

BASE DE DATOS 3

ESCUELA TECNOLOGICA INSTITUTO TECNICO CENTRAL

SISTEMA DE ANÁLISIS DEPORTIVO CONSUMIDO

PRESENTADO POR:

Matthew Espinosa Santiago

Carvajal

PRESENTADO A:

Brayan Sebastián Yepes

BOGOTA DC

MAYO 2025

SPRINT 1: Planeacion y Diseño del Proyecto (Semana 1 a 2)

JUSTIFICACIÓN DEL PROYECTO

En el mundo del análisis deportivo actual, especialmente en el fútbol, tomar decisiones basadas en datos ya no es una opción, sino una necesidad. Entrenadores, analistas y equipos buscan cada vez más apoyo en herramientas tecnológicas que les permitan interpretar el rendimiento en tiempo real. Sin embargo, acceder a datos confiables, estructurados y actualizados puede ser complicado, costoso o poco accesible para muchos.

Este proyecto surge precisamente para cubrir esa necesidad. La idea es desarrollar una solución que automatice la recolección, almacenamiento y análisis de datos futbolísticos a partir de una API pública y gratuita: football-data.org. A través de esta fuente, el sistema podrá extraer información sobre partidos, equipos y estadísticas clave, para luego procesarla y presentarla de forma clara mediante visualizaciones y análisis comparativos. Esto no solo ahorra tiempo al evitar el ingreso manual de datos, sino que además garantiza que la información provenga de una fuente confiable y actualizada.

La solución será desarrollada con herramientas accesibles pero robustas: Python como lenguaje principal, Flask para la construcción de la API backend, y una base de datos relacional (SQLite o PostgreSQL) para almacenar la información. Todo esto se desplegará en una instancia EC2 de AWS en el SO de Linux, lo que asegura un entorno escalable, económico y autónomo.

Además, este proyecto tiene potencial para crecer en el futuro. La arquitectura está pensada para poder adaptarse fácilmente a otros deportes, ligas o tipos de análisis, e incluso podría integrarse con sistemas de scouting, modelos de predicción o plataformas de rendimiento en tiempo real.

2. Descripción del Proyecto

El proyecto consiste en el desarrollo de un sistema que automatiza la recolección, almacenamiento y exposición de datos estadísticos del fútbol, utilizando como fuente la API pública football-data.org. El núcleo del sistema será una API REST creada con **Flask**, que actuará como intermediario entre la fuente de datos y los usuarios finales.

Los datos obtenidos —como resultados de partidos, información de equipos y estadísticas clave (goles, posesión, xG, entre otros)— serán almacenados en una base de datos relacional, utilizando **PostgreSQL** como motor principal. Una vez estructurados, estos datos estarán disponibles tanto para consulta a través de la API como para visualización en dashboards interactivos.

El sistema ofrecerá las siguientes funcionalidades principales:

- Recolección de datos históricos y actuales sobre partidos, equipos y estadísticas relevantes.
- Almacenamiento y normalización de los datos recolectados para facilitar su análisis posterior.
- Exposición de la información a través de una API propia, permitiendo su consumo por otras aplicaciones o servicios.
- Visualización de datos mediante dashboards dinámicos que faciliten la interpretación de tendencias y comparativas.
- Soporte para análisis externos, permitiendo que otras herramientas se conecten al sistema para realizar consultas o procesamientos adicionales.
- En conjunto, esta solución busca simplificar el acceso a datos deportivos estructurados, habilitando su uso en proyectos de análisis, visualización y toma de decisiones basadas en información real y actualizada.

3. Objetivos

Objetivo General: Desarrollar una API REST funcional y documentada, que consuma, almacene y exponga datos deportivos reales, utilizando estándares de diseño y despliegue profesional.

Objetivos Específicos:

- Integrar y consumir datos desde una API (football-data.org).
- Crear una base de datos PostgreSQL para almacenamiento.
- Desarrollar en Flask para exposición de los datos.
- Desplegarla en una instancia EC2 de AWS.
- Implementar paneles de visualización para el análisis de datos deportivos.
- Establecer mecanismos de seguridad y autenticación básicos.

4. Alcance Incluye:

- Conexión con una API.
- Base de datos.
- Backend con Flask.
- Despliegue inicial en AWS EC2.
- Panel básico de visualización y documentación técnica o dashboard.

No incluye:

- Aplicaciones móviles.
- Funcionalidades de login avanzado o autenticación múltiple.
- Inteligencia artificial o predicción automática de resultados.

5. Cronograma Tentativo (por sprints)

Sprint	Semana	Entregables principales
1	1-2	Acta, WBS, ERD, roles, BPMN, configuración inicial
2	3-4	Conexión con API y pruebas en local
3	5-6	Modelado base de datos y endpoints
4	7-8	Visualización, integración y seguridad
5	9-10	Pruebas finales, presentación y despliegue completo

6. Requisitos del Proyecto • API pública football-data.org.

- PostgreSQL 13 o superior.
- Python 3.10, Flask, SQLAlchemy, requests.
- AWS EC2 Ubuntu Server 22.04.
- Herramientas de apoyo: Trello, GitHub, DBeaver, ngrok, Insomnia, Jira.

7. Roles y Responsabilidades

Nombre	Rol	Responsabilidad principal
Santiago Carvajal	Líder del Proyecto	Dirección, gestión, pruebas, desarrollo backend y despliegue
Matthew Espinosa	DBA y Arquitecto	Diseño BD, Desarrollo Básico, pruebas y soporte técnico

8. Supuestos

- El API football-data.org mantendrá su disponibilidad durante el proyecto ya que es algo público.
- El equipo contará con acceso estable a Internet y servicios de AWS ya que es nuestro entorno y pues todavía no nos cobran a ninguno.

9. Restricciones

- No saber de lo que nos piden.
- No saber usar la IA.
- No hacer nada, pues ya que el internet lo tiene todo.

10. Aprobación del Proyecto

Nombre	Rol	Firma	Fecha
Santiago Carvajal	Líder del Proyecto	San\$iag'Carvaj,IF	26/05/2025
Matthew Espinosa	DBA / Arquitecto	Ma1hewEspin'sa	26/05/2025

WBS DEL PROYECTO DE ANÁLISIS DEPORTIVO CON API

1. Inicio del Proyecto

- 1.1 Definición del problema
- 1.2 Justificación
- 1.3 Objetivos generales y específicos
- 1.4 Acta de constitución del proyecto

2. Planeación

- 2.1 Asignación de roles (Santiago - Backend, Matthew - Visualización)
- 2.2 Identificación de herramientas (API, Python, Flask, BD, EC2)

3. Análisis y Diseño

- 3.1 Consulta a la API
- 3.2 Definición de endpoints y estructura de datos

4. Desarrollo e Implementación

- 4.1 Conexión con API externa usando Python
- 4.2 Creación y conexión con base de datos (SQLite o PostgreSQL)
- 4.3 Desarrollo del backend con Flask
- 4.4 Procesamiento de datos y limpieza

5. Visualización y Consumo de Datos

- 5.1 Desarrollo de interfaz visual o dashboard (local o web)
- 5.2 Visualización de métricas: goles, posesión, xG, etc.
- 5.3 Exportación o visualización de resultados

6. Pruebas y Ajustes

- 6.1 Pruebas de conexión API y rendimiento
- 6.2 Validación de integridad de los datos en BD
- 6.3 Validación de visualización y lógica de negocio

7. Documentación y Entrega

- 7.1 Capturas de evidencia

7.2 Informe final del proyecto

7.3 Presentación del producto terminado

DEFINICIÓN DE USUARIOS DEL SISTEMA

1. Administrador del Sistema

Descripción:

Usuario con conocimientos técnicos, generalmente parte del equipo de desarrollo o mantenimiento del sistema.

Responsabilidades:

- Configurar la conexión con la API externa.
- Administrar la base de datos (crear, eliminar, actualizar).
- Controlar el acceso al sistema.
- Ejecutar mantenimientos y actualizaciones del backend.

Permisos:

- Acceso completo al sistema
- Control de API keys y tokens
- Modificación de scripts y endpoints
- Visualización y edición total de los datos

2. Analista Deportivo

Descripción:

Profesional que interpreta los datos recolectados por el sistema para generar informes, análisis y recomendaciones.

Responsabilidades:

- Consultar estadísticas históricas y actuales.
- Interpretar métricas como goles, posesión, tarjetas, rendimiento ofensivo/defensivo, etc.
- Exportar gráficos o tablas para presentaciones.
- Detectar patrones o tendencias en el desempeño de equipos.

Permisos:

- Acceso completo a visualizaciones y consultas
- Filtros por fechas, equipos, ligas

3. Tester o Evaluador

Descripción:

Usuario encargado de validar el correcto funcionamiento del sistema en cuanto a rendimiento, integridad de los datos y experiencia de usuario.

Responsabilidades:

- Validar que los datos del API se integren correctamente.
- Verificar que las visualizaciones estén bien renderizadas.
- Reportar errores o bugs encontrados en el sistema.

Permisos:

- Acceso a todos los módulos funcionales

4. Usuario Final (Observador)

Descripción:

Persona que simplemente accede a la plataforma para visualizar resultados de manera rápida y sencilla, sin conocimientos técnicos.

Responsabilidades:

- Navegar por el sistema de visualización.
- Observar estadísticas clave de equipos o partidos.
- Obtener información para fines personales, académicos o recreativos.

Permisos:

- Solo lectura de datos y visualizaciones

5. Docente o Evaluador Académico

Descripción:

Profesor o tutor que revisa el avance del proyecto para evaluación académica.

Responsabilidades:

- Consultar documentación del proyecto.
- Probar funcionalidades clave del sistema.
- Verificar coherencia entre lo planeado y lo ejecutado.

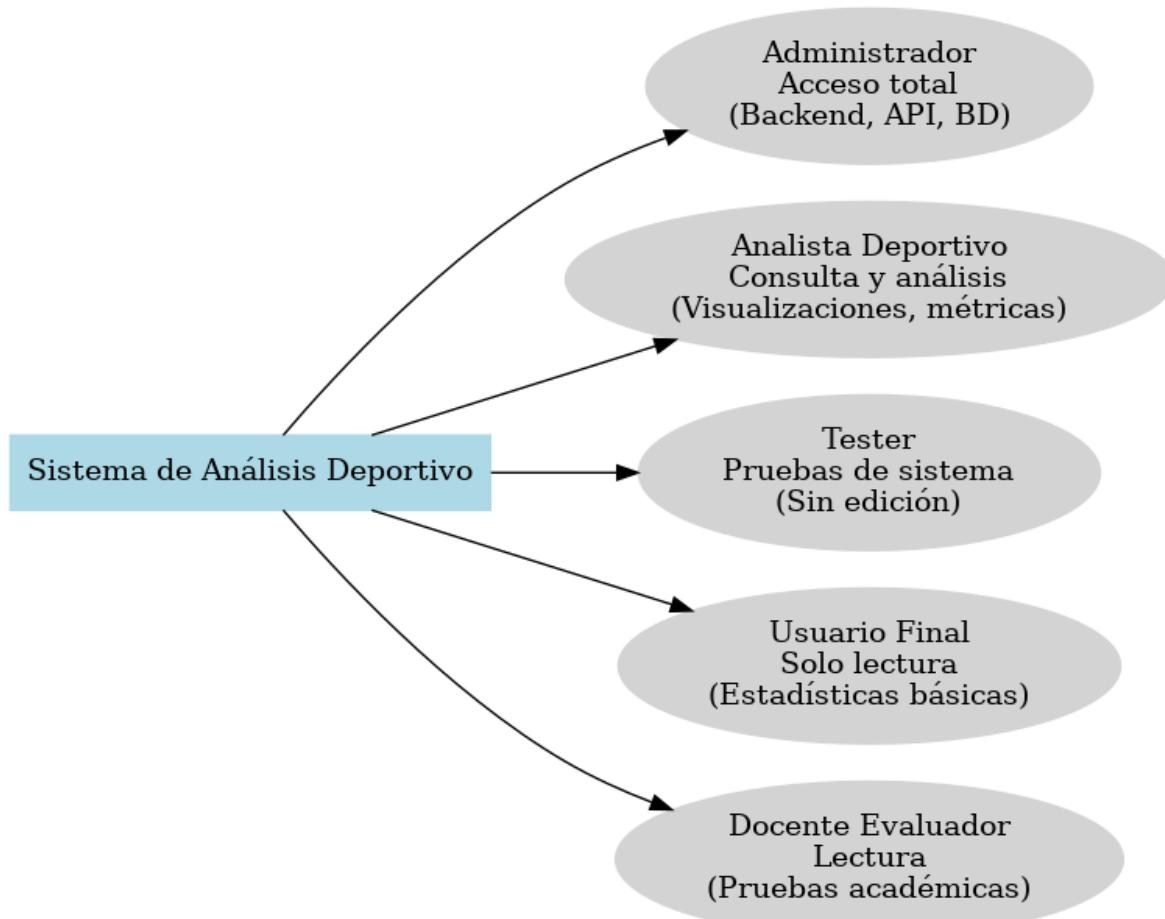
Permisos:

- Acceso a documentación, endpoints principales y dashboard

TABLA DE USUARIOS:

Tipo de Usuario	Descripción	Permisos Clave
Administrador del Sistema	Encargado de configurar, mantener y actualizar el sistema	Acceso completo, puede modificar datos y configuración.
Analista Deportivo	Analiza los datos y genera conclusiones estadísticas.	Acceso a visualizaciones, filtros y exportaciones.
Tester o Evaluador	Valida el correcto funcionamiento del sistema.	Acceso de prueba, sin modificar datos.
Usuario Final (Observador)	Accede a los resultados de forma pasiva.	Solo lectura básica de resultados.
Docente o Evaluador Académico	Evalúa el proyecto con fines académicos.	Lectura limitada, acceso a documentación y dashboard.

DIAGRAMA:



MODELO ENTIDAD RELACION:

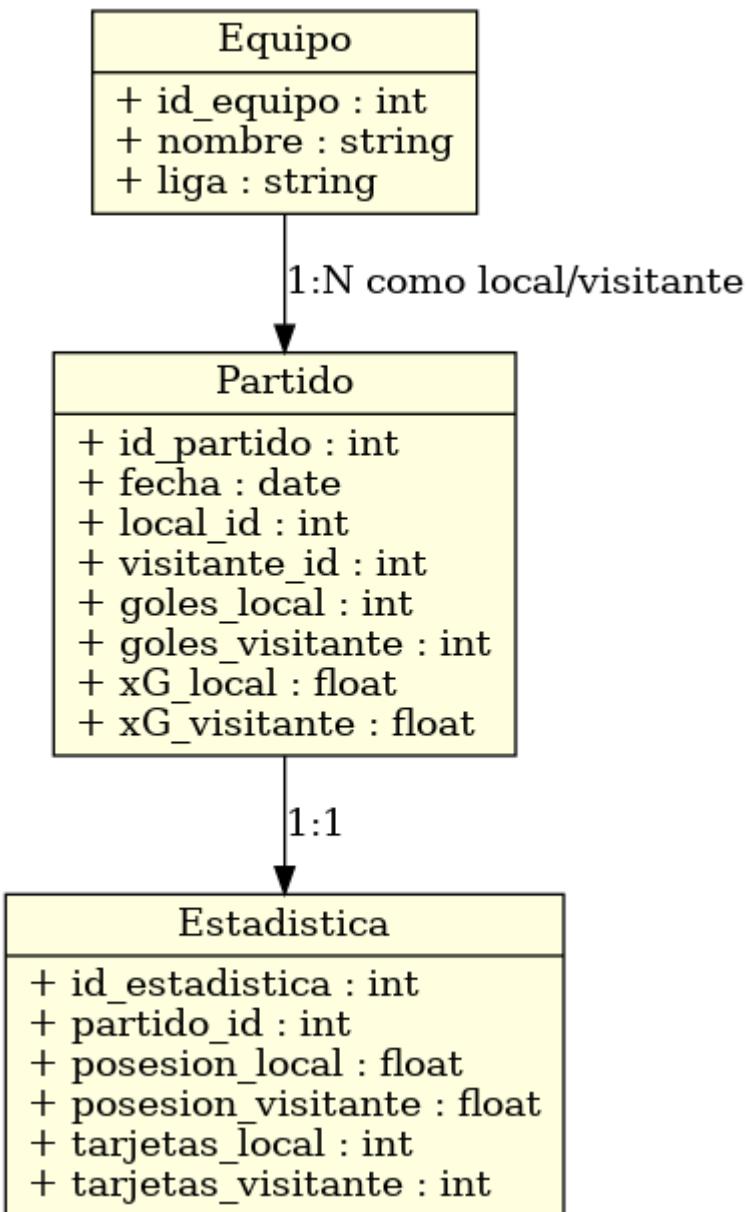
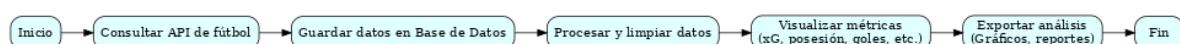


DIAGRAMA BPMN:



ERD normalizado:

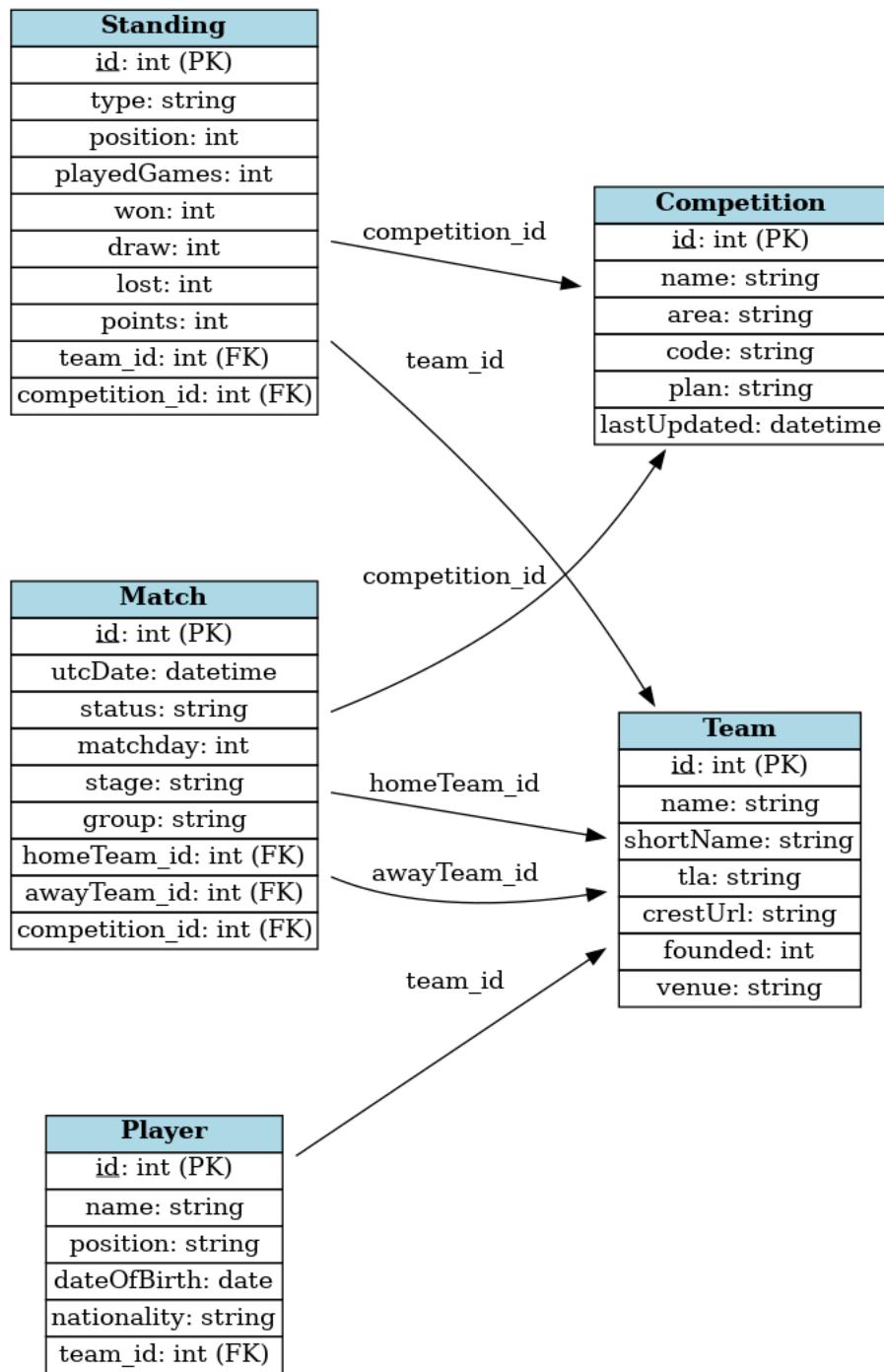


DIAGRAMA DE CLASES:

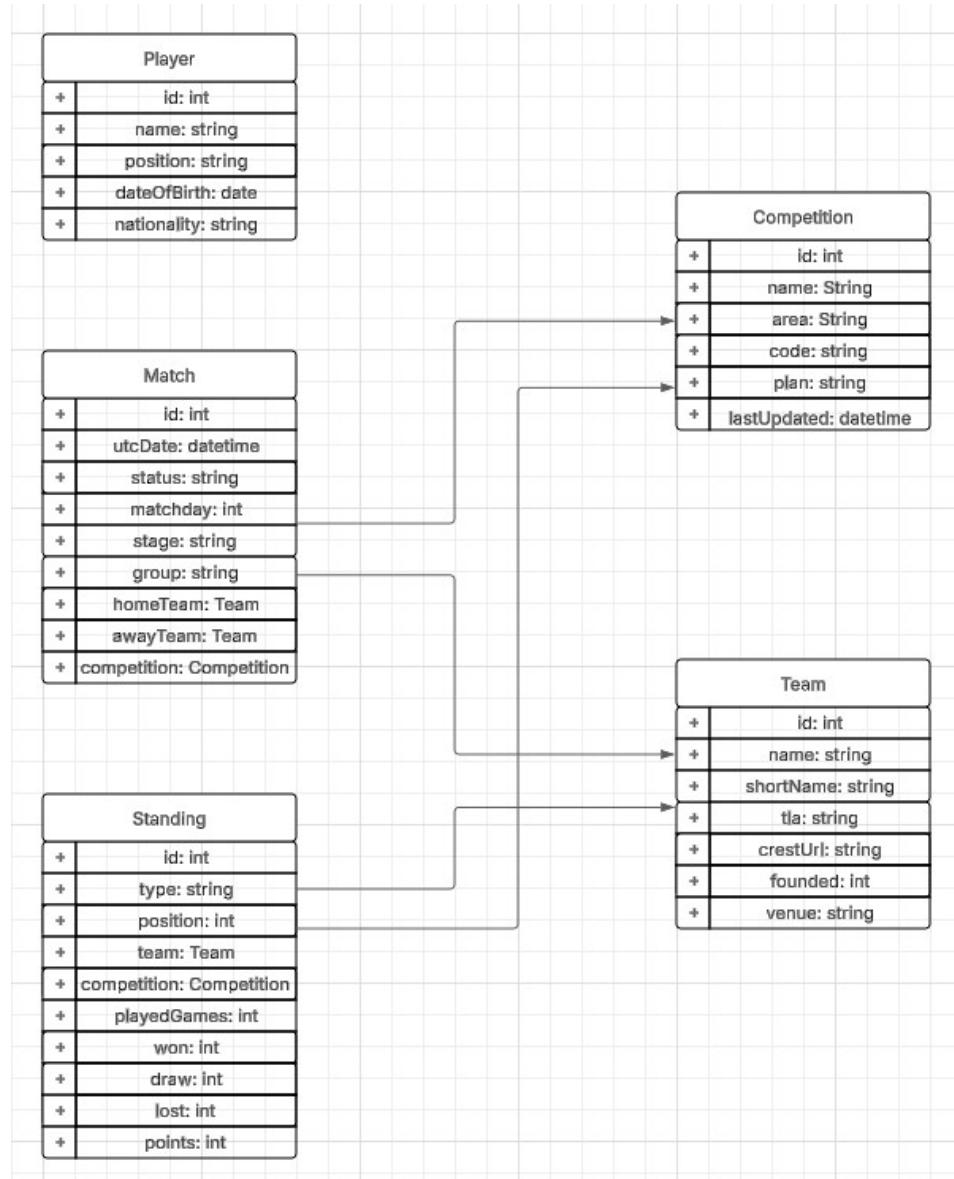
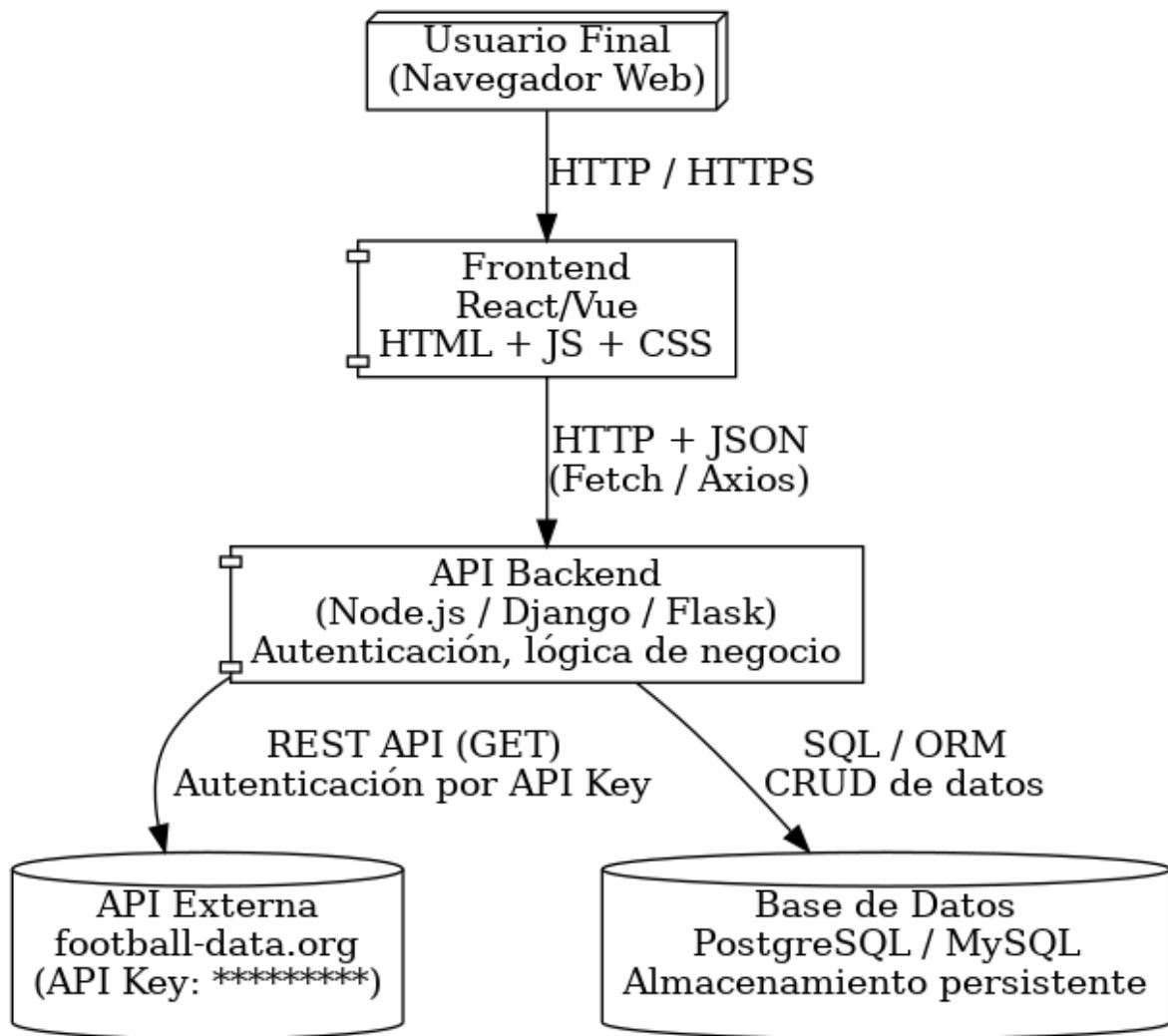


DIAGRAMA DE DESPLIEGUE:



Documentación inicial en Jira

Link: <https://pryectobdts8a.atlassian.net/jira/software/projects/BBDD3/issues/BBDD3-1?jql=project%20%3D%20%22BBDD3%22%20ORDER%20BY%20created%20ASC>

The screenshot shows the Jira interface for the project 'BBDD3'. The top navigation bar includes 'Proyectos', the project logo 'BBDD3', and a three-dot menu. Below the bar are links for 'Resumen', 'Cronograma', and 'B...'. A search bar contains the query 'project = "BBDD3"'. The main area displays a list of issues under the 'Creado' filter. Each issue card includes the title, a small icon, and a user profile picture.

Título	Asignado a
Planificación Inicial del Proyecto	BBDD3-1
Conexión con API externa	BBDD3-2
Modelado y Gestión de la Base de Datos	BBDD3-3
Desarrollo del Backend con Flask	BBDD3-4
Visualización de Datos	BBDD3-5
Despliegue en AWS	BBDD3-6
Pruebas y Validaciones	BBDD3-7

Commits en GitHub

Link: <https://github.com/matthewespinosa07/Proyecto-BBDD3/tree/main>

The screenshot shows a GitHub repository named "Proyecto-BBDD3" which is public. The main branch is "main". There is 1 branch and 0 tags. A search bar allows going to a file. Buttons for "Add file" and "Code" are present. A commit from user "matthewespinosa07" is listed, updating README.md 14 minutes ago. The commit details show the creation of several files: Documentos, README.md, app.py, db.py, and requirements.txt, all completed last week.

File	Action	Time
Documentos	Create Documentos	last week
README.md	Update README.md	14 minutes ago
app.py	Update app.py	last week
db.py	Create db.py	last week
requirements.txt	Create requirements.txt	last week

SPRINT 2: Implementacion Técnica y ETL (Semana 3 a 4)

1. Selección y prueba de API externa

Se eligió la API de football-data.org como fuente principal de datos deportivos.

Esta decisión se basó en dos razones fundamentales:

Preferencia personal por el dominio del fútbol

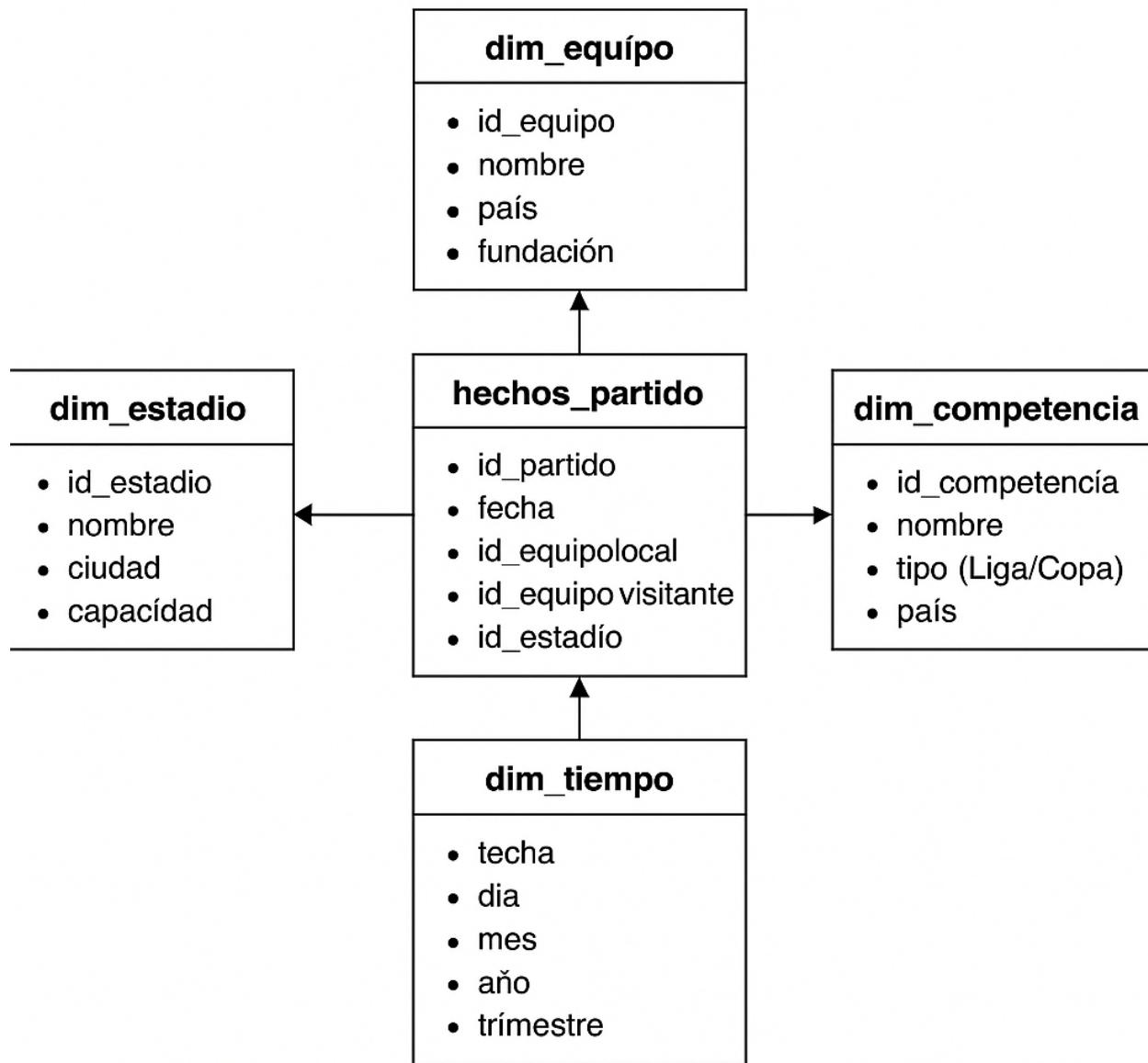
El equipo siente afinidad por este deporte, lo que facilita la comprensión del contexto de los datos (partidos, equipos, estadísticas).

Facilidad de manejo y curva de aprendizaje baja

Dado que ya se tenía un conocimiento previo del funcionamiento de esta API y del tipo de datos que ofrece, se redujo el tiempo de adaptación y prueba en comparación con otras APIs más complejas o con dominios menos familiares.

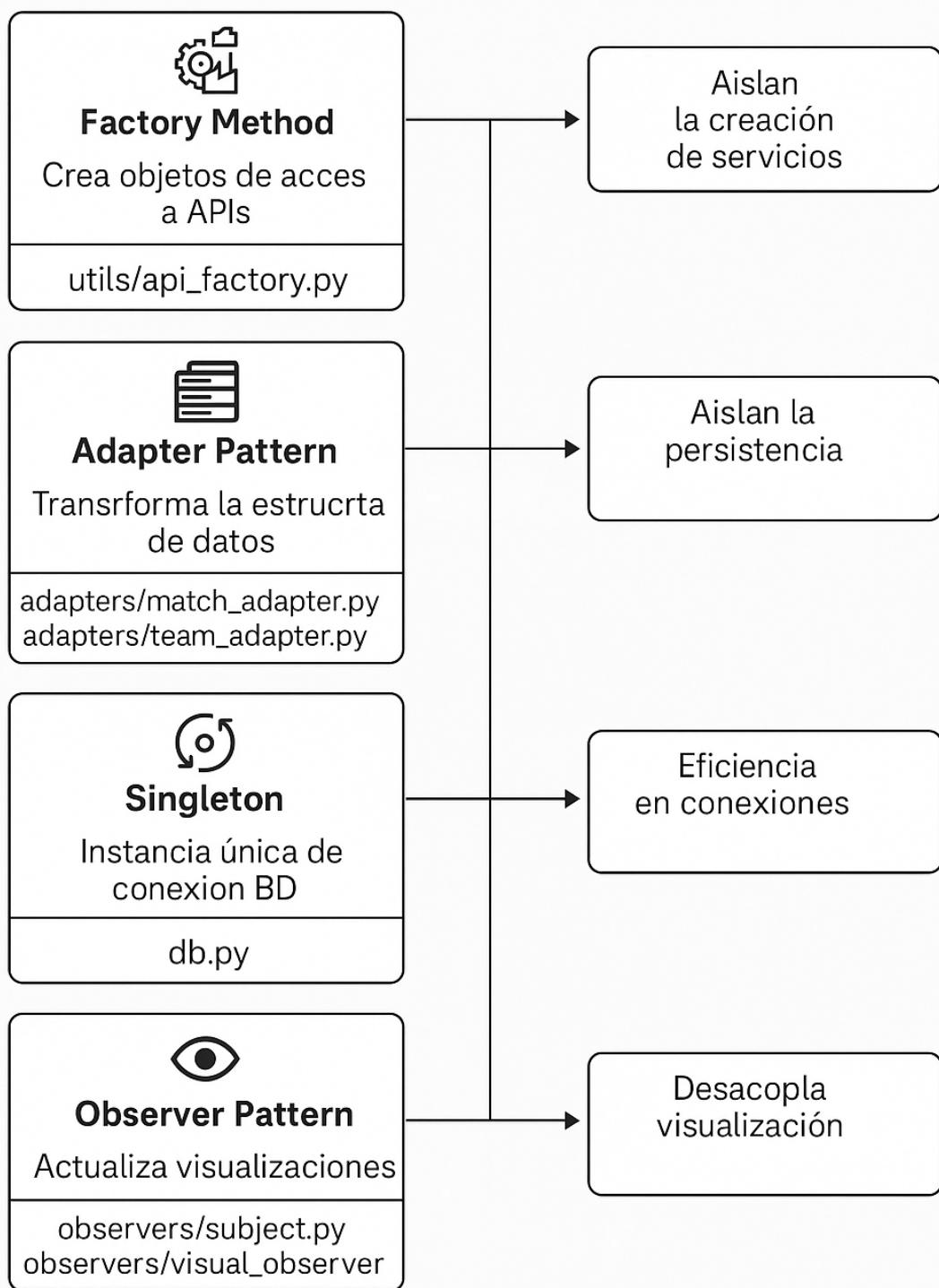
La API permitió obtener datos consistentes y bien estructurados, lo que facilitó el diseño del modelo dimensional y la integración con los patrones de diseño implementados.

2. Modelo dimensional en estrella



3. Aplicación patrones de diseño

Aplicación de Patrones de Diseño en el Proyecto



4. Interfaz HTML para mostrar partidos (Frontend)

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Listado de Partidos</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f0f4f8;
            margin: 0;
            padding: 20px;
        }

        h1 {
            text-align: center;
            color: #333;
        }

        table {
            margin: 0 auto;
            width: 90%;
            border-collapse: collapse;
            background: white;
            box-shadow: 0 4px 6px rgba(0,0,0,0.1);
        }

        th, td {
            padding: 12px;
            text-align: center;
            border-bottom: 1px solid #ddd;
        }

        th {
            background-color: #007BFF;
            color: white;
        }

        tr:hover {
            background-color: #f1f1f1;
        }
    </style>
</head>
<body>
    <h1>Listado de Partidos de Fútbol</h1>
    <table>
        <thead>
            <tr>
                <th>ID</th>
                <th>Fecha</th>
                <th>Equipo Local</th>
                <th>Equipo Visitante</th>
                <th>Goles Local</th>
                <th>Goles Visitante</th>
            </tr>
        <tbody>
        </tbody>
    </table>
</body>
```

```

        </tr>
    </thead>
    <tbody id="tabla-body"></tbody>
</table>

<script>
    fetch('/partidos')
        .then(response => response.json())
        .then(data => {
            const tbody = document.getElementById('tabla-body');
            if (data.length === 0) {
                tbody.innerHTML = '<tr><td colspan="6">No hay datos disponibles</td></tr>';
            } else {
                data.forEach(p => {
                    const fila = document.createElement('tr');
                    fila.innerHTML =
                        `<td>${p.id}</td>
                        <td>${p.fecha}</td>
                        <td>${p.equipo_local}</td>
                        <td>${p.equipo_visitante}</td>
                        <td>${p.goles_local}</td>
                        <td>${p.goles_visitante}</td>
                    `;
                    tbody.appendChild(fila);
                });
            }
        })
        .catch(err => {
            console.error(err);
            document.getElementById('tabla-body').innerHTML =
                '<tr><td colspan="6" style="color:red;">Error al cargar los datos</td></tr>';
        });
    </script>
</body>
</html>
```

		Equipo Local	Equipo Visitante	Goles Local	Goles Visita
497761	2025-05-10	Fulham FC	Everton FC	1	3
497762	2025-05-10	Ipswich Town FC	Brentford FC	0	1
497767	2025-05-10	Southampton FC	Manchester City FC	0	0
497769	2025-05-10	Wolverhampton Wanderers FC	Brighton & Hove Albion FC	0	2
497760	2025-05-10	AFC Bournemouth	Aston Villa FC	0	1
497765	2025-05-11	Newcastle United FC	Chelsea FC	2	0
497764	2025-05-11	Manchester United FC	West Ham United FC	0	2
497766	2025-05-11	Nottingham Forest FC	Leicester City FC	2	2
497768	2025-05-11	Tottenham Hotspur FC	Crystal Palace FC	0	2
497763	2025-05-11	Liverpool FC	Arsenal FC	2	2
497771	2025-05-16	Aston Villa FC	Tottenham Hotspur FC	2	0
497774	2025-05-16	Chelsea FC	Manchester United FC	1	0
497776	2025-05-18	Everton FC	Southampton FC	2	0
497779	2025-05-18	West Ham United FC	Nottingham Forest FC	1	2
497772	2025-05-18	Brentford FC	Fulham FC	2	3
497777	2025-05-18	Leicester City FC	Ipswich Town FC	2	0
497770	2025-05-18	Arsenal FC	Newcastle United FC	1	0
497773	2025-05-19	Brighton & Hove Albion FC	Liverpool FC	3	2
497775	2025-05-20	Crystal Palace FC	Wolverhampton Wanderers FC	4	2
497778	2025-05-20	Manchester City FC	AFC Bournemouth	3	1
497780	2025-05-25	AFC Bournemouth	Leicester City FC		

4. Backend

```

from flask import Flask, jsonify
import pandas as pd

app = Flask(__name__)

# Ruta para leer el CSV y devolver datos JSON
@app.route('/partidos')
def partidos():
    try:
        # Leer archivo CSV (asegúrate que la ruta sea correcta)
        df = pd.read_csv('partidos.csv')

        # Reemplazar NaN por None para evitar errores en JSON
        df = df.where(pd.notnull(df), None)

        # Convertir DataFrame a lista de diccionarios para jsonify
        datos = df.to_dict(orient='records')

        return jsonify(datos)

    except Exception as e:
        return jsonify({"error": str(e)}), 500

if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0', port=8000)

```

SPRINT 3: Analisis, Visualizacion y Despliegue (Semana 5 a 6)

1. Dashboard funcional

```
import pandas as pd
import streamlit as st
import matplotlib.pyplot as plt

# -----
# FUNCIONES
# -----


def cargar_datos():
    df = pd.read_csv('partidos.csv')

    # Convertir fecha a datetime
    df['fecha'] = pd.to_datetime(df['fecha'])

    # KPIs adicionales
    df['resultado'] = df.apply(lambda row: 'Victoria Local' if
    row['goles_local'] > row['goles_visitante']
                                else 'Victoria Visitante' if
    row['goles_local'] < row['goles_visitante']
                                else 'Empate', axis=1)

    return df


def mostrar_kpis(df):
    total_partidos = len(df)
    total_goles = df['goles_local'].sum() + df['goles_visitante'].sum()
    promedio_goles = round(total_goles / total_partidos, 2)

    st.metric("📊 Total de Partidos", total_partidos)
    st.metric("⚽ Goles Totales", total_goles)
    st.metric("✗ Promedio de Goles por Partido", promedio_goles)


def graficar_goles_por_equipo(df):
    goles_local = df.groupby('equipo_local')['goles_local'].sum()
    goles_visitante =
    df.groupby('equipo_visitante')['goles_visitante'].sum()

    goles_totales = goles_local.add(goles_visitante,
    fill_value=0).sort_values(ascending=False)

    fig, ax = plt.subplots(figsize=(10, 4))
    goles_totales.plot(kind='bar', ax=ax, color='skyblue')
```

```

        ax.set_title('Goles Totales por Equipo')
        ax.set_ylabel('Goles')
        st.pyplot(fig)

def grafico_resultados(df):
    resultado_counts = df['resultado'].value_counts()
    fig, ax = plt.subplots()
    ax.pie(resultado_counts, labels=resultado_counts.index,
    autopct='%.1f%%', startangle=90)
    ax.axis('equal')
    st.pyplot(fig)

# -----
# INTERFAZ STREAMLIT
# -----


def main():
    st.set_page_config(page_title="Dashboard de Fútbol", layout="wide")
    st.title("📊 Dashboard de Partidos de Fútbol")

    df = cargar_datos()

    # Mostrar tabla
    with st.expander("📋 Ver datos originales"):
        st.dataframe(df)

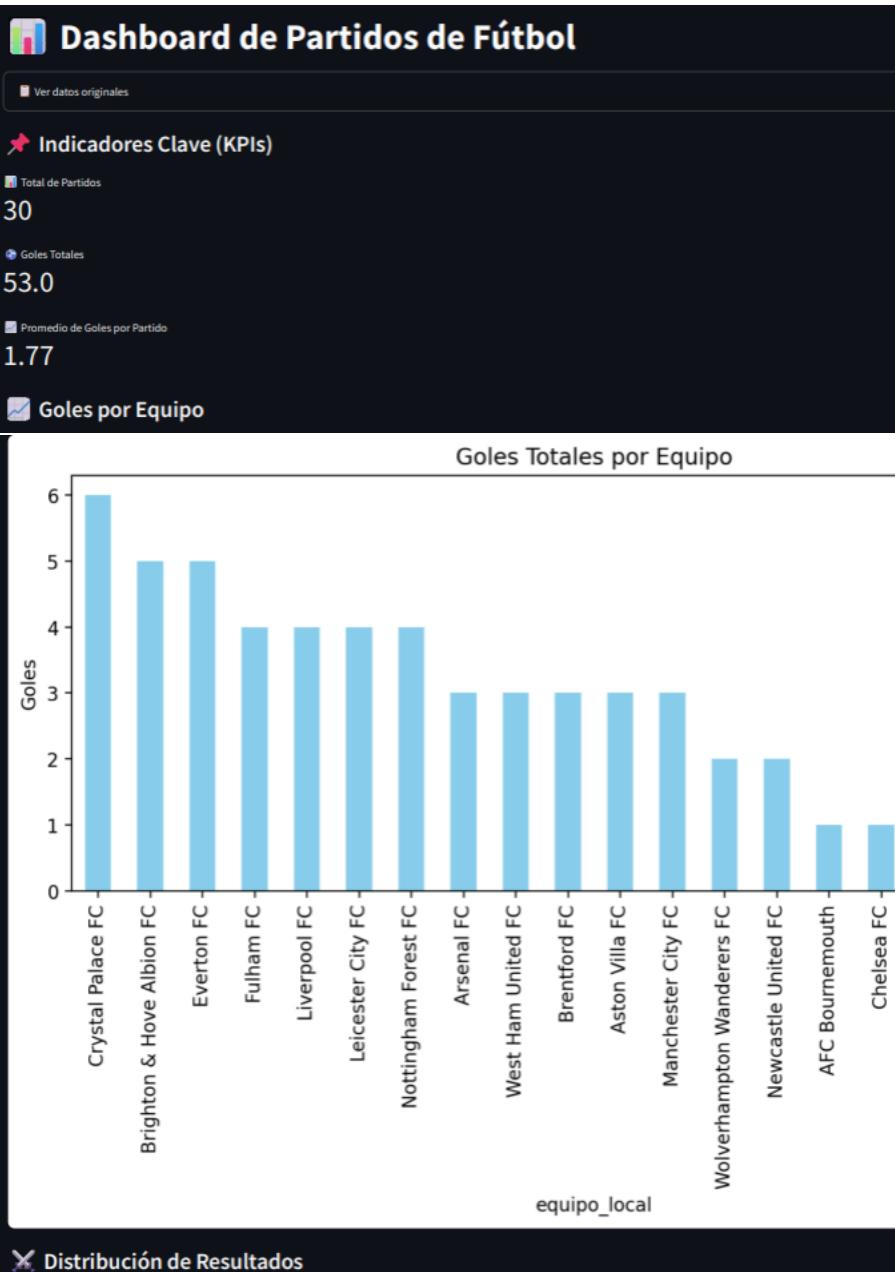
    # KPIs
    st.subheader("📌 Indicadores Clave (KPIs)")
    mostrar_kpis(df)

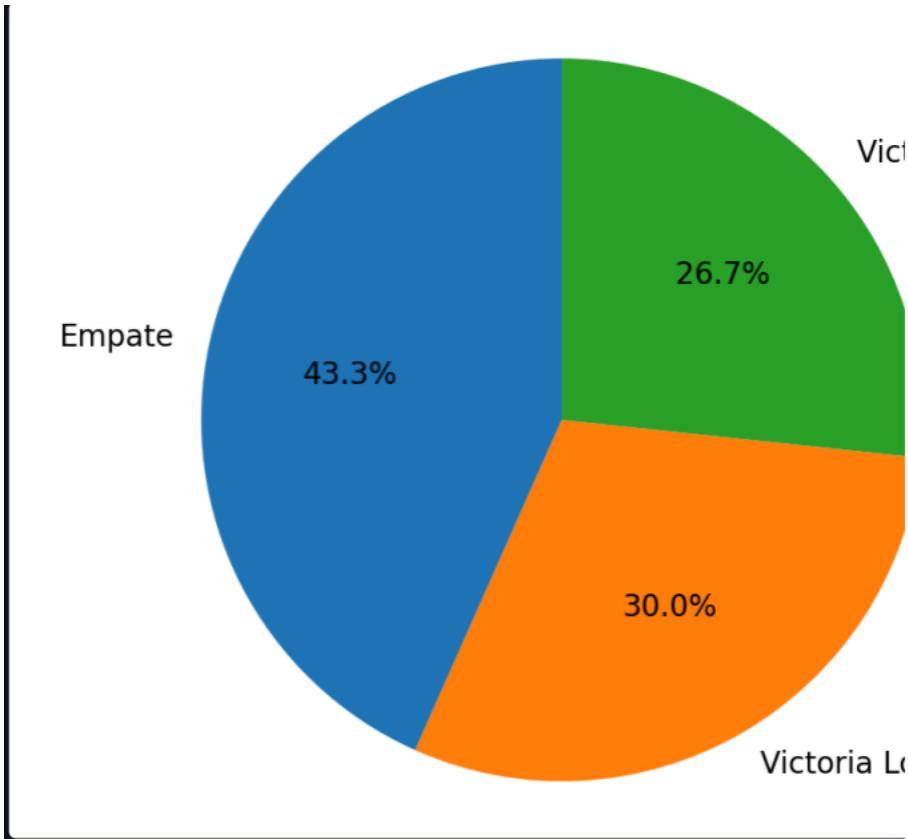
    # Gráficos
    st.subheader("📈 Goles por Equipo")
    graficar_goles_por_equipo(df)

    st.subheader("🔀 Distribución de Resultados")
    grafico_resultados(df)

if __name__ == "__main__":
    main()

```





2.Grafo con KPIs calculados y modelo predictivo

```
# 📦 Instalar dependencias necesarias
!apt-get update
!apt-get install -y graphviz libgraphviz-dev
!pip install pygraphviz
!pip install requests pandas matplotlib seaborn networkx scikit-learn
statsmodels

# 📈 Cargar datos desde la API de football-data.org
import requests
import pandas as pd
import networkx as nx # Import the networkx library
from networkx.drawing.nx_agraph import graphviz_layout # Import
graphviz_layout
import matplotlib.pyplot as plt # Add this line to import matplotlib

API_TOKEN = "9b6fb9d028c84f48a2362feb2210d0f9"
HEADERS = {'X-Auth-Token': API_TOKEN}
BASE_URL = "https://api.football-data.org/v4/"

def obtener_partidos(competicion='PL', temporada=2023, limite=100):
```

```

    url =
f"{{BASE_URL}}competitions/{{competicion}}/matches?season={{temporada}}"
    response = requests.get(url, headers=HEADERS)
    data = response.json()
    partidos = data['matches'][:limite]
    return pd.json_normalize(partidos)

df_partidos = obtener_partidos()
print("✅ Partidos descargados:", df_partidos.shape)
df_partidos[['utcDate', 'homeTeam.name', 'awayTeam.name',
'score.fullTime.home', 'score.fullTime.away']].head()

# 🔍 Modelo de regresión lineal (con datos simulados)
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
import statsmodels.api as sm
import seaborn as sns

df_modelo = df_partidos[[
    'score.fullTime.home', 'score.fullTime.away',
    'homeTeam.name', 'awayTeam.name'
]].copy()

np.random.seed(42)
df_modelo['HS'] = np.random.randint(5, 25, size=len(df_modelo))
df_modelo['AS'] = np.random.randint(5, 25, size=len(df_modelo))

df_modelo['Goles'] = df_modelo['score.fullTime.home'] +
df_modelo['score.fullTime.away']
df_modelo['Disparos'] = df_modelo['HS'] + df_modelo['AS']
df_modelo = df_modelo[['Goles', 'Disparos']].dropna()

print("\n✅ Datos del modelo:", df_modelo.shape)
print(df_modelo.head())

# 📈 Visualizar relación goles/disparos
sns.set(style="whitegrid")
sns.scatterplot(x="Disparos", y="Goles", data=df_modelo)
plt.title("Relación entre disparos y goles")
plt.xlabel("Disparos")
plt.ylabel("Goles")
plt.show()

# 📈 Entrenar modelo

```

```

X = df_modelo[["Disparos"]]
y = df_modelo["Goles"]
modelo = LinearRegression()
modelo.fit(X, y)
y_pred = modelo.predict(X)

print(f"\n📊 Coeficiente (pendiente): {modelo.coef_[0]:.4f}")
print(f"📈 Intercepto: {modelo.intercept_:.4f}")
print(f"📉 R² Score: {r2_score(y, y_pred):.4f}")

# 📈 Statsmodels
X_sm = sm.add_constant(X)
modelo_sm = sm.OLS(y, X_sm).fit()
print(modelo_sm.summary())

# 📈 Gráfico con línea de regresión
plt.figure(figsize=(8, 5))
sns.scatterplot(x="Disparos", y="Goles", data=df_modelo, label="Datos reales")
plt.plot(df_modelo["Disparos"], y_pred, color="red", label="Regresión lineal")
plt.title("Regresión lineal: Goles vs Disparos")
plt.xlabel("Disparos")
plt.ylabel("Goles")
plt.legend()
plt.show()

# 🏴 Grafo completo de la Premier League (todos los equipos conectados)
def construir_grafo_completo(df):
    G = nx.DiGraph()

    for _, row in df.iterrows():
        local = row['homeTeam.name']
        visitante = row['awayTeam.name']
        goles_local = row['score.fullTime.home']
        goles_visitante = row['score.fullTime.away']

        if pd.isna(goles_local) or pd.isna(goles_visitante):
            continue

        if goles_local > goles_visitante:
            G.add_edge(local, visitante, weight=3)
        elif goles_local < goles_visitante:
            G.add_edge(visitante, local, weight=3)
        else:
            G.add_edge(local, visitante, weight=1)
            G.add_edge(visitante, local, weight=1)

```

```

        G.add_edge(local, visitante, weight=1)
        G.add_edge(visitante, local, weight=1)

    return G

# Construir y graficar
G_completo = construir_grafo_completo(df_partidos)

plt.figure(figsize=(15, 12))
pos = graphviz_layout(G_completo, prog='dot')

nx.draw(
    G_completo, pos, with_labels=True,
    node_color='skyblue', node_size=1600,
    edge_color='gray', arrows=True, font_size=9, font_weight='bold'
)
edge_labels = nx.get_edge_attributes(G_completo, 'weight')
nx.draw_networkx_edge_labels(G_completo, pos, edge_labels=edge_labels,
font_color='darkred')

plt.title("∅ Grafo completo de partidos de la Premier League",
fontsize=16)
plt.axis('off')
plt.show()

# 🏆 Grafo reducido con los 7 mejores equipos
def construir_grafo_top7(df):
    G = nx.DiGraph()
    puntos = {}

    # Calcular puntos por equipo
    for _, row in df.iterrows():
        local = row['homeTeam.name']
        visitante = row['awayTeam.name']
        goles_local = row['score.fullTime.home']
        goles_visitante = row['score.fullTime.away']

        if pd.isna(goles_local) or pd.isna(goles_visitante):
            continue

        if goles_local > goles_visitante:
            puntos[local] = puntos.get(local, 0) + 3
        elif goles_local < goles_visitante:
            puntos[visitante] = puntos.get(visitante, 0) + 3
        else:
            puntos[local] = puntos.get(local, 0) + 1
            puntos[visitante] = puntos.get(visitante, 0) + 1

    return G, puntos

```

```

        puntos[local] = puntos.get(local, 0) + 1
        puntos[visitante] = puntos.get(visitante, 0) + 1

    # Seleccionar los 7 mejores
    top7 = sorted(puntos.items(), key=lambda x: x[1], reverse=True)[:7]
    equipos_top7 = set(e[0] for e in top7)

    # Construir grafo solo con partidos entre ellos
    for _, row in df.iterrows():
        local = row['homeTeam.name']
        visitante = row['awayTeam.name']
        goles_local = row['score.fullTime.home']
        goles_visitante = row['score.fullTime.away']

        if (local not in equipos_top7) or (visitante not in
equipo_top7):
            continue
        if pd.isna(goles_local) or pd.isna(goles_visitante):
            continue

        if goles_local > goles_visitante:
            G.add_edge(local, visitante, weight=3)
        elif goles_local < goles_visitante:
            G.add_edge(visitante, local, weight=3)
        else:
            G.add_edge(local, visitante, weight=1)
            G.add_edge(visitante, local, weight=1)

    return G, top7

# Crear grafo
G_top7, top7_lista = construir_grafo_top7(df_partidos)

# Mostrar los 7 mejores equipos
print("\n🏆 Top 7 equipos por puntos:")
for equipo, puntos in top7_lista:
    print(f"{equipo}: {puntos} puntos")

# 🎨 Dibujar grafo de los 7 mejores
plt.figure(figsize=(12, 10))
pos = graphviz_layout(G_top7, prog='dot')

nx.draw(
    G_top7, pos, with_labels=True,
    node_color='gold', node_size=1600,

```

```

        edge_color='gray', arrows=True, font_size=10, font_weight='bold'
    )
edge_labels = nx.get_edge_attributes(G_top7, 'weight')
nx.draw_networkx_edge_labels(G_top7, pos, edge_labels=edge_labels,
font_color='black')

plt.title("↑ Grafo entre los 7 mejores equipos de la Premier League",
fontsize=16)
plt.axis('off')
plt.show()

# 🏴 Grafo de Liverpool con conexiones directas (más limpio y tipo
estrella)
def construir_grafo_estilo_estrella(df, equipo):
    G = nx.DiGraph()
    df_filtrado = df[(df['homeTeam.name'] == equipo) |
(df['awayTeam.name'] == equipo)]

    for _, row in df_filtrado.iterrows():
        local = row['homeTeam.name']
        visitante = row['awayTeam.name']
        goles_local = row['score.fullTime.home']
        goles_visitante = row['score.fullTime.away']

        if pd.isna(goles_local) or pd.isna(goles_visitante):
            continue

        if equipo in (local, visitante):
            rival = visitante if local == equipo else local

            if goles_local > goles_visitante and local == equipo:
                G.add_edge(equipo, rival, weight=3)
            elif goles_visitante > goles_local and visitante == equipo:
                G.add_edge(equipo, rival, weight=3)
            elif goles_local < goles_visitante and local == equipo:
                G.add_edge(rival, equipo, weight=3)
            elif goles_visitante < goles_local and visitante == equipo:
                G.add_edge(rival, equipo, weight=3)
            else:
                G.add_edge(equipo, rival, weight=1)
                G.add_edge(rival, equipo, weight=1)

    return G

# Define the team name to focus on

```

```

equipo_focal = "Liverpool FC"
G_limpio = construir_grafo_estilo_estrella(df_partidos, equipo_focal)

print("\n")

# 🖼 Dibujar grafo estilo estrella
plt.figure(figsize=(10, 8))
pos = nx.spring_layout(G_limpio, k=0.8, seed=42) # Distribución limpia
tipo red

nx.draw(
    G_limpio, pos, with_labels=True,
    node_color='lightgreen', node_size=1500,
    edge_color='gray', arrows=True, font_size=10, font_weight='bold'
)
edge_labels = nx.get_edge_attributes(G_limpio, 'weight')
nx.draw_networkx_edge_labels(G_limpio, pos, edge_labels=edge_labels,
font_color='red')

plt.title(f"Conexiones directas de {equipo_focal} con sus rivales",
fontsize=14)
plt.axis('off')
plt.show()

# 🔍 Layout jerárquico ordenado (tipo estructura matemática)
plt.figure(figsize=(12, 10))
# Use graphviz_layout from networkx
# Ensure G_liverpool is defined - it seems G_limpio is the correct graph
# pos = graphviz_layout(G_liverpool, prog='dot')
pos = graphviz_layout(G_limpio, prog='dot')


nx.draw(
    # Ensure G_liverpool is defined - it seems G_limpio is the correct
graph
    # G_liverpool, pos, with_labels=True,
    G_limpio, pos, with_labels=True,
    node_size=1200, node_color='red', font_size=9,
    edge_color='gray', arrows=True, font_weight='bold'
)
edge_labels = nx.get_edge_attributes(G_limpio, 'weight') # Use G_limpio
nx.draw_networkx_edge_labels(G_limpio, pos, edge_labels=edge_labels,
font_color='red') # Use G_limpio

print("\n")

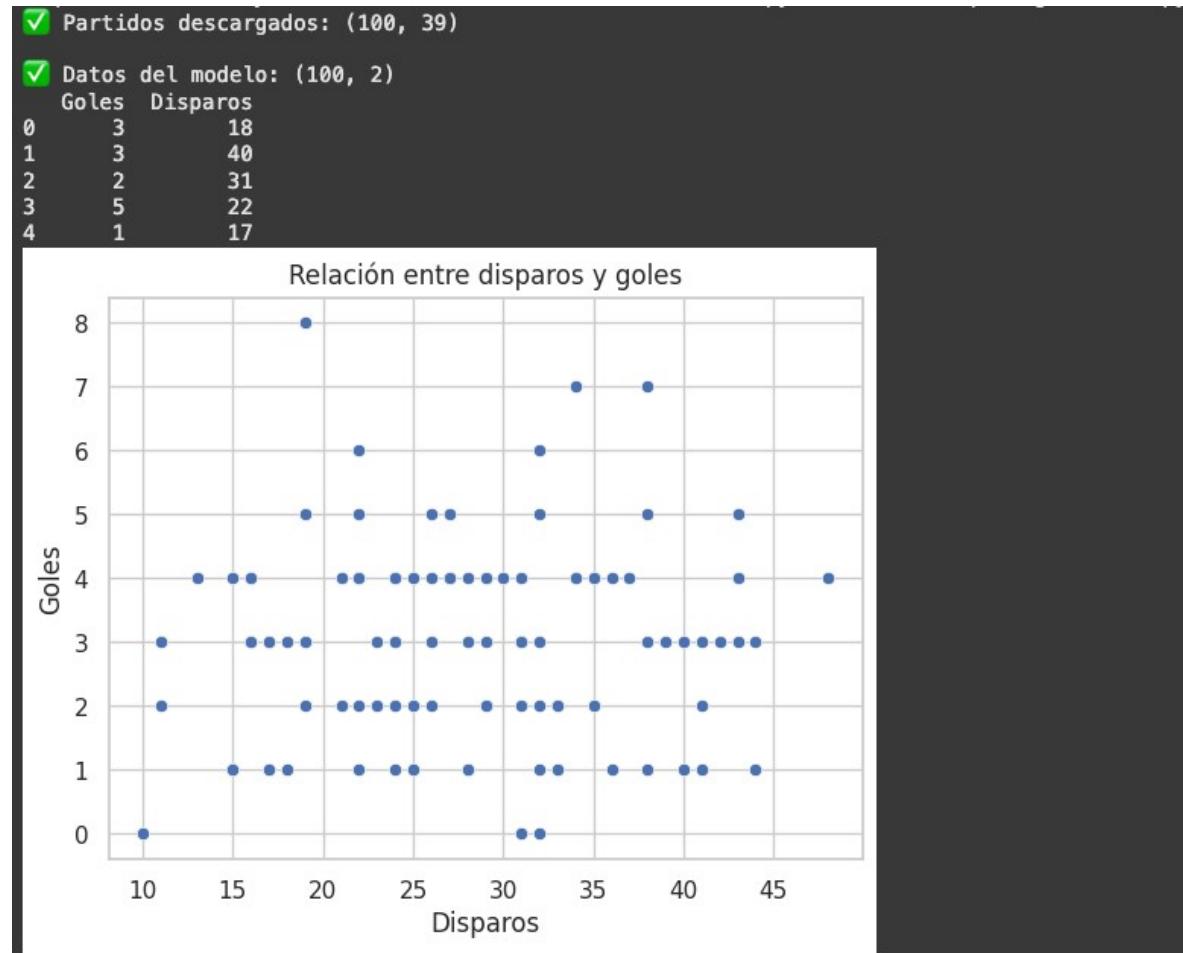
```

```

plt.title(f"Partidos del {equipo_focal}", fontsize=14)
plt.axis('off')
plt.show()

# Mostrar tabla de puntos del Liverpool y sus rivales
# Ensure G_liverpool is defined - it seems G_limpio is the correct graph
# kpis_liverpool =
pd.DataFrame.from_dict(dict(G_liverpool.in_degree(weight='weight')),
orient='index', columns=['Puntos recibidos'])
kpis_equipo_focal =
pd.DataFrame.from_dict(dict(G_limpio.in_degree(weight='weight')),
orient='index', columns=['Puntos recibidos'])
print(f"\n❸ KPI de puntos (solo {equipo_focal} y sus rivales):\n") # Use
equipo_focal
# print(kpis_liverpool.sort_values(by='Puntos recibidos',
ascending=False))
print(kpis_equipo_focal.sort_values(by='Puntos recibidos',
ascending=False))

```



 Coeficiente (pendiente): 0.0071
 Intercepto: 2.8330
 R² Score: 0.0016

OLS Regression Results

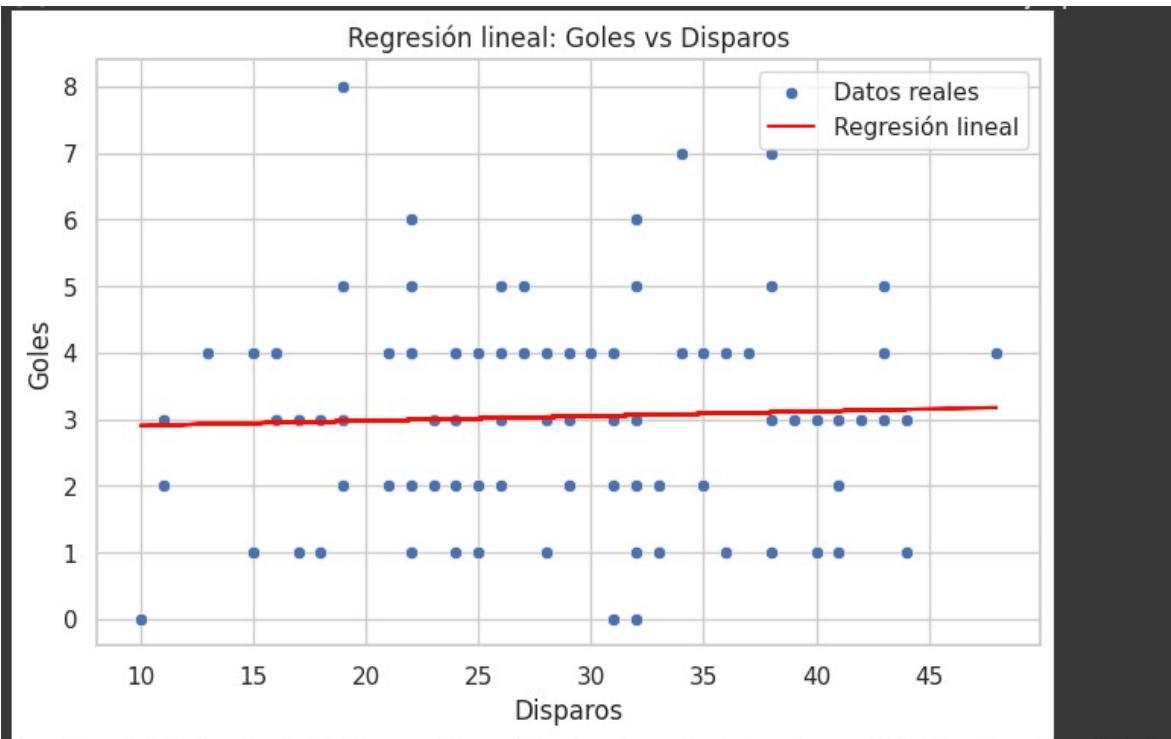
Dep. Variable:	Goles	R-squared:	0.002
Model:	OLS	Adj. R-squared:	-0.009
Method:	Least Squares	F-statistic:	0.1572
Date:	Mon, 26 May 2025	Prob (F-statistic):	0.693
Time:	02:50:33	Log-Likelihood:	-185.36
No. Observations:	100	AIC:	374.7
Df Residuals:	98	BIC:	379.9
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P> t	[0.025	0.975]
const	2.8330	0.521	5.440	0.000	1.800	3.866
Disparos	0.0071	0.018	0.396	0.693	-0.028	0.043

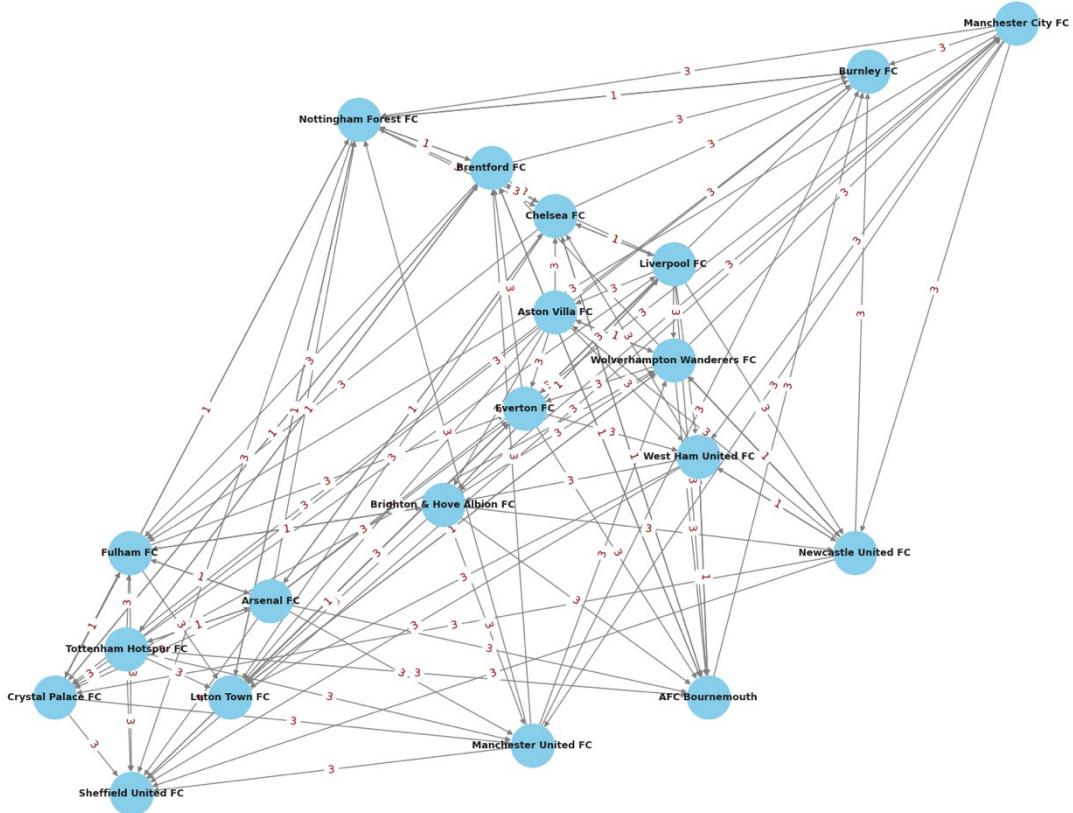
Omnibus:	3.865	Durbin-Watson:	1.794
Prob(Omnibus):	0.145	Jarque-Bera (JB):	3.186
Skew:	0.388	Prob(JB):	0.203
Kurtosis:	3.403	Cond. No.	97.1

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.



□ Grafo completo de partidos de la Premier League



🏆 Top 7 equipos por puntos:

Tottenham Hotspur FC: 26 puntos
Manchester City FC: 24 puntos
Arsenal FC: 24 puntos
Liverpool FC: 23 puntos
Aston Villa FC: 22 puntos
Brighton & Hove Albion FC: 17 puntos
Newcastle United FC: 17 puntos