada tabla individualmente.

Las relaciones que tengo son de uno a muchos de la tabla de imágenes (tabla de dimensiones) a las tres tablas (atacantes, mediocentros, defensas) que actúan como tablas de hechos y se relacionan mediante el campo ID Equipo.

Figura 61. Relaciones entre tablas

3.3.4 Visualización del informe de Power BI

El informe contiene 6 pestañas. Voy a explicar su funcionamiento y en que consiste cada una.

3.3.4.1 Pestaña Partidos – España

Figura 62. Pestaña Partidos-España

Este informe muestra los partidos de España en el Mundial y sus resultados. Para realizar este informe utilizo los datos obtenidos con Python. Utilizo el Excel que contiene la tabla Partidos España. Se puede observar a primera vista como ha finalizado el partido España en cada partido que ha jugado. También hay dos botones al lado del partido que vamos a analizar, en este caso Portugal vs España.

A la izquierda hay **botón de información** que al pulsarlo aparece una nueva ventana dentro de esta pestaña mostrando la información del partido, como que día se jugó, en que estadio, en que fase de la competición estaban, el arbitro y los entrenadores de cada equipo.

Figura 63. Pestaña Partidos-España (botón información)

Esta funcionalidad la consigo en Power BI mediante marcadores. Los marcadores son una funcionalidad que les permite a los usuarios guardar el estado actual de una página del informe incluyendo la configuración de los filtros, el estado de la visualización o la posición de las selecciones realizadas en el informe.

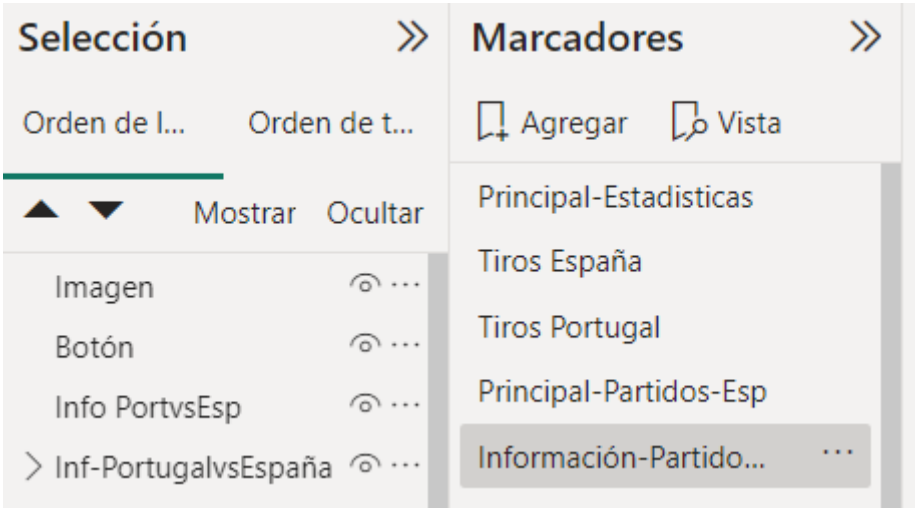


Figura 64. Marcador (botón información)

Como vemos en para que se muestre el marcador guardamos el estado de que todo sea visible. En cambio, cuando queremos que se vea el informe sin la información del marcador tendremos que ocultarlo como vemos en la Figura 65.

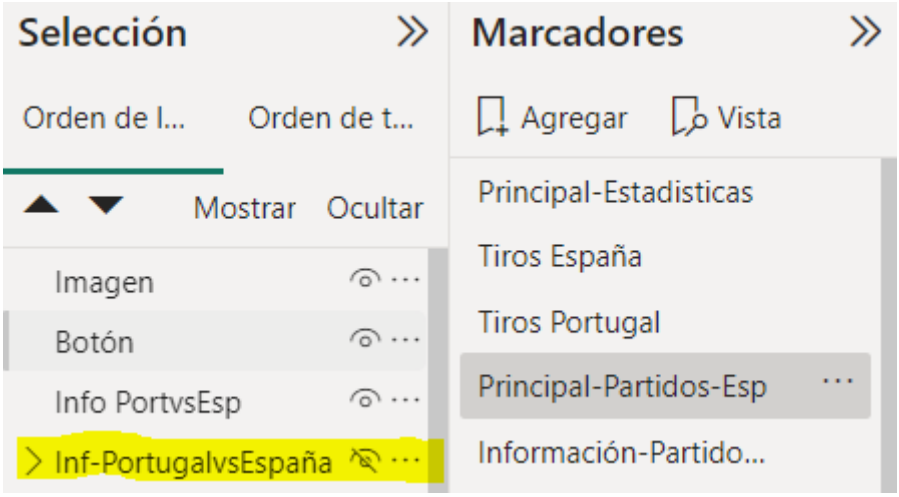


Figura 65. Marcador (informe sin la información del partido)

De esta forma guardamos el estado en el marcador y solo tenemos que darle acción al botón para que utilice el marcador que deseamos. En este caso, cuando queremos que salga la información del partido, cuando le damos al botón de información, le indicamos la acción a ese botón para que utilice el marcador ya guardado previamente “Información-Partidos”.

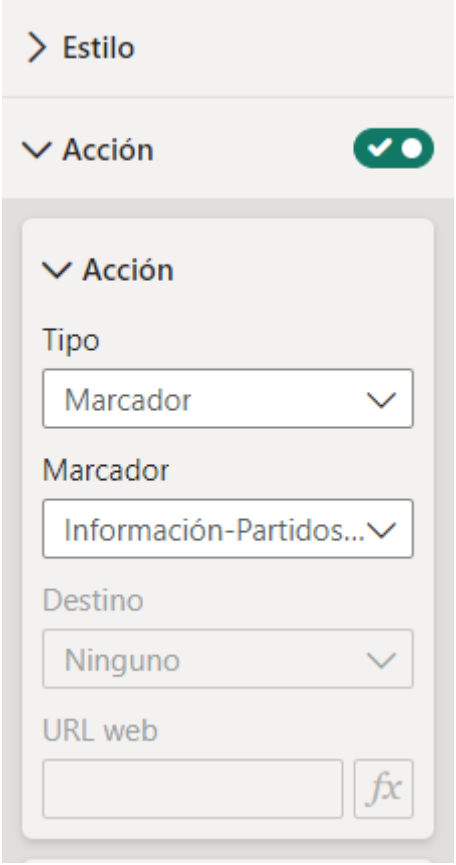


Figura 66. Acción al marcador de información

Por otro lado, en el **botón de la derecha** del partido le vamos a dar acción, pero en este casa una acción de navegación de pestaña, para que al pulsarlo navegue a la pestaña que le indiquemos. En este caso a la pestaña de Estadísticas generales.

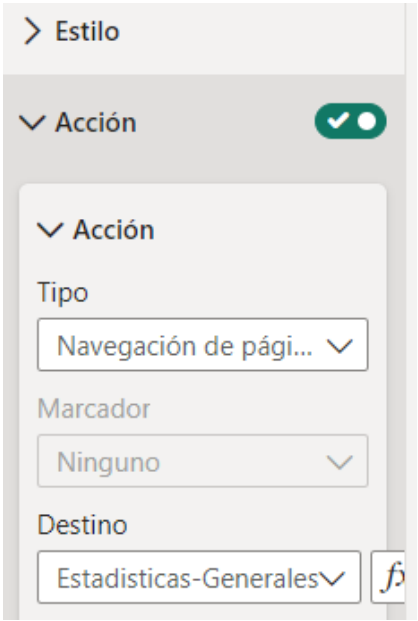


Figura 67. Acción al navegación de página a Estadísticas-Generales

3.3.4.2 Estadísticas-Generales Portugal vs España

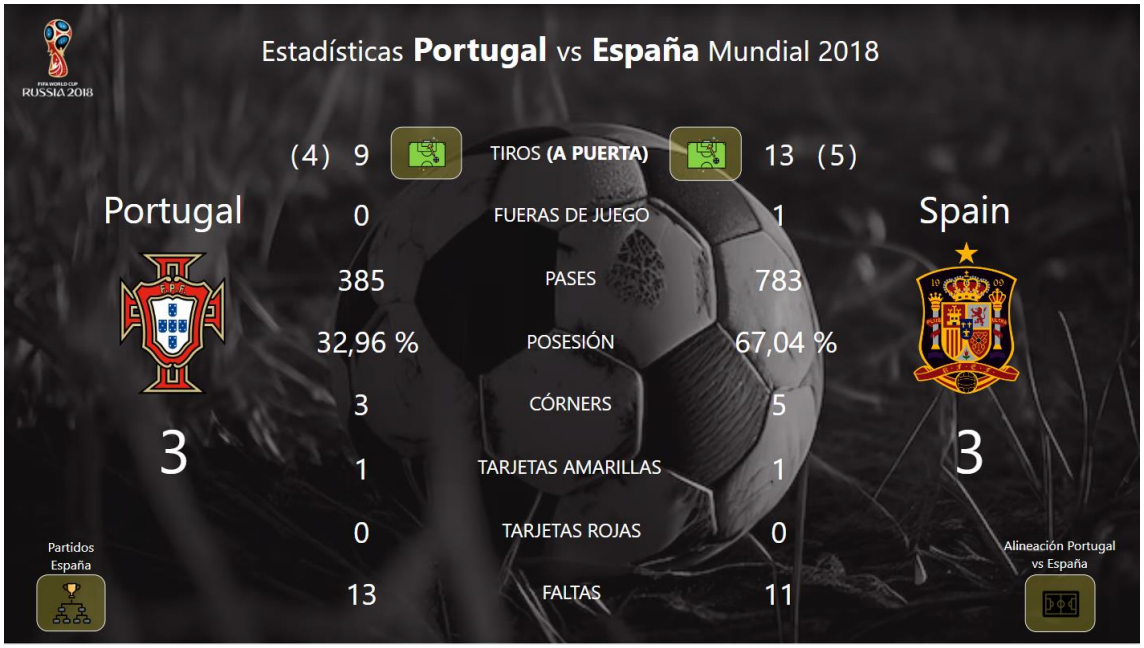


Figura 68. Pestaña Estadísticas Portugal vs España Mundial 2018

Una vez accedemos en esta pestaña, nos muestra el resultado del partido entre Portugal y España y nos muestra las estadísticas generales del partido. La información la consigo a través de la tabla de Estadisticas_Partido_Esp_Port.xlsx, también extraído anteriormente con el análisis en Python.

Se puede observar datos sobre los tiros del partido y cuantos fueron a puerta, los fueras de juego, el número de pases que hizo cada equipo, la posesión en porcentaje, los córners, las tarjetas amarillas, las tarjetas rojas y cuantas faltas cometió cada equipo.

En la fila de tiros (a puerta) se pueden observar dos botones. Cada uno de los botones es un marcador que ya he explicado anteriormente, en que si pulsas el botón se abrirá el grafico de tiros de cada equipo, extraído en el análisis con Python que se muestra en las figuras 69 y 70.

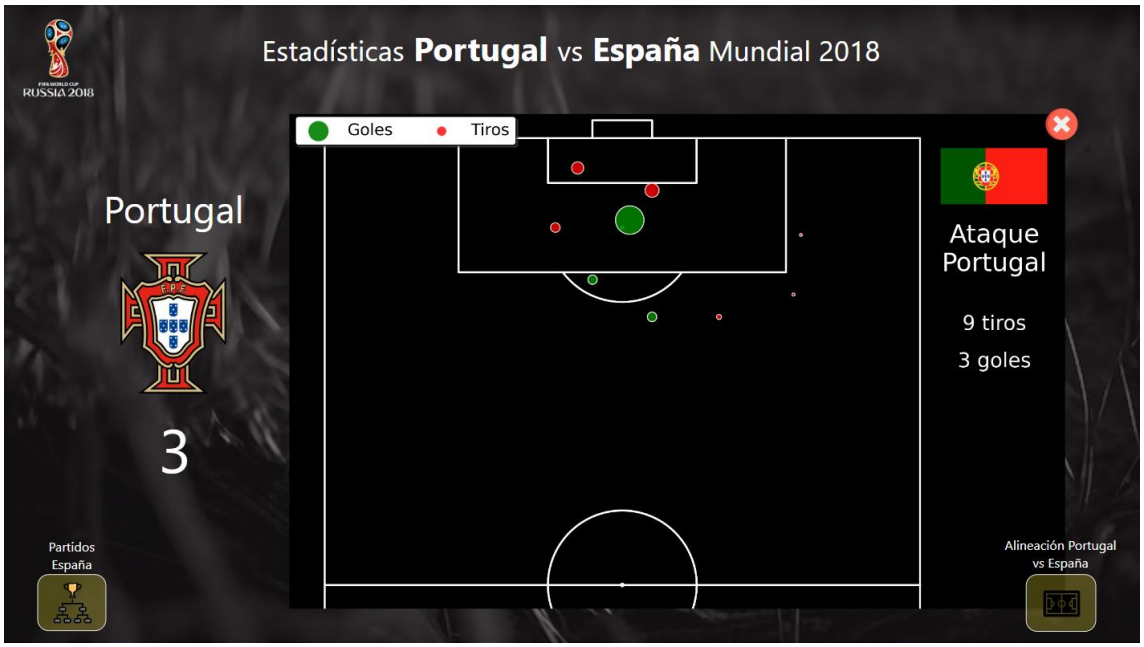


Figura 69. Botón tiros Portugal

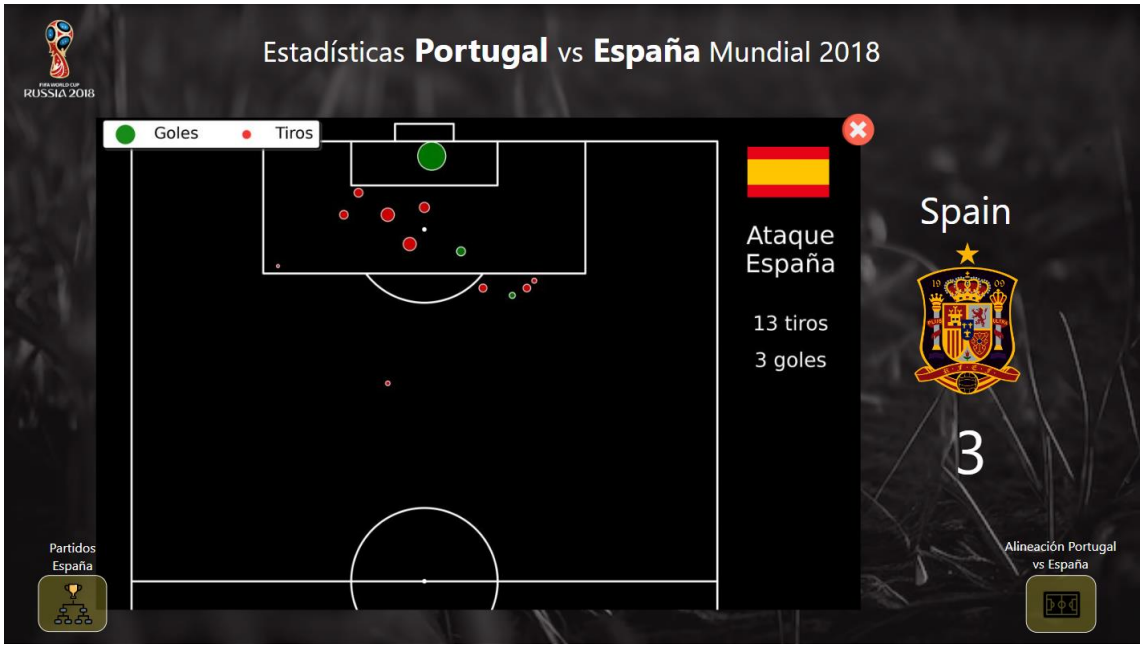


Figura 70. Botón tiros España

Para finalizar esta pestaña, tiene dos botones inferiores. En este caso los dos botones son de navegación entre pestañas. El botón de "Partidos España" navega a la pestaña anterior y el botón de Alineación Portugal vs España navega a la pestaña que voy a explicar en el siguiente punto.

3.3.4.3 Alineación Portugal vs España Mundial 2018

Figura 71. Pestaña Alineación Portugal vs España

En esta pestaña muestro la alineación con la que jugaron en ese partido Portugal y España. También muestro que con formación jugaron, los suplentes y a nivel visual muestro quien metió gol, quien fui sustituido y a quien le sacaron tarjetas en el partido.

Arriba a la izquierda se encuentra un botón de navegación de páginas, que sirve para volver a la pestaña anterior.

Esta pestaña sirve de enlace para las pestañas de atacantes, mediocentros y defensas, ya que cada jugador sirve como botón. Ese botón invoca un marcador. Por lo tanto, al pulsar encima de cada jugador, navega a la pestaña de atacantes, mediocentros o defensas según el jugador que quieras analizar.

3.3.4.4 Pestañas de atacantes, mediocentros y defensas

Voy a explicar las tres pestañas a la vez, ya que todas en su funcionamiento son iguales, lo único que las diferencia son los datos que voy a mostrar en cada una de ellas, ya que como he explicado anteriormente, en cada posición quería mostrar diferentes análisis por posición.

En este caso voy a hacer el ejemplo con los mediocentros. Como ejemplo he pulsado en la pestaña de alineación a Sergio Busquets.

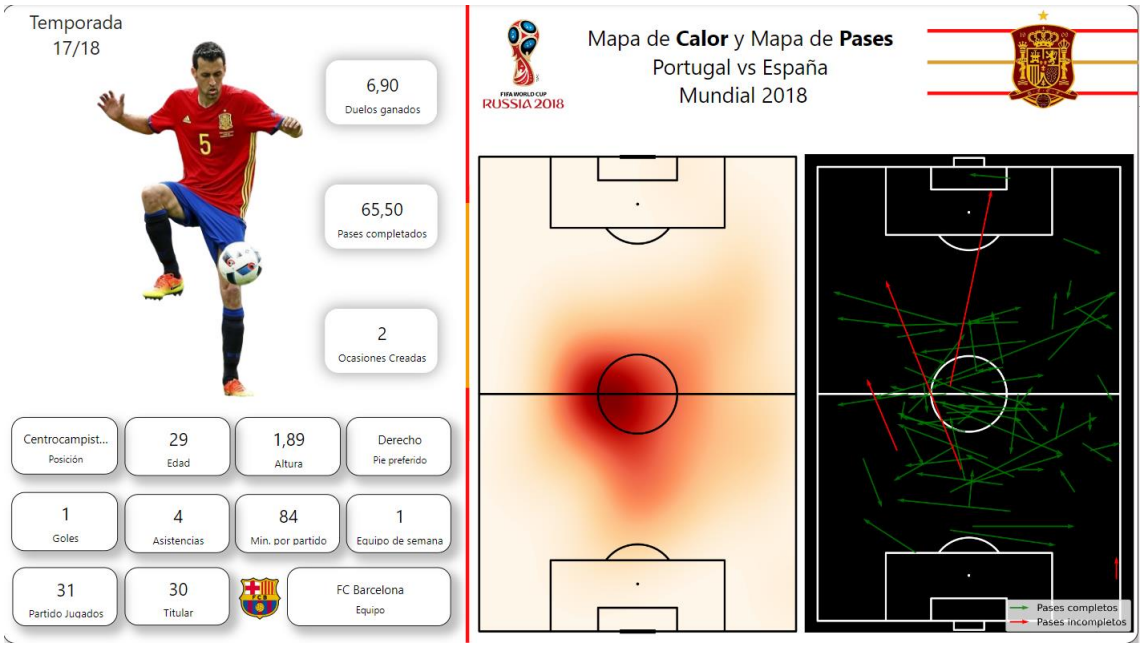


Figura 72. Pestaña ejemplo de mediocentros

Esta pestaña muestra la información de cada jugador. En la parte izquierda se muestran datos relevantes de la temporada 17/18, ya que son alguna de las razones por las que han sido convocados a jugar el mundial. Se muestra la imagen del jugador y en este caso al ser mediocentros, a la derecha del jugador se muestra lo que son para mi los datos mas relevantes para esa posición, como son el promedio de duelos ganados por partido, el promedio de pases completados por partido y el promedio de ocasiones creados por partido. En la parte inferior de la imagen se muestra información general del jugador y se muestra la imagen del club donde jugaba en esa temporada. Toda esta información ha sido extraída de la base de datos MySQL que he explicado anteriormente, en este caso la tabla de mediocentros.

En la parte derecha se muestra el mapa de calor y el mapa de pases de cada jugador que tuvo minutos en ese partido. Los gráficos han sido obtenidos del análisis en Python.

Para lograr este funcionamiento he creado un marcador por cada jugador, en el que guardo los datos, el estado del filtro y la navegación de la pestaña. También cada jugador tiene un filtro con su nombre seleccionado y de esa forma cuando guardo el marcador se muestran los datos de cada jugador ya filtrado.

4. Conclusiones

El proyecto demuestra la capacidad y el impacto del análisis de datos y más concreto en el análisis de datos en el ámbito del fútbol profesional, aplicando una metodología que incluye la recopilación, almacenamiento, análisis y visualización de datos. A través del uso de la librería Statsbombpy, Python, SQL y Power BI, se ha logrado establecer un sistema que facilita la toma de decisiones informadas y estratégicas para entrenadores, jugadores y directivos y la consulta de datos de personadas interesadas en el mundo del futbol.

La integración de tecnologías avanzadas y metodologías analíticas en el fútbol profesional representa un avance hacia la modernización del deporte. Equipos que adoptan estas prácticas están mejor posicionados para competir a altos niveles, reflejando un compromiso con la excelencia y la innovación.

Este proyecto no solo demuestra el valor del análisis de datos en el fútbol, sino que también pone en valor la importancia de combinar diversas tecnologías y herramientas para obtener datos relevantes y que sirvan para tomar mejores decisiones.

Además, se demuestra la importancia de la visualización de los datos, ya que un dato enseñado de dos formas diferentes visualmente puede significar cosas totalmente diferentes. Es importante que sea visualmente atractivo y con sentido para que la persona que visualiza esa información no tome conclusiones erróneas.

A nivel personal también tomo conclusiones muy positivas, ya que he conseguido el objetivo de lograr combinar la tecnología con el deporte, demostrando que los datos pueden transformar la compresión del fútbol y a la vez puedo demostrar lo que he aprendido durante estos años.

5. Propuesta de trabajos futuros

Respecto a propuestas para mejorar o realizar trabajos futuros, algunas de ellas serían los siguientes:

- **Añadir mas partidos de esa competición:** Aprovechando ya el código y el análisis realizado, una idea es mejorar el análisis y hacer lo mismo para todos los partidos que jugó España en ese mundial. Si se quiere profundizar más aún, se podría hacer el análisis de todos los partidos del mundial, independiente a si juega España o no. Se podría lograr relativamente fácil reutilizando el código ya hecho, añadiendo jugadores en la base de datos ya creada y añadiendo imágenes de más clubes y selecciones.

- **Incorporar Inteligencia Artificial:** Utilizar técnicas de aprendizaje automático para desarrollar modelos predictivos que anticipen el rendimiento de los jugadores y puedan ayudar a predecir resultados, identificar posibles lesiones y recomendar estrategias de juego.

- **Desarrollo de aplicaciones móviles:** Crear aplicaciones móviles para que entrenadores y jugadores puedan observar esa información en cualquier momento y lugar.

6. Bibliografía

Github statsbomb. (2024). (Github) Obtenido de https://github.com/statsbomb/open-data

Jupyter Notebook. (2024). (Jupyter) Obtenido de https://jupyter.org/

Matplotlib. (2024). (Matplotlib) Obtenido de https://matplotlib.org/

mplsoccer. (2021). (mplsoccer) Obtenido de https://mplsoccer.readthedocs.io/en/latest/

Numpy. (2024). (Numpy) Obtenido de https://numpy.org/

os. (2023). (python) Obtenido de https://docs.python.org/es/3.10/library/os.html

Pandas. (2024). (Pandas) Obtenido de https://pandas.pydata.org/docs/user_guide/index.html

Power BI. (2024). (Microsoft) Obtenido de https://learn.microsoft.com/es-es/power-bi/fundamentals/power-bi-overview

Python. (2024). (Python) Obtenido de https://es.wikipedia.org/wiki/Python

Statsbomb. (2023). (Statsbomb) Obtenido de https://statsbomb.com/es/quienes-somos/

Wikipedia-SofasCore. (2023). (Wikipedia) Obtenido de https://en.wikipedia.org/wiki/Sofascore