



UNIVERSIDAD DE BURGOS  
ESCUELA POLITÉCNICA SUPERIOR  
Grado en Ingeniería Informática



**TFG del Grado en Ingeniería  
Informática**

**"FutboStats", análisis y  
estadísticas de fútbol**



Presentado por Miguel Ángel Extremo  
Cabornero  
en Universidad de Burgos — 11 de junio  
de 2024  
Tutor: César Represa Pérez





UNIVERSIDAD DE BURGOS  
ESCUELA POLITÉCNICA SUPERIOR  
Grado en Ingeniería Informática



D. César Represa Pérez, profesor del departamento de Ingeniería Electromecánica, área de Tecnología Electrónica.

Expone:

Que el alumno D. Miguel Ángel Extremo Cabornero, con DNI 71483511V, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado "FutboStats", análisis y estadísticas de fútbol.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 11 de junio de 2024

Vº. Bº. del Tutor:

César Represa Pérez



## Resumen

*FutboStats* es una aplicación web diseñada especialmente para los aficionados al fútbol y a las estadísticas. Esta plataforma interactiva permite a los usuarios buscar y visualizar datos de partidos, ligas y clasificaciones, ofreciendo además la capacidad de filtrar esta información según los parámetros que el usuario defina.

Un aspecto innovador de *FutboStats* es la integración de algoritmos avanzados que analizan tendencias en equipos o ligas. Por ejemplo, se puede calcular la probabilidad de que un disparo resulte en gol según su ubicación en el campo. Para ello, la aplicación emplea diversos modelos de datos que optimizan el filtrado y representación de la información, facilitando la creación de gráficos claros y fáciles de interpretar.

Además, *FutboStats* incorpora un sistema de seguimiento basado en inteligencia artificial, que permite analizar vídeos de el movimiento de los jugadores en el campo. Esta herramienta proporciona una perspectiva valiosa y detallada del comportamiento y la estrategia de juego, enriqueciendo la experiencia analítica del usuario.

## Descriptores

Aplicación web, estadísticas, fútbol, equipos, ligas, partidos, goles esperados, predicciones, representaciones gráficas, tracking.

## Abstract

*FutboStats* is a web application designed especially for football and statistics enthusiasts. This interactive platform allows users to search and visualize data from matches, leagues, and rankings, also offering the capability to filter this information based on user-defined parameters.

A novel aspect of *FutboStats* is the integration of advanced algorithms that analyze trends in teams or leagues. For example, it is possible to calculate the probability of a shot resulting in a goal based on its location on the field. For this purpose, the application uses various data models that optimize the filtering and representation of information, facilitating the creation of clear and easy-to-interpret graphics. Furthermore, *FutboStats* incorporates an artificial intelligence-based tracking system that allows for the real-time visualization of player movements on the field. This tool provides a valuable and detailed perspective on player behavior and game strategy, enhancing the user's analytical experience.

## Keywords

Web application, statistics, football, teams, leagues, matches, expected goals, predictions, graphics, tracking.

---

# Índice general

---

Índice general	iii
Índice de figuras	v
Índice de tablas	vii
<b>1. Introducción</b>	<b>1</b>
1.1. Estructura de la memoria . . . . .	2
1.2. Materiales adjuntos entregados . . . . .	3
<b>2. Objetivos del proyecto</b>	<b>4</b>
2.1. Objetivos generales . . . . .	4
2.2. Objetivos técnicos . . . . .	4
2.3. Objetivos personales . . . . .	5
<b>3. Conceptos teóricos</b>	<b>6</b>
3.1. API REST . . . . .	6
3.2. Proveedores de información . . . . .	7
3.3. Goles esperados (xG) . . . . .	9
3.4. Puntos esperados (xPoints) . . . . .	12
3.5. ImageAI . . . . .	15
<b>4. Técnicas y herramientas</b>	<b>17</b>
4.1. Metodologías . . . . .	17
4.2. Control de versiones . . . . .	17
4.3. Hosting del repositorio . . . . .	18
4.4. Entorno de desarrollo . . . . .	18

## *ÍNDICE GENERAL*

IV

4.5. Desarrollo front-end . . . . .	18
4.6. Desarrollo back-end . . . . .	19
4.7. Gestión de usuarios con Google . . . . .	20
4.8. Documentación . . . . .	20
4.9. Librerías . . . . .	21
<b>5. Aspectos relevantes del desarrollo del proyecto</b>	<b>31</b>
5.1. Inicio del proyecto . . . . .	31
5.2. Metodologías . . . . .	32
5.3. Formación . . . . .	34
5.4. Desarrollo de la aplicación . . . . .	35
5.5. Despliegue de la aplicación . . . . .	47
5.6. Testing . . . . .	49
<b>6. Trabajos relacionados</b>	<b>52</b>
<b>7. Conclusiones y Líneas de trabajo futuras</b>	<b>54</b>
<b>Bibliografía</b>	<b>56</b>

---

# Índice de figuras

---

3.1. Ejemplo de respuesta JSON StatsBombPy en GitHub . . . . .	8
3.2. Filtrado básico de datos con Python utilizando StatsBombPy . . . . .	8
3.3. Representación del modelo de goles esperados (xG) . . . . .	11
4.1. Logo de Angular . . . . .	21
4.2. Logo de Flaticon . . . . .	22
4.3. Logo de Animate.css . . . . .	23
4.4. Logo de Moment.js . . . . .	23
4.5. Logo de SweetAlert2 . . . . .	23
4.6. Logo de Angular Material . . . . .	24
4.7. Logo de Flask . . . . .	24
4.8. Logo de StatsBomb . . . . .	25
4.9. Logo de MplSoccer . . . . .	25
4.10. Logo de Pandas . . . . .	26
4.11. Logo de Matplotlib . . . . .	26
4.12. Logo de NumPy . . . . .	27
4.13. Logo de Scipy . . . . .	27
4.14. Logo de Scikit-learn . . . . .	27
4.15. Logo de ImageAI . . . . .	28
4.16. Logo de Ffmpeg . . . . .	28
4.17. Logo de JustInMind . . . . .	29
4.18. Logo de Postman . . . . .	29
5.1. Logo de FutboStats . . . . .	32
5.2. Tablero Kanban utilizado en el desarrollo del proyecto. . . . .	33
5.3. Prototipo de la vista de inicio desarrollado en JustInMind . . . . .	36
5.4. Sugerencias al usuario en la vista 'Competiciones' . . . . .	37
5.5. Configuración de rutas en Google Cloud . . . . .	39
5.6. Ejemplo de búsqueda de partidos del Burgos CF . . . . .	40

5.7. Adaptación a móvil de FutboStats . . . . .	40
5.8. Código empleado para solucionar el problema CORS . . . . .	42
5.9. Ejemplo de mapa de calor generado en Python. . . . .	43
5.10. Ejemplo de mapa de posiciones de un jugador generado en Python. . . . .	43
5.11. Ejemplo de mapa de pases de un jugador generado en Python. . . . .	44
5.12. Ejemplo de gráfico <i>match momentum</i> generado en Python. . . . .	45
5.13. Ejemplo de gráfico de una portería que representa los disparos que tuvo Argentina en el mundial 2022. . . . .	45
5.14. Arquitectura de la aplicación en local . . . . .	47
5.15. FutboStats desplegada en Netlify y Render . . . . .	48
5.16. Arquitectura de la aplicación en producción. . . . .	48
5.17. Ejemplo de petición a API-FOOTBALL . . . . .	50
5.18. Ejemplo de petición a la API de Flask . . . . .	51

---

## Índice de tablas

---

3.1.	Clasificación predicha para la temporada 2015/16 de la liga española empleando la métrica de goles esperados. . . . .	14
3.2.	Clasificación real para la temporada 2015/16 de la liga española. . . . .	15
3.3.	Comparación de modelos de detección de objetos . . . . .	16
4.1.	Resumen de herramientas y tecnologías utilizadas en cada parte del proyecto . . . . .	30
6.1.	Comparativa de funcionalidades entre el TFG de la Universidad de Sevilla, One Football y FutboStats . . . . .	53

---

# 1. Introducción

---

En la actualidad, el deporte ha sido uno de los campos que más ha aprovechado las innovaciones digitales para transformar cómo se analizan y se entienden los partidos o juegos. La integración de la tecnología en el deporte no solo ha mejorado el rendimiento de los atletas y equipos, sino que también ha revolucionado la experiencia de los aficionados y profesionales del sector. Este Trabajo de Fin de Grado presenta el desarrollo de una aplicación web deportiva centrada en el fútbol, que busca servir como herramienta para el análisis estadístico y la predicción de resultados en este deporte.

La aplicación aporta a los usuarios la posibilidad de ver estadísticas detalladas y buscar partidos según diferentes criterios, ofreciendo detalles sobre los equipos y sus posiciones en distintas ligas. Al emplear una API externa, la aplicación proporciona información actualizada y exacta que resulta imprescindible para cualquier estudio deportivo. También cuenta con una parte de predicciones que, usando modelos matemáticos, calcula los goles esperados de cada equipo, ofreciendo predicciones sobre los puntos que podrían lograr al final de la temporada.

La métrica innovadora de los goles esperados evalúa la posibilidad de que una oportunidad de gol se convierta en gol considerando factores como la posición del disparo y la formación defensiva. Este número no solo predice el rendimiento futuro, sino que también proporciona una perspectiva más completa del juego, lo que permite a entrenadores y analistas entender mejor las tácticas y el desempeño de los equipos.

La aplicación implementa como característica novedosa el uso de *ImageAI* (biblioteca de inteligencia artificial que permite realizar detección de objetos en imágenes y vídeos) para el análisis de vídeos de partidos de fútbol. Mediante técnicas de inteligencia artificial, se realiza un seguimiento detallado de los movimientos de los jugadores en el campo, proporcionando visualizaciones que mejoran el análisis táctico y la comprensión de las dinámicas del juego. Esta tecnología no solo enriquece la estrategia y la preparación de los equipos, sino que también mejora la narrativa para los seguidores del deporte, permitiendo una apreciación más rica y detallada de las habilidades y estrategias en juego.

En conclusión, esta aplicación no solo es un reflejo de cómo la tecnología puede ser aplicada para enriquecer el análisis deportivo, sino que también es una herramienta que podría influir significativamente en la toma de decisiones en el fútbol profesional. Al combinar estadísticas avanzadas, predicciones precisas y análisis visual de movimientos, puede llevarnos a una nueva forma de entender el fútbol, marcando un punto de inflexión para entrenadores, jugadores, analistas y, en última instancia, para los aficionados.

## 1.1. Estructura de la memoria

La memoria sigue la siguiente estructura:

- **Introducción:** Breve descripción de los contenidos del proyecto y de la aplicación sin entrar en detalles técnicos. Estructura de la memoria y listado de los materiales adjuntos.
- **Objetivos del proyecto:** Descripción de los objetivos que busca el proyecto.
- **Conceptos teóricos:** Breve explicación de los conceptos teóricos importantes para la comprensión de la solución propuesta.
- **Técnicas y herramientas:** Descripción de las técnicas metodológicas y herramientas utilizadas para la organización y desarrollo del proyecto.
- **Aspectos relevantes del desarrollo:** Exposición de los problemas y soluciones que se llevaron a cabo en el transcurso de la realización del proyecto.

- **Trabajos relacionados:** Proyectos y aplicaciones deportivas que me han servido de inspiración para desarrollar el proyecto.
- **Conclusiones y líneas de trabajo futuras:** Conclusiones obtenidas tras la realización del proyecto y posibilidades de mejora.

## 1.2. Materiales adjuntos entregados

Los materiales que se adjuntan junto con la memoria en el usb son:

- Aplicación Angular para el desarrollo del front-end.
- Aplicación Flask para el desarrollo de los algoritmos de predicciones, uso de imageAI, generación de gráficos.
- Vídeos de uso de la aplicación.
- Anexos del proyecto.
- Memoria del proyecto.
- Máquina virtual para ejecutar el proyecto en local.
- Fichero README con claves de la máquina virtual como el usuario, correo de Google para probar las funcionalidades que requieren iniciar sesión.

Otros recursos accesibles a través de internet:

- Página web del proyecto: <https://futbostats.netlify.app/>
- Repositorio del proyecto en GitHub: <https://github.com/MiguelExtremo/TFG>

---

## 2. Objetivos del proyecto

---

A continuación, voy a detallar en tres tipos los objetivos que se persiguen con el proyecto y que han motivado la realización del mismo.

### 2.1. Objetivos generales

- Desarrollar una aplicación web de fútbol y estadísticas que permita a los usuarios consultar datos y estadísticas de forma interactiva.
- Facilitar al usuario la interpretación de los algoritmos y estadísticas mediante gráficas y tablas.
- Implementar en una aplicación web un sistema de tracking para visualizar el movimiento de los jugadores en el campo.
- Proporcionar al usuario información actualizada sobre estadísticas.
- Desarrollar una interfaz de usuario intuitiva.

### 2.2. Objetivos técnicos

- Utilizar el *framework Angular* para estructurar un front-end modular y mantenable, que permita que el proyecto crezca y mejore en el futuro de forma sencilla. Además, adaptar la aplicación haciéndola responsive para mejorar la compatibilidad de la aplicación web con móviles.
- Desarrollar servicios de back-end usando *Flask* para manejar peticiones *REST* desde el front-end.

- Implementar protección de rutas mediante el uso de *guards* en Angular para proteger las rutas a usuarios en función de roles.
- Integrar análisis de vídeos deportivos implementando una herramienta de tracking para observar cómo los jugadores se mueven en el campo.
- Integrar una API externa que me proporcione datos actualizados de partidos, equipos y ligas. Manejar errores para garantizar una buena experiencia en el usuario.
- Utilizar los datos de *StatsBomb* para incluir métricas avanzadas como los goles esperados (xG), generar mapas de calor, mapas de pases. Desarrollar algoritmos que generen predicciones detalladas sobre clasificaciones y equipos.
- Desplegar la aplicación para su puesta en producción.
- Utilizar un sistema de control de versiones como *Git* para facilitar el seguimiento de los cambios en el código de la aplicación. Llevar a cabo buenas prácticas de desarrollo de software como revisiones de código para mantener un código limpio y coherente.
- Desarrollar pruebas unitarias para asegurar que el código funcione correctamente.
- Crear guías de instalación, configuración, uso para facilitar a los usuarios el uso del proyecto.

### 2.3. Objetivos personales

- Introducirme al mundo del análisis deportivo y la estadística.
- Formarme en nuevos lenguajes de programación (HTML5, CSS3, TypeScript) y en el desarrollo de una aplicación web para navegador de escritorio y móvil.
- Explorar metodologías y herramientas utilizadas globalmente en el mundo laboral como Angular para el desarrollo *front-end* y Flask para el desarrollo *back-end*.
- Introducirme en tecnologías de *tracking* de objetos o personas en vídeos con algoritmos de inteligencia artificial.

---

## 3. Conceptos teóricos

---

En este apartado, se presentan los conceptos teóricos que fundamentan el desarrollo y el funcionamiento de la aplicación web deportiva centrada en el análisis de estadísticas de fútbol. La comprensión de estos conceptos es importante para apreciar la utilidad y la innovación que aporta esta herramienta al análisis y predicción en el ámbito futbolístico.

A continuación, se van a explicar los conceptos de API REST, goles esperados ( $xG$ ), puntos esperados y el funcionamiento de *imageAI* para realizar el tracking de objetos y en este caso jugadores de fútbol. Además, se expondrán los proveedores de datos que utiliza FutboStats.

### 3.1. API REST

Una *API REST* (Representational State Transfer) es una interfaz de programación de aplicaciones que usa un conjunto de reglas para construir y consumir servicios web de manera segura [1]. Utiliza los métodos HTTP estándar y los principios de diseño *RESTful*. Las API REST son ampliamente utilizadas en el desarrollo web debido a su simplicidad, escalabilidad y compatibilidad con HTTP. Algunos de los conceptos clave de una *API REST* son:

- En REST todo es conocido como un recurso. Un recurso es cualquier entidad que se pueda identificar y manejar, como un usuario o un producto. Cada recurso se representa con una URL única.
- Las URLs se utilizan para identificar recursos.
- REST utiliza los métodos HTTP estándar para realizar operaciones sobre los recursos. Estos métodos son GET (Recuperar información

de un recurso), POST (Crear un nuevo recurso), PUT (Actualiza un recurso existente), DELETE (Elimina un recurso).

- Los recursos se pueden representar en diferentes formatos, como JSON, XML o HTML. JSON es el formato más común debido a su simplicidad y facilidad de uso.
- Las API REST no tienen un estado, por lo que cada solicitud del cliente al servidor debe contener toda la información necesaria para entender y procesar la solicitud. El servidor no mantiene el estado de la sesión del cliente entre solicitudes.

## 3.2. Proveedores de información

Un proveedor de información es una entidad que recopila, procesa y distribuye datos específicos a los usuarios. Estos proveedores pueden ofrecer datos en tiempo real, históricos para integrar en una aplicación web.

### 3.2.1 StatsBomb

StatsBomb es una empresa líder en el análisis de datos deportivos, especializada en fútbol. Proporciona datos detallados y avanzados sobre partidos de fútbol, incluyendo información sobre eventos como pases, tiros, entradas. Estos datos son utilizados por clubes, analistas, y medios de comunicación para obtener una visión más profunda del rendimiento y las tácticas en el fútbol.

Además, se puede acceder a sus datos a través de la biblioteca de Python llamada *StatsBombPy*. Proporciona una interfaz simple y eficiente para manipular los datos, facilitando el trabajo a analistas y desarrolladores [12]. Un ejemplo de archivo JSON que devuelve *StatsBombPy* se puede encontrar en el repositorio de *GitHub* de *StatsBomb*.

[open-data / data / matches / 11 / 27.json](#)

```

1   [
2     {
3       "match_id" : 3825848,
4       "match_date" : "2015-09-23",
5       "kick_off" : "20:00:00.000",
6       "competition" : {
7         "competition_id" : 11,
8         "country_name" : "Spain",
9         "competition_name" : "La Liga"
10      },
11      "season" : {
12        "season_id" : 27,
13        "season_name" : "2015/2016"
14      },
15      "home_team" : {
16        "home_team_id" : 221,
17        "home_team_name" : "Levante UD",
18        "home_team_gender" : "male",
19        "home_team_group" : null,
20        "country" : {
21          "id" : 214,
22          "name" : "Spain"
23        }
24      }
25    }
26  ]

```

Figura 3.1: Ejemplo de respuesta JSON StatsBombPy en [GitHub](#)

En la figura 3.2 se muestra como se puede acceder a los eventos de un partido en específico, utilizando *StatsBombPy*. En concreto, se accede por el identificador y el año utilizando la función '*matches*' de StatsBomb para obtener los eventos del mundial de futbol 2022. Después, realizo un filtrado aún mayor para obtener los eventos de la final del mundial de 2022.

```

from statsbombpy import sb
def matchMomentumFinal():
    world_cup = sb.matches(competition_id=43, season_id=106)
    world_cup[world_cup['competition_stage'] == 'Final']

```

Figura 3.2: Filtrado básico de datos con Python utilizando StatsBombPy

### 3.2.2 RapidAPI y API-FOOTBALL

RapidAPI es una plataforma que permite a los desarrolladores descubrir, probar y conectar con miles de APIs de diversos proveedores desde un único lugar. Funciona como un mercado de APIs, facilitando la integración de servicios externos en aplicaciones mediante un proceso unificado [11].

API-FOOTBALL es una de las miles de APIs externas que se pueden encontrar en RapidAPI. Esta API conforma un servicio que proporciona datos detallados y en tiempo real sobre fútbol. Ofrece información sobre ligas, equipos, jugadores, estadísticas de partidos, resultados en vivo, y eventos detallados como goles, tarjetas y sustituciones. La API está diseñada para ser fácil de integrar y ofrece datos actualizados para mejorar la experiencia del usuario final [6].

### 3.3. Goles esperados (xG)

Los goles esperados (*expected goals* en inglés) constituyen una métrica estadística para calcular la probabilidad de que un disparo acabe o no en gol dependiendo de factores como la distancia a la portería, coordenadas del jugador en el campo en el momento del disparo, parte del cuerpo con la que el jugador remató, si el disparo es o no penalti, si es un tiro de falta o el ángulo de disparo.

Si cogemos todos los disparos producidos en una temporada por un equipo podemos llegar a la conclusión de que tan eficiente es el equipo en el ataque y dónde puede mejorar. Es decir, un equipo puede identificar cuales son sus fortalezas y debilidades y también sacar conclusiones del rendimiento individual de los jugadores [3].

Los goles esperados se calculan utilizando datos sobre disparos y goles del pasado. Con esos disparos se construye un modelo de predicción que sea capaz de predecir si un disparo nuevo va a acabar en gol o no. Podemos destacar algunas de las utilidades que tienen los goles esperados:

- **Eficiencia defensiva y ofensiva de los equipos:** los goles esperados son utilizados en el análisis estadístico del fútbol para comprender y mejorar las características ofensivas y defensivas de los equipos y con ello mejorar su rendimiento. Esta métrica ayuda a definir ciertas tendencias a largo plazo en el rendimiento de equipos y jugadores y por conseciente ayuda a reflexionar a los equipos y jugadores sobre que estrategia deben tomar para mejorar su rendimiento.

- **Analizar el rendimiento de los jugadores:** si comparamos los goles reales de un jugador con los predecidos, podemos evaluar su eficacia en las situaciones de ataque y su aportación en la ofensiva del equipo. Ayuda a conocer el desempeño del jugador y a identificar aspectos específicos en los que puede mejorar.
- **Predicciones y tendencias a largo plazo:** se utilizan datos de disparos pasados para tratar de predecir tendencias futuras de un equipo y sus jugadores.

## Cómo calcular los goles esperados (xG)

Los goles esperados se calculan analizando datos históricos de tiros y goles. Se construyen modelos matemáticos que asignan una puntuación a cada disparo en función de la probabilidad de acabar en gol. Estas puntuaciones se suman para obtener el xG total de un equipo o jugador. En mi modelo de goles esperados obtengo los registros históricos de los disparos de las 5 grandes ligas europeas (La Liga, Ligue 1, Premier League, Bundesliga, Serie A) en la temporada 2015/2016. Con estos se calculan las probabilidades de goles esperados para todos los tiros utilizando un modelo de aprendizaje automático, en este caso, un modelo de regresión logística [16].

La regresión logística es un método de análisis estadístico utilizado para modelar la relación entre una variable dependiente binaria y una o más variables independientes. A diferencia de la regresión lineal, que es utilizada para predecir valores continuos, la regresión logística se utiliza cuando la variable dependiente es categórica y toma uno de dos posibles valores como 'sí' o 'no', o en el caso de los goles esperados tomaría 'gol' o 'no gol'.

La fórmula de la regresión logística es la siguiente:

$$P(Y = 1|X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_n X_n)}}$$

donde:

- $P(Y = 1|X)$  es la probabilidad de que la variable dependiente  $Y$  sea 1 dado el conjunto de variables independientes  $X$ .
- $\beta_0$  es la intersección (término constante).
- $\beta_1, \beta_2, \dots, \beta_n$  son los coeficientes de las variables independientes  $X_1, X_2, \dots, X_n$ .
- $e$  es la base del logaritmo natural.

El proceso para calcular estas probabilidades consiste en entrenar el modelo con datos históricos de tiros y después utilizar ese modelo para predecir las probabilidades de gol para nuevos tiros.

La regresión logística es un modelo adecuado para calcular los goles esperados ( $xG$ ) por varias razones. Primero, controla problemas con variables dependientes binarias, como predecir si un tiro resultará en gol(1) o no(0). Además, proporciona probabilidades para cada predicción, lo cual es útil para medir la calidad de oportunidades de gol. Los coeficientes obtenidos son fáciles de interpretar y ayuda a entender cómo diferentes factores influyen en la probabilidad de gol.

Es un modelo computacionalmente eficiente y fácil de implementar, permitiendo así llevar a cabo su desarrollo utilizando herramientas y bibliotecas estándar de Python.

La figura 3.3 representa el modelo de goles esperados ( $xG$ ), donde podemos observar que la probabilidad de que un disparo termine en gol es mayor cuanto menor se la distancia a la portería (círculos de color amarillo). En cambio, la probabilidad disminuye a medida que los disparos son más lejanos (círculos de color más oscuro).

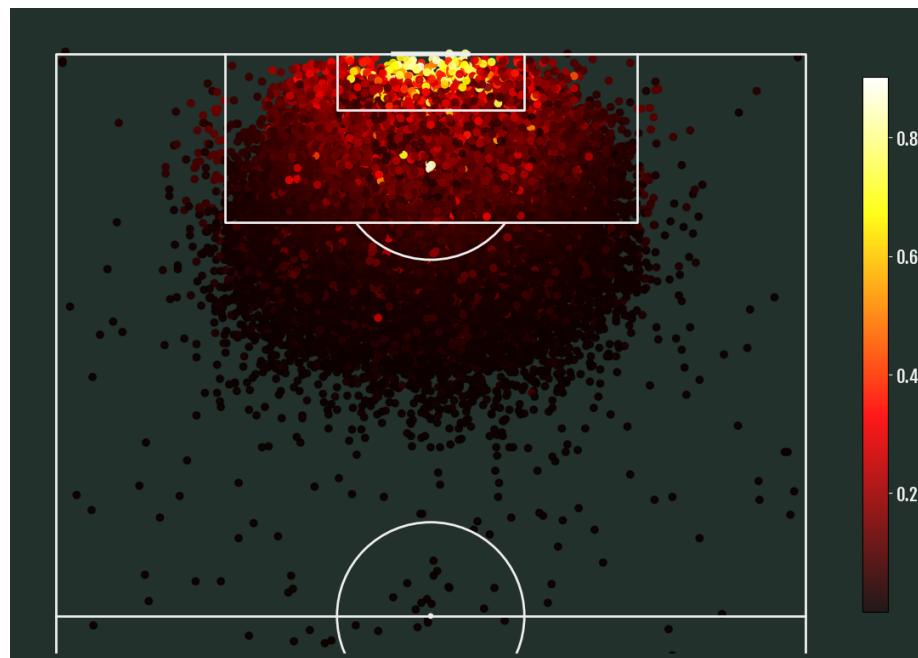


Figura 3.3: Representación del modelo de goles esperados ( $xG$ )

Gracias a la representación del modelo podemos ver como claramente la probabilidad de que un disparo acabe en gol o no depende potencialmente de la localización desde donde se realice el disparo. Cuanto más cerca de la portería se efectúe el disparo, mayor será la probabilidad de que ese disparo acabe en gol.

De los goles esperados se pueden sacar las siguientes conclusiones [13]:

- Si el ángulo del disparo es menor, la probabilidad de que un disparo acabe en gol es mayor. Por ello, son mejores las zonas centrales en lugar de las laterales.
- Los remates a puerta con los pies tienen mayor probabilidad de acabar en gol que los remates de cabeza desde la misma distancia a la portería.
- La posición, el ángulo, la distancia y otras propiedades del disparo son más importantes que el jugador que lo realiza.
- Un jugador es mejor que otro si tiene la capacidad de generar más cantidad de disparos desde zonas con mayor probabilidad de gol.

### 3.4. Puntos esperados (xPoints)

Los puntos esperados (*expected points* en inglés) es una métrica que nos permite calcular la cantidad de puntos que un equipo obtendría si simulásemos muchas veces el resultado de un partido, basándonos en la probabilidad que otorga los goles esperados (xG) a cada tiro.

El modelo de puntos esperados utiliza la métrica de los goles esperados de un partido. Una vez tenemos los goles esperados de cada equipo, se genera mediante un modelo de distribución, lo que hubiera ocurrido si simuláramos el mismo partido miles de veces y así podríamos obtener la probabilidad que tendría cada equipo de ganar, empatar y perder con esos goles esperados. Acabamos obteniendo un porcentaje de probabilidad de victoria para cada equipo y un resultado (victoria, empate, derrota) para ambos equipos.

Después de simular con el modelo de distribución y multiplicar las probabilidades obtenidas por los valores de cada resultado (victoria son 3 puntos, empate 1 punto para cada equipo y derrota 0 puntos), obtendríamos así los puntos esperados de ese partido en concreto si ocurriera miles de veces [2]. Ahora, si en vez de hacerlo con un partido, lo hacemos con todos los partidos de una temporada para todos los equipos obtendríamos finalmente para cada equipo unos puntos esperados y por consecuente una clasificación esperada.

Como modelo de distribución, he utilizado la distribución de Poisson [14]. La distribución de Poisson es una distribución de probabilidad que describe el número de eventos que ocurrirán en un intervalo de tiempo fijo o en una región específica con la condición de que ocurran con una tasa promedio constante y de manera independiente entre sí.

La función de probabilidad de la distribución de Poisson es:

$$P(X = k) = \frac{\lambda^k e^{-\lambda}}{k!}$$

Donde:

- $P(X = k)$  es la probabilidad de observar  $k$  eventos en un intervalo.
- $\lambda$  es el número promedio esperado de eventos en el intervalo.
- $k$  es el número de eventos observados.
- $e$  es la base del logaritmo natural.

La distribución de Poisson se utiliza para simular el número de goles que un equipo podría marcar en un partido, dada su tasa promedio de goles esperados. Es decir, es utilizada para generar un número aleatorio de goles para cada equipo en cada simulación de partido, basado en sus goles esperados.

Para cada partido, se obtienen los valores de xG para ambos equipos y se realizan las múltiples simulaciones donde para cada simulación se genera un número aleatorio de goles para cada equipo usando la distribución de Poisson con los valores de goles esperados como la tasa media. Para cada simulación, se comparan los goles generados para determinar el resultado del partido y repartir los puntos entre los equipos. Finalmente, se obtienen los puntos esperados para todos los equipos.

Las siguientes tablas muestran la representación de las clasificaciones de puntos esperados y la real. La tabla 3.1 muestra un ejemplo de la aplicación de la métrica goles esperados(xG) de la temporada 2015/2016 de la liga española de fútbol. La tabla 3.2 muestra la clasificación real de la temporada 2015/2016 de la liga española.

Posición	Equipo	Puntos
1	Barcelona	84.62
2	Real Madrid	72.83
3	Atlético Madrid	69.89
4	Sevilla	59.40
5	Athletic Club	58.00
6	Real Sociedad	52.77
7	Celta Vigo	52.54
8	Málaga	52.40
9	Eibar	52.39
10	Villarreal	50.07
11	Rayo Vallecano	47.90
12	RC Deportivo La Coruña	47.46
13	Valencia	46.66
14	Espanyol	46.38
15	Real Betis	44.28
16	Getafe	43.57
17	Sporting Gijón	41.78
18	Granada	40.83
19	Las Palmas	40.46
20	Levante UD	40.01

Tabla 3.1: Clasificación predicha para la temporada 2015/16 de la liga española empleando la métrica de goles esperados.

Podemos observar que el equipo campeón de la liga española en la temporada 2015/2016 empleando la métrica de goles esperados es el Barcelona, seguido del Real Madrid y el Atlético de Madrid.

Los tres equipos que descienden son el Granada, Las Palmas y el Levante UD.

Posición	Equipo	Puntos
1	Barcelona	91
2	Real Madrid	90
3	Atletico Madrid	88
4	Villarreal	64
5	Athletic Club	62
6	Celta Vigo	60
7	Sevilla	52
8	Málaga	48
9	Real Sociedad	48
10	Real Betis	45
11	Las Palmas	44
12	Valencia	44
13	Eibar	43
14	Espanyol	43
15	RC Deportivo La Coruña	42
16	Granada CF	39
17	Sporting Gijón	39
18	Rayo Vallecano	38
19	Getafe	36
20	Levante UD	32

Tabla 3.2: Clasificación real para la temporada 2015/16 de la liga española.

La clasificación real y el modelo de goles esperados coinciden en las posiciones de los tres primeros equipos.

En cambio, realmente descendieron el Rayo Vallecano, Getafe y Levante UD, por lo que el único equipo que realmente descendió de los predichos por el modelo fue el Levante UD.

### 3.5. ImageAI

*ImageAI* es una biblioteca de *Python* que utiliza inteligencia artificial para facilitar la implementación de tareas de detección de objetos, clasificación de imágenes. Esta biblioteca utiliza modelos de aprendizaje profundo pre-entrenados y proporciona un sistema para facilitar su integración en proyectos diversos. La biblioteca es compatible con varios modelos de detección de objetos como *YOLOv3*, *Retinanet* y *TinyYOLOv3*, y se basa en frameworks robustos como *TensorFlow* y *Keras* [9].

*TinyYOLOv3* es una versión simplificada y más ligera de *YOLOv3*, diseñada para funcionar en dispositivos con recursos limitados. Pierde algo de precisión a cambio de mejorar la velocidad y reducir los requisitos de hardware.

*YOLOv3* es un modelo que se caracteriza por su alta velocidad y precisión en la detección de objetos. Puede procesar imágenes en tiempo real y es capaz de detectar objetos de diferentes tamaños y formas debido a su arquitectura [7].

*Retinanet* es un modelo de detección de objetos de una sola etapa que funciona bien con objetos densos y de pequeño tamaño difíciles de detectar [4]. En la tabla 3.3 se muestra un resumen de las características a destacar de cada modelo:

Modelo	Velocidad	Precisión	Requisitos de Hardware
YOLOv3	Alta	Alta	GPU medios
Retinanet	Media	Muy Alta	GPU potente
TinyYOLOv3	Muy Alta	Media	Recursos limitados

Tabla 3.3: Comparación de modelos de detección de objetos

En una aplicación de estadísticas de fútbol, *ImageAI* puede ser una herramienta interesante para el análisis y la interpretación de videos de partidos. Se puede analizar el movimiento y la posición de los jugadores en el campo encontrando patrones de juego y estrategias de cada equipo.

---

## 4. Técnicas y herramientas

---

### 4.1. Metodologías

#### Kanban

Kanban es un método ágil de gestión de proyectos y control de flujo que se centra en la visualización del trabajo y la optimización del flujo de tareas. Kanban utiliza un tablero visual, dividido en columnas que representan las distintas etapas del proceso de trabajo(Por hacer, Listo para hacer, En progreso, En revisión y Hecho) para gestionar y monitorizar las tareas. Las tarjetas se van moviendo a través del tablero conforme avanzan en el proceso [15].

#### Scrum

Scrum es un marco de trabajo ágil utilizado para el desarrollo y la gestión de proyectos. Se basa en ciclos de trabajos iterativos e incrementales llamados sprints que duran entre dos y cuatro semanas. En cada sprint se trata de entregar un incremento del producto hasta conseguir el proyecto completo [17].

### 4.2. Control de versiones

- Herramienta elegida: [Git](#)

Git es un sistema de control de versiones distribuido que se utiliza para gestionar y hacer el seguimiento de los cambios en el código fuente durante

el desarrollo de software. Git permite a múltiples desarrolladores trabajar de manera simultánea en un proyecto sin interferir con el trabajo de los demás. Cada desarrollador tiene una copia completa del repositorio, incluyendo todo el historial de cambios.

### 4.3. Hosting del repositorio

- Herramienta elegida: [GitHub](#)

GitHub es una plataforma de alojamiento de repositorios basada en Git que permite a los desarrolladores almacenar, gestionar y compartir su código fuente de manera colaborativa. GitHub es una de las herramientas más populares para el desarrollo software debido a sus características que facilitan el trabajo en equipo y la gestión de proyectos y su integración con Git.

### 4.4. Entorno de desarrollo

- Herramienta elegida: [Visual Studio Code](#)

Visual Studio Code es un editor de código fuente gratuito, multiplataforma desarrollado por Microsoft. Se ha convertido en una de las herramientas de desarrollo más populares debido a su versatilidad(permite una amplia variedad de lenguajes de programación), rendimiento y amplio conjunto de características. Su integración con Git facilita el control de versiones, revisión del código y la colaboración en equipo.

### 4.5. Desarrollo front-end

- Herramientas consideradas: [Flask](#) , [Angular](#)
- Herramienta elegida: [Angular](#)

Flask es un microframework de Python utilizado principalmente para el desarrollo del back-end de aplicaciones web. Aunque es posible gestionar las vistas con Flask, este no está diseñado específicamente para el desarrollo de

una aplicación front-end.

En cambio, Angular es un framework de desarrollo de aplicaciones web front-end de código abierto, mantenido por Google. Está basado en TypeScript y permite construir aplicaciones web de una sola página(SPA) mediante un enfoque modular y componentes reutilizables. Angular proporciona una estructura robusta para gestionar vistas dinámicas y facilita la creación de interfaces de usuario interactivas y complejas. Además, permite la construcción de servicios para hacer peticiones a APIs externas, guards para la protección de rutas e instalar dependencias avanzadas de una manera sencilla. Angular utiliza el llamado Angular CLI (Command Line Interface) para crear componentes y probar la aplicación con comandos. Al ser mantenido por Google y contar con una gran comunidad de desarrolladores, Angular recibe actualizaciones regularmente, mejoras de seguridad y un soporte extenso, lo que asegura su relevancia y confiabilidad a largo plazo.

En resumen, Angular es una mejor herramienta para el desarrollo del front-end de aplicaciones web debido a su arquitectura modular, soporte de TypeScript, capacidades de enlace de datos bidireccional y su ecosistema robusto, en comparación con Flask, que está más orientado al desarrollo del back-end.

## 4.6. Desarrollo back-end

- Herramientas consideradas: [Django](#), [Flask](#)
- Herramienta elegida: [Flask](#)

Flask y Django son dos de los frameworks más populares para el desarrollo de aplicaciones web en Python, pero tienen diferentes enfoques y características.

Flask es un microframework minimalista para el desarrollo de aplicaciones web. Fue diseñado para ser simple y flexible, permitiendo a los desarrolladores elegir las bibliotecas y herramientas que desean utilizar.

Flask proporciona lo esencial al usuario para comenzar, dejando a su elección las bibliotecas y herramientas adicionales. Además, es altamente modular y extensible, lo que permite agregar solo los componentes necesarios y su curva de aprendizaje es suave (fácil de aprender y de usar) haciéndolo ideal para proyectos pequeños.

En cambio, Django viene con una estructura y componentes predefinidos que pueden limitar su uso si se desean utilizar otras herramientas. Además, tiene una curva de aprendizaje más fuerte que Flask debido a su complejidad y la cantidad de funcionalidades integradas.

En resumen, Flask es ideal para proyectos que requieren flexibilidad, simplicidad y un mayor control sobre los componentes del backend. Es adecuado para desarrolladores que quieren construir proyectos desde cero y elegir las herramientas que mejor se adapten a sus necesidades. Django, por otro lado, es más adecuado para proyectos más grandes y complejos que se benefician de un conjunto de funcionalidades integradas y una estructura definida desde el inicio.

## 4.7. Gestión de usuarios con Google

- Herramienta elegida: [Google Cloud](#)

La gestión de inicio de sesión con Google Cloud OAuth 2.0 es una estrategia efectiva para autenticar usuarios en una aplicación web. Este método ofrece una alta seguridad mediante la protección de datos. Mejora la experiencia del usuario para simplificar el inicio de sesión y aprovechar la confianza en la seguridad de Google. Permite una gestión de identidad unificada e integración con otros servicios de Google Cloud, facilitando la sincronización de datos. Google asegura un manejo eficiente de un gran número de usuarios. Al ser un estándar ampliamente adoptado, OAuth 2.0 garantiza compatibilidad y se beneficia de actualizaciones regulares que mantienen las mejores prácticas y estándares de seguridad.

## 4.8. Documentación

- Herramientas consideradas: [Overleaf](#), [Texmaker](#)
- Herramienta elegida: [Overleaf](#)

TexMaker y Overleaf son dos herramientas populares para trabajar con documentos en LaTeX.

LaTeX es un sistema de composición de documentos y un lenguaje marcado para la creación de textos científicos y técnicos de alta calidad.

Overleaf ofrece varias ventajas sobre TexMaker, destacándose por su capacidad de colaboración en tiempo real, acceso desde cualquier dispositivo con conexión a internet, y no requerir instalación local ni configuración de compiladores. Además, Overleaf proporciona compilación automática y visualización instantánea de cambios, integración con GitHub y otros servicios de almacenamiento en la nube, y una amplia biblioteca de plantillas y recursos, lo que simplifica el desarrollo y mantenimientos de documentos en LaTeX.

## 4.9. Librerías

### Angular



Figura 4.1: Logo de Angular

Las librerías instaladas en Angular para desarrollar este proyecto han sido las siguientes:

#### OAuth

[OAuth2](#) es un estándar de autorización que permite a las aplicaciones obtener acceso limitado a los recursos del usuario en un servidor sin compartir las credenciales del usuario. Es decir, permite que las aplicaciones interactúen con servicios web utilizando tokens de acceso en lugar de contraseñas.

La biblioteca 'angular-oauth2-oidc' facilita la implementación de OAuth2 en aplicaciones Angular. Proporciona herramientas y servicios que simplifican la integración de autenticación y autorización con proveedores de identidad como Google.

Esta librería ha sido utilizada para implementar un sistema de inicio de sesión con Google. Esto permite a los usuarios autenticarse en la aplicación de manera sencilla con sus cuentas de Google, ofreciendo una experiencia de usuario sencilla y segura.

#### Flex-Layout

La biblioteca [flex-layout](#) es una herramienta en Angular que facilita la creación de interfaces de usuario responsivas y adaptables utilizando Flexbox

CSS y CSS Grid. Esta biblioteca permite a los desarrolladores definir el diseño y la disposición de los elementos de una aplicación de manera intuitiva y eficiente.

Para explorar y probar cómo funcionan las distintas directivas y configuraciones de flex-layout, he utilizado la [página de demostración y documentación](#) que aporta dicha biblioteca. Esta página interactiva te permite experimentar con diferentes configuraciones de diseño y ver los resultados en tiempo real, lo cuál nos permite aprender rápido como se usa y aplicarlo en el proyecto real.

### Flaticon

[Flaticon](#) es una plataforma que ofrece una amplia variedad de iconos gratuitos para usar en una aplicación web. Es una herramienta popular para diseñadores gráficos, desarrolladores web y cualquiera que este buscando iconos de calidad para sus proyectos. Esta biblioteca se puede instalar vía [NPM](#) para ser utilizada en Angular.

Flaticon es una buena biblioteca de iconos para Angular debido a su amplia variedad, opciones de personalización, facilidad de integración y disponibilidad en múltiples formatos.



Figura 4.2: Logo de Flaticon

### Animate.css

[Animate.css](#) es una biblioteca de animaciones CSS que facilita la implementación de efectos animados a los elementos de una página web. Esta biblioteca ofrece una colección variada de animaciones CSS y no se requieren conocimientos de JavaScript, lo que la hace accesible para desarrolladores de todos los niveles.

En pocas palabras, su facilidad de uso, variedad de animaciones, consistencia, rendimiento y capacidad de personalización la convierten en una buena opción para mejorar la apariencia y experiencia de usuario en cualquier sitio web.



Figura 4.3: Logo de Animate.css

### Moment.js

[Moment.js](#) es una popular biblioteca de JavaScript para la manipulación y formateo de fechas y tiempos. Proporciona una manera sencilla de trabajar con fechas y tiempos, permitiendo realizar operaciones como formateo, manipulación y validaciones.

Esta librería es utilizada en mi proyecto para formatear la fecha de el datepicker y el usuario seleccione una fecha que se familiarice con el por el idioma. Además, se debe de formatear la fecha nuevamente para enviarla a la API externa y que la petición sea un éxito.

Esta biblioteca se puede instalar en Angular vía [NPM](#).



Figura 4.4: Logo de Moment.js

### Sweetalert2

[SweetAlert2](#) es una biblioteca de JavaScript que facilita la creación de alertas y diálogos personalizados en aplicaciones web. A diferencia de las alertas estándar del navegador, que son bastante básicas y limitadas en funcionalidad y diseño, SweetAlert2 permite crear alertas visualmente atractivas y personalizables que pueden mejorar la experiencia del usuario. Además, esta biblioteca está diseñada para ser responsive, por lo que las alertas se verán bien en dispositivos de diferente tamaño.



Figura 4.5: Logo de SweetAlert2

## Angular Material

[Angular Material](#) es una biblioteca de componentes de interfaz de usuario (UI) diseñada y mantenida por Angular de Google. Está basada en el sistema de diseño Material Design de Google, que proporciona una guía integral para el diseño visual, de movimiento y de interacción en todas las plataformas y dispositivos. Angular Material ofrece una amplia variedad de componentes preconstruidos que se ajustan a las especificaciones de Material Design. Utilizar estos componentes permite a los desarrolladores crear rápidamente interfaces de usuario funcionales y atractivas sin tener que diseñar y desarrollar desde cero. Una ventaja respecto a competidores como [Bootstrap](#) es que Angular Material está diseñado específicamente para trabajar con Angular, lo que significa que se integra perfectamente con las directivas, componentes y el ciclo de vida de Angular.



Figura 4.6: Logo de Angular Material

## Flask



Figura 4.7: Logo de Flask

Las librerías instaladas en Flask para desarrollar este proyecto han sido las siguientes:

## StatsBombPy

[StatsBombPy](#) es una biblioteca de Python desarrollada para facilitar el acceso al uso de los datos de StatsBomb. Esta biblioteca permite a los usuarios descargar y manipular los datos de fútbol de StatsBomb de manera sencilla y eficiente. Es especialmente útil para analistas de datos, científicos de datos y desarrolladores que trabajan en proyectos relacionados con el análisis de datos de fútbol.

StatsbombPy se utiliza para acceder a la API de StatsBomb y realizar distintas operaciones con los datos de fútbol como descargarlos, manipularlos, filtrarlos y construir gráficos con ellos que nos ayuden a entender la información.

El único inconveniente que tiene esta librería es que los datos que hay en abierto gratuitamente para utilizar son limitados. Por ejemplo, de las 5 grandes ligas europeas solo se dispone de todos los eventos completamente de la temporada 2015/2016. Aún así, tiene datos actualizados como los del pasado mundial de fútbol 2022. Según se fueran añadiendo datos de más temporadas se podría trabajar con ellos. En el GitHub de StatsBomb se pueden encontrar todos los datos que hay disponibles en [open-data](#).

En resumen, esta librería proporciona datos detallados sobre partidos y jugadores y está diseñada para facilitar el acceso y el uso de los datos.



Figura 4.8: Logo de StatsBomb

## MplSoccer

[MplSoccer](#) es una biblioteca de Python diseñada específicamente para la visualización y análisis de fútbol. Proporciona herramientas y funcionalidades especializadas para trabajar con datos de fútbol como dibujar campos de fútbol, mapas de tiros, y otros gráficos que son comunes en el análisis de rendimiento deportivo. Mplsoccer facilita la creación de gráficos que representan campos de fútbol. Estos gráficos pueden ser utilizados para representar la posición de jugadores, eventos de partidos (pases, tiros y entradas) y otros datos relevantes.

Esta librería permite una gran personalización de los gráficos y se integra muy bien con statsbomb.



Figura 4.9: Logo de MplSoccer

## Pandas

[Pandas](#) es una biblioteca de Python de código abierto que proporciona estructuras de datos y herramientas de análisis de datos de alto rendimiento y fáciles de usar. Sus principales estructuras de datos son las Series y los DataFrames, que permiten manejar y analizar datos tabulares de una manera eficiente. Además, Pandas permite leer datos de una variedad de formatos como CSV, Excel, SQL, JSON. Pandas se integra bien con bibliotecas de visualización como Matplotlib facilitando la creación de gráficos a partir de DataFrames.



Figura 4.10: Logo de Pandas

## Matplotlib

[Matplotlib](#) es una biblioteca de Python para crear gráficos y visualizaciones de datos. Es ampliamente utilizada en la ciencia de datos y análisis debido a su capacidad para generar una gran variedad de gráficos con alta calidad y personalización. Matplotlib se integra bien con otras bibliotecas de análisis de datos como NumPy y Pandas. Es compatible con diversos entornos y su extensa documentación y número de usuarios facilitan su uso y aprendizaje.



Figura 4.11: Logo de Matplotlib

## NumPy

[Numpy](#) es una biblioteca de Python para el cálculo científico. Proporciona soporte para arrays y matrices multidimensionales y un conjunto de funciones matemáticas para operar con los datos de manera eficiente.



Figura 4.12: Logo de NumPy

### SciPy

[SciPy](#) es una biblioteca de Python utilizada para cálculos científicos. Se construye sobre NumPy, proporcionando funciones adicionales para optimización, integración, álgebra lineal, interpolación y estadística. La biblioteca es conocida por su eficiencia y su capacidad para realizar cálculos de alto rendimiento.



Figura 4.13: Logo de Scipy

### Scikit-learn

[Scikit-learn](#) (sklearn) es una biblioteca de Python para aprendizaje automático y minería de datos. Ofrece herramientas simples y eficientes para la modelización y el análisis predictivo, incluyendo algoritmos de clasificación, regresión. Construida sobre NumPy, SciPy y Matplotlib, scikit-learn facilita la integración con otras bibliotecas científicas de Python.



Figura 4.14: Logo de Scikit-learn

### ImageAI

[ImageAI](#) es una biblioteca de Python diseñada para la implementación fácil y rápida de sistemas de inteligencia artificial para la detección y reconocimiento de objetos en imágenes y vídeos. ImageAI simplifica el uso de

modelos de aprendizaje profundo preentrenados para la detección de objetos, clasificación de imágenes y reconocimiento de acciones.

ImageAI está diseñada para ser implementada fácilmente y extrae gran parte de la complejidad técnica, permitiendo a los desarrolladores implementar modelos de detección de objetos con pocas líneas de código.

Además, soporta varios modelos preentrenados como YOLOv3, TinyYOLOv3 y Retinanet, que son precisos y eficientes en la detección de objetos. Esto permite a los desarrolladores utilizar modelos avanzados sin tener que entrenarlos desde 0.

Es una práctica interesante integrar ImageAI con Python para el tracking de jugadores en vídeos de fútbol debido a su facilidad de uso. Se puede consultar su documentación en el [Github de ImageAI](#).



Figura 4.15: Logo de ImageAI

## Otros

### Ffmpeg

[Ffmpeg](#) es una herramienta de línea de comandos de código abierto utilizada para manipular archivos multimedia, incluyendo la conversión, grabación y transmisión de audio y vídeo. Proporciona una colección de bibliotecas y programas que permiten la decodificación, codificación de archivos de vídeo y audio.

Los navegadores web soportan solo algunos codecs de vídeo, por lo que, utilizar Ffmpeg facilita transformar videos a formatos compatibles y asegura que los vídeos se reproduzcan correctamente en todos los navegadores.



Figura 4.16: Logo de Ffmpeg

### JustInMind

[JustInMind](#) es una herramienta que permite realizar prototipos de aplicaciones web para escritorio y móvil. Permite crear prototipos interactivos y completos sin necesidad de escribir código. Se pueden simular funcionalidades y el flujo de la aplicación. Utilizar una herramienta de prototipado permite identificar y solucionar problemas de usabilidad en las primeras etapas del proceso de diseño.



Figura 4.17: Logo de JustInMind

### PostMan

[Postman](#) es una herramienta popular para el desarrollo y la prueba de APIs. Permite a los usuarios diseñar, probar, documentar y monitorear sus APIs de manera eficiente. Con Postman, los usuarios pueden crear y enviar solicitudes HTTP a un servidor y ver las respuestas, lo que facilita la comprobación de la funcionalidad de la API y la detección de problemas.



Figura 4.18: Logo de Postman

Herramientas	Angular17	API EXTERNA	Flask	Memoria
HTML5	X			
CSS3	X			
TypeScript	X			
Angular		X		
API-FOOTBALL		X		
Angular Material	X			
FlatIcon	X			
Flex-Layout	X			
OAuth	X			
Animate.css	X			
Moment.js	X			
SweetAlert2	X			
Python			X	
StatsBombPy			X	
MplSoccer			X	
Pandas			X	
Matplotlib			X	
JSON	X		X	
NumPy			X	
SciPy			X	
Scikit-learn			X	
ImageAI			X	
Ffmpeg			X	
Git	X		X	X
Git-Hub		X	X	X
LaTeX				X
Overleaf				X
Google Cloud	X			
Visual Studio Code	X		X	
PostMan	X	X	X	

Tabla 4.1: Resumen de herramientas y tecnologías utilizadas en cada parte del proyecto

---

## **5. Aspectos relevantes del desarrollo del proyecto**

---

En este punto de la memoria se van a recoger los aspectos más importantes del desarrollo del proyecto. Se explicarán las decisiones que se tomaron para desarrollar el proyecto de tal forma y los problemas a los que me enfrenté y sus soluciones.

### **5.1. Inicio del proyecto**

Siempre he tenido interés en el deporte, especialmente en el fútbol y me pareció interesante profundizar en como influye la tecnología y la informática en este deporte.

El reto de crear una aplicación web dedicada a la consulta de datos y estadísticas de fútbol me atrajo. Sin conocimientos previos en el desarrollo web, me propuse este proyecto como una oportunidad para aprender y desarrollar mis conocimientos de la informática obtenidos en la carrera. A través de este proyecto, he podido explorar y aplicar conceptos de programación, manejo de datos y desarrollo web mientras descubro nuevas cosas sobre el fútbol. Es un trabajo de fin de grado que combina mis gustos personales con mis estudios, permitiéndome crear una herramienta útil y educativa para otras personas aficionadas al deporte.

Combinando estas ideas para el trabajo de fin de grado, empecé el desarrollo del mismo.

En la figura 5.1 se puede ver el logo de la aplicación web "FutboStats":



Figura 5.1: Logo de FutboStats

## 5.2. Metodologías

En el desarrollo del proyecto he utilizado las metodologías ágiles Scrum y Kanban para gestionar y optimizar el flujo de trabajo de manera efectiva.

### Scrum

Scrum es un marco de trabajo ágil que se utiliza de forma común en el desarrollo y gestión de proyectos. Esta basado en ciclos de trabajo iterativos e incrementalos llamados *sprints*. Los *sprint* en este proyecto han tenido una duración promedio de dos semanas. Este se consideró el adecuado para poder desarrollar las funcionalidades necesarias de la aplicación.

Para llevar a cabo el proceso de Scrum he seguido las siguientes etapas:

- **Planificación del *Sprint*:** al inicio de cada *sprint*, se creaban las nuevas *issues* que se basaban en los objetivos y funcionalidades que se debían completar durante el *sprint*. Cada issue tenía un objetivo claro para garantizar un buen desarrollo.
- **Ejecución del *Sprint*:** durante el *sprint*, se trabajaban las issues planificadas. La duración de dos semanas me permitían implementar nuevas características a la aplicación y resolver otros problemas que surgían en el proceso.

- **Revisión del sprint:** al finalizar el *sprint*, se realizaba una revisión de las issues y de las funcionalidades implementadas.
- **Planificación del siguiente Sprint:** se planificaban las *issues* para el siguiente *sprint*.

## Kanban

Kanban es una metodología ágil que se centra en la visualización del trabajo y la optimización del flujo de tareas. En el proyecto, se utilizó un tablero Kanban para gestionar y monitorizar las tareas a través de diferentes etapas del proceso de trabajo.

En la figura 5.2 se puede ver el tablero Kanban. Este se dividió en varias columnas, cada una representando una etapa del flujo de trabajo:

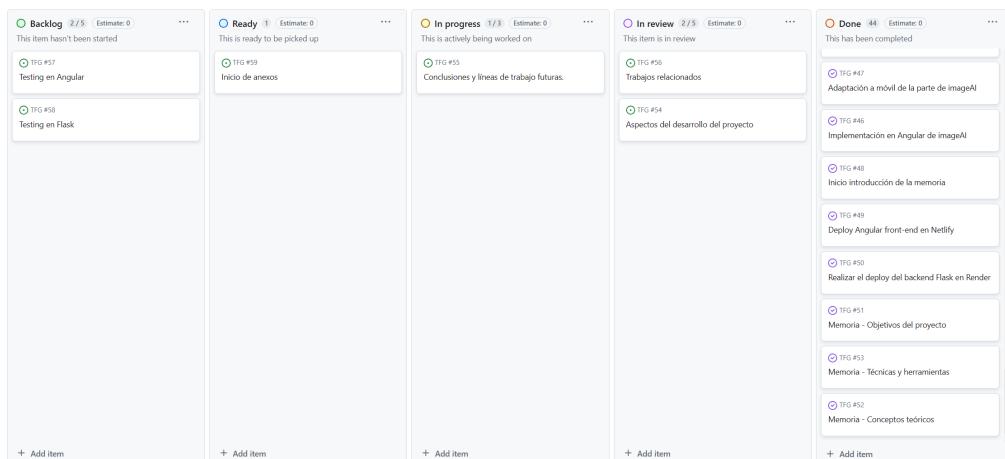


Figura 5.2: Tablero Kanban utilizado en el desarrollo del proyecto.

- **Por hacer:** esta columna contenía las tareas que aún no se habían iniciado, pero estaban dentro de la planificación del sprint, o para futuros sprints.
- **Listo para hacer:** esta columna contenía las tareas listas para ser comenzadas.
- **En progreso:** las tareas en esta columna se encontraban en desarrollo.
- **En revisión:** una vez una tarea era completada se movía a esta columna para su posterior revisión. En esta revisión, se comprobaba la calidad del código y la funcionalidad desarrollada.

- **Hecho:** las tareas que pasaban la revisión se movían finalmente a esta columna, indicando así que estaban completamente terminadas.

El uso combinado de Scrum y Kanban permitió una gestión ágil y eficiente del proyecto, proporcionando un sistema estructurado para la planificación y ejecución de las tareas. Además, este sistema era flexible ante cambios y así permitía optimizar el flujo de trabajo. Con este sistema se aseguró la entrega continua de valor a través de iteraciones incrementales para finalmente llegar al desarrollo completo de la aplicación.

### 5.3. Formación

Para llevar a cabo este proyecto, se requerían conocimientos técnicos que no disponía al principio. En concreto, se necesitaban ciertos conocimientos avanzados en Angular como la estructuración modularizada, creación de modulos, componentes, servicios, creación de rutas, protección de rutas, conexiones con servicios externos (API-FOOTBALL), conexión con mi API en Flask, como desplegar una aplicación desarrollada en Angular.

Para la formación en Angular, se realizó el siguiente curso online:

- **Curso en Udemy:** 'Angular: De cero a experto [8].'

Este curso de Udemy ha tenido mucha importancia para llevar a cabo el desarrollo del proyecto, ya que, abarca conocimiento de Angular desde lo más básico de la programación para aprender TypeScript, hasta conceptos avanzados del *framework*.

Gracias a este curso intensivo de Angular, he podido aprender a mayor velocidad y aplicar estos conocimientos del curso en la aplicación propuesta. Los conocimientos que aprendí de Angular en este curso y que aplique en el desarrollo de FutboStats fueron:

- Base de TypeScript.
- Creación de componentes.
- Creación de servicios.
- Creación de módulos.
- Implementación de Guards.

- Implementación de rutas.
- Creación y uso de formularios a nivel avanzado.
- Estructura modularizada de los proyectos.
- Uso de Angular CLI.
- Peticiones HTTP.
- Utilización de una API externa.

Este curso mitiga de gran forma la curva de aprendizaje del *framework*.

## 5.4. Desarrollo de la aplicación

### 5.4.1 Desarrollo del front-end: Angular

Una gran parte del proyecto ha sido desarrollar la aplicación web utilizando como *framework* Angular.

Antes de iniciar la aplicación, trabaje inicialmente en unos bocetos creados con la aplicación [JustInMind](#). Esta aplicación permitía hacer prototipos e incluso simulaciones de los prototipos (ver figura 5.3). De esta forma, asenté las ideas de la aplicación que quería desarrollar.

La primera vista que diseñé de la aplicación fue la de pantalla de Inicio, siendo esta la más fácil de todas, ya que no requería conocimientos avanzados. Además, creé la cabecera con el menú de navegación de la aplicación y el pie de la página. Estos primeros pasos me ayudaron a familiarizarme con el *framework*. Cabe destacar, que aunque esta vista fue la primera en diseñarse, ha sido la última en acabarse, ya que se han introducido mejoras para orientar al usuario a como usar la aplicación con pequeños tutoriales introducidos en esta misma vista.



Figura 5.3: Prototipo de la vista de inicio desarrollado en JustInMind

#### 5.4.2 Integración de APIs: API-FOOTBALL

Siguiendo con el desarrollo, me planteé la idea de hacer una vista en la que un usuario pudiera introducir un nombre de una liga y pudiera obtener la clasificación de la misma. Para ello, me puse a investigar sobre APIs externas que pudiera usar y que estuvieran bien documentadas. Finalmente, me decanté por API-FOOTBALL [6] debido a las siguientes razones:

- **Amplia cobertura de datos:** proporciona datos detallados y actualizados sobre muchas ligas, competiciones y equipos de fútbol de todo el mundo.
- **Actualizaciones en tiempo real:** ofrece actualizaciones en tiempo real, lo cual es muy importante para aplicaciones que necesitan mostrar información actualizada con frecuencia.
- **Facilidad de integración:** esta diseñada para ser fácil de integrar con distintos tipos de tecnologías de desarrollo web. Angular, proporciona endpoints RESTful que se pueden consumir fácilmente utilizando HttpClient, lo que facilita la obtención y manipulación en la aplicación. Además, tiene una documentación clara y detallada que ayuda a implementar de forma eficiente las funcionalidades necesarias.
- **Escalabilidad y fiabilidad:** esta API está construida para manejar grandes volúmenes de solicitudes, asegurando que la aplicación pueda escalar sin problemas a medida que crece en número de usuarios. La fiabilidad y disponibilidad de la API son esenciales para mantener la integridad de la aplicación y garantizar una experiencia de usuario sin interrupciones.

Una vez tenía decidida la API que iba a utilizar y la funcionalidad que quería implementar continué con el desarrollo de la vista llamada 'Competiciones' en la que el usuario podría introducir el nombre de una liga y filtrar por el año para visualizar la clasificación del año que el usuario desease.

Como primer problema que tuve que enfrentar fue entender como funcionaban los servicios y las peticiones en Angular utilizando una API externa que requería de una Api-key y de headers como parámetros en la petición http para poder recibir correctamente el JSON que la API me daba como respuesta en función de los parámetros introducidos.

Para que el usuario pudiera buscar una clasificación por el nombre de una liga, se creó un servicio con el que a través del nombre de la liga se obtenía el identificador de esa liga. Con ese identificador se llamaría posteriormente al endpoint de clasificaciones para obtener los datos pertenecientes a la clasificación de una liga. Una vez entendí estos conceptos sobre la API y sobre Angular fue sencillo corregirlo y mostrar la información en la web.

Una vez estaba implementada la funcionalidad base, trabajé en crear sugerencias interactivas al usuario para que a medida que el usuario introducía el nombre de la liga, aparecieran algunas de las posibles opciones en función de los caracteres introducidos por el usuario (ver figura 5.4).

Finalmente, para acabar esta vista implemente mejoras visuales mediante estilos CSS y librerías de Angular tanto en el formulario como en la tabla mostrada.

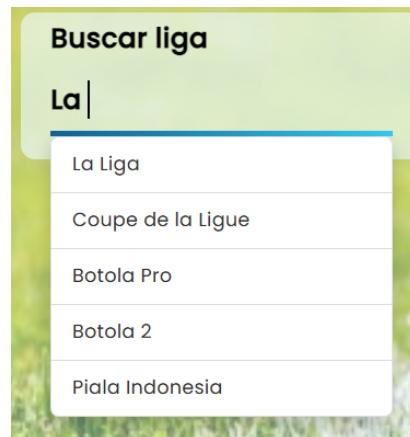


Figura 5.4: Sugerencias al usuario en la vista 'Competiciones'

Como buenas prácticas de modularización del código, por cada *endpoint* utilizado de la API creé sus interfaces para poder usar el tipado en Angular

y un servicio en el cual se hacían las distintas peticiones HTTP para obtener los datos en formato JSON.

Fue muy útil utilizar la aplicación Postman para probar las diversas peticiones y entender como funcionaban antes de llevarlas a cabo en Angular.

### 5.4.3 Autenticación

Posteriormente, me pareció interesante implementar un inicio de sesión en la aplicación. En concreto, desarrollé un inicio de sesión con Google utilizando Google Cloud y OAuth2.

Para implementar esta parte de la aplicación apliqué los conocimientos del curso de Angular sobre los Guards y las rutas. Los Guards permiten establecer restricciones sobre algunos módulos, como que el usuario no pueda acceder a ellos si no está autenticado en la aplicación. De esta forma los módulos que tuvieran los Guards implementados no se cargarían hasta que el usuario se autenticase en la aplicación. Esto supone una mejora en rendimiento a largo plazo, ya que cuantos menos módulos haya cargados en la aplicación mayor será la escalabilidad del proyecto a largo plazo. Por otro lado, investigué y me leí la documentación de la librería OAuth2 y de Google Cloud para finalizar la implementación del inicio de sesión con Google. Con la librería instalada utilicé sus funciones internas para crear un inicio de sesión y un cierre de sesión. En Google Cloud bastó con crear un nuevo proyecto y especificar la url del proyecto tanto en local como en producción para poder utilizar la funcionalidad en ambas aplicaciones.

La figura 5.5 muestra un ejemplo de la configuración de Google Cloud para los dominios en local y producción:

**Orígenes autorizados de JavaScript** ?

Para usar con solicitudes de un navegador

URI 1 *	<input type="text" value="http://localhost:4200"/>
URI 2 *	<input type="text" value="https://futbostats.netlify.app"/>
<a href="#">+ AGREGAR URI</a>	

**URI de redireccionamiento autorizados** ?

Para usar con solicitudes de un servidor web

URI 1 *	<input type="text" value="http://localhost:4200/"/>
URI 2 *	<input type="text" value="https://futbostats.netlify.app/"/>

Figura 5.5: Configuración de rutas en Google Cloud

Por último, plantée una vista que permitiera realizar la búsqueda de partidos a gusto del usuario. Para ello, creé un formulario con los campos nombre de liga, nombre de equipo y fecha. Los únicos parámetros obligatorios para realizar la búsqueda los partidos son el nombre del equipo o el nombre de la liga (uno de ellos o los dos). Si se especifica uno de ellos, todos los demás son opcionales. Si el usuario realiza mal la búsqueda o introduce una fecha en la que no hay partidos o introduce un nombre de un equipo y liga incompatibles será informado del error, para que pueda corregirlo.

El calendario que aporta Angular Material viene por defecto en inglés, por lo que utilizando la librería *Moment.js* se realizó el formateado de la fecha a idioma español. Una vez el usuario introduce la fecha, esta debe volver a formatearse a formato inglés para pasarse a la API en la petición, debido a que API-FOOTBALL solo acepta fechas en formato 'MM-DD-YYYY'.

Una vez implementada la funcionalidad, el siguiente paso fue mostrar la información de forma ordenada y visual en tarjetas.

La figura 5.6 muestra la representación de los partidos buscados por un usuario:



Figura 5.6: Ejemplo de búsqueda de partidos del Burgos CF

Posteriormente, se trabajó en hacer *responsive* todas las pantallas de la aplicación para poder usarla en un dispositivo móvil. Se utilizaron *media-queries* y *flex-layout* para ajustar los componentes de la aplicación. En la figura 5.7 se muestra la vista 'Partidos' adaptada totalmente para dispositivos móviles:



Figura 5.7: Adaptación a móvil de FutboStats

#### 5.4.4 Desarrollo del backend: Flask

Otra parte importante del proyecto, fue el desarrollo del back-end con Flask.

Inicialmente, estudié el funcionamiento y la estructura modular de Flask y me creé ficheros de tipo '.ipynb'. En estos ficheros Python desarrollé los modelos de goles esperados ( $xG$ ) y puntos esperados ( $xPoints$ ), así como sus visualizaciones y los distintos gráficos que fui creando con las distintas librerías de Python. Una vez desarrollados, pasaba a implementarlos en la carpeta del proyecto de Flask.

Para desarrollar el modelo de goles esperados ( $xG$ ) creé un '.csv' con los datos de Stastbomb de las 5 grandes ligas europeas (La Liga, Ligue 1, Bundesliga, Premier League, Serie A).

Este '.csv' contenía todos los datos de los eventos de todas las jornadas para todos los equipos. Partiendo de tener los datos en un '.csv' pude desarrollar el modelo de goles esperados.

Debido a que cada vez que se hacía una petición al *back-end* se debía leer un '.csv' de tamaño elevado y ejecutar el modelo, el tiempo de ejecución se elevaba a unos minutos. Para reducir ese tiempo, guardé los resultados finales en un 'JSON'. Este 'JSON' es mandado al *front-end* en Angular para mostrar información del modelo.

Para almacenar las imágenes de gráficos en Flask creé una carpeta llamada 'static'.

Para almacenar los vídeos en Flask creé una carpeta llamada 'uploads'.

El siguiente paso, fue crear las rutas en Flask para poder definir los *endpoints* a los cuáles iba a llamar desde Angular. Aquí, me enfrenté al error 'CORS' (Cross-origin resource sharing), el cual es un error que ocurre cuando se intenta hacer una solicitud http desde una aplicación que se ejecuta en una URL diferente a la del backend. Este problema surge debido a las políticas de seguridad implementadas en los navegadores web para prevenir solicitudes HTTP no autorizadas entre diferentes dominios.

Para su solución en Flask, bastó con instalar la librería '*flaskCors*' de CORS y con ella aplicar una configuración en mi aplicación para permitir solicitudes desde el frontend.

```
from flask import Flask
from flask_cors import CORS

def create_app():
    app = Flask(__name__)
    CORS(app, supports_credentials=True)
```

Figura 5.8: Código empleado para solucionar el problema CORS

### Generación de gráficos en Python usando datos de la librería StatsBomb

Para generar los gráficos en Python utilizando los datos de la librería *StatsBombPy* se han llevado a cabo los siguientes pasos:

1. Acceder a los datos de *StatsBombPy* y filtrar dichos datos en función del tipo de gráfico que se quiera crear (datos de jugadores, datos de equipos, datos de una fase eliminatoria). En concreto, se pueden obtener detalles de los partidos de una competición específica como puede ser el mundial de fútbol y filtrar los eventos relacionados con un equipo en particular o una ronda (en la aplicación los gráficos pertenecen a la final del mundial entre Argentina y Francia).
2. Los datos obtenidos se procesan para extraer la información relevante. Por ejemplo, se filtran los datos para obtener los de un jugador en específico.
3. Los gráficos han sido creados utilizando la librería *matplotlib*.

Los gráficos creados en el *backend* con Python han sido los siguientes:

- **Mapas de calor:** utilizando los datos de posición de un jugador se muestran las zonas del campo dónde estuvo más activo. Se utiliza la función '*kdeplot*' de *mplsoccer* para crear un gráfico de densidad que resalta las áreas más frecuentadas por el jugador. Con este tipo de gráfico podemos sacar como conclusiones si un jugador tuvo una participación mayormente ofensiva, defensiva o central.

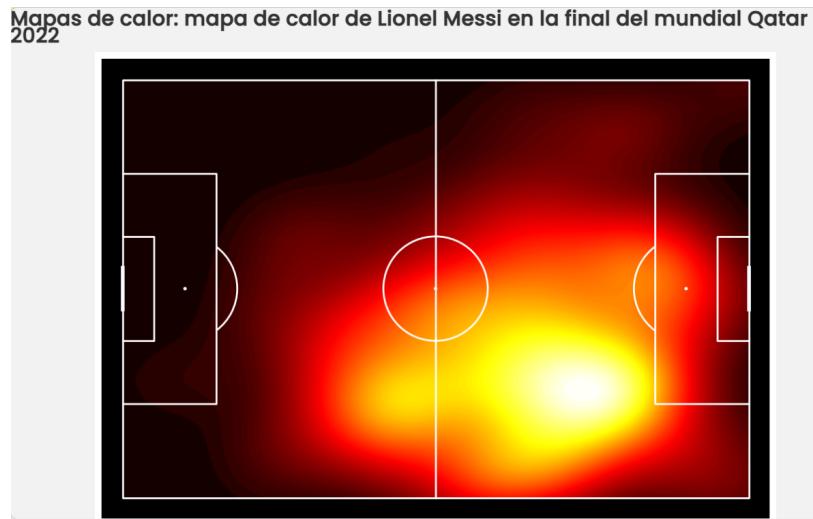


Figura 5.9: Ejemplo de mapa de calor generado en Python.

- **Mapas de posiciones:** utilizando los datos de las coordenadas en las que un jugador estuvo en el campo se puede crear un gráfico que refleja todas las coordenadas mostrando si un jugador fue más ofensivo o defensivo. Con este tipo de gráfico podemos comprobar si un jugador estuvo presente en muchas zonas del campo, es decir, estuvo presente en recuperaciones de balón, atacando, defendiendo.

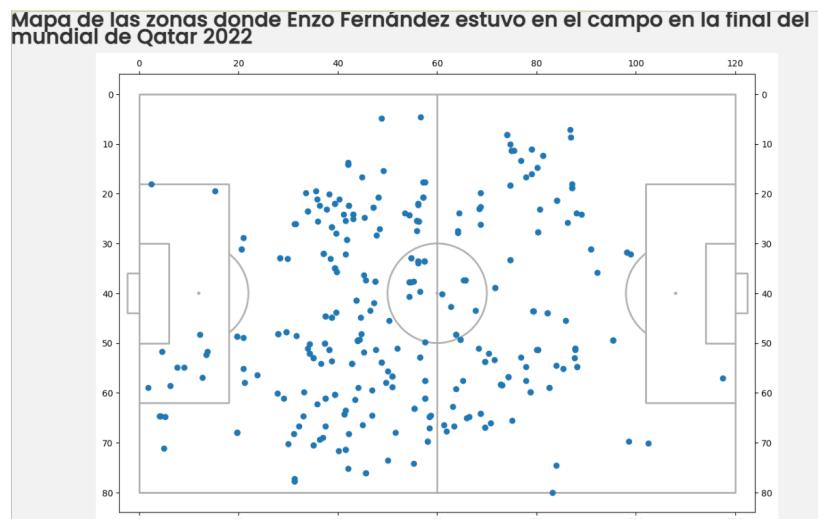


Figura 5.10: Ejemplo de mapa de posiciones de un jugador generado en Python.

- **Mapas de pases:** utilizando los datos de los pases de un jugador se pueden crear gráficos que muestren las trayectorias de los pases. Los pases en verde reflejan pases que tuvieron éxito en la entrega a otro jugador. En cambio, los pases en rojo son aquellos que no se entregaron correctamente. Este gráfico nos presenta una posible tendencia del jugador en el momento de hacer los pases y en qué zonas suele acertar o fallar los pases.



Figura 5.11: Ejemplo de mapa de pases de un jugador generado en Python.

- **Match momentum:** este gráfico utiliza los datos de los eventos de un partido como pases, conducciones de balón, disparos, goles. Se genera un gráfico que muestra de forma gráfica qué equipo dominó en cada momento del partido, observando si los goles llegaron en un momento bueno o malo del equipo.

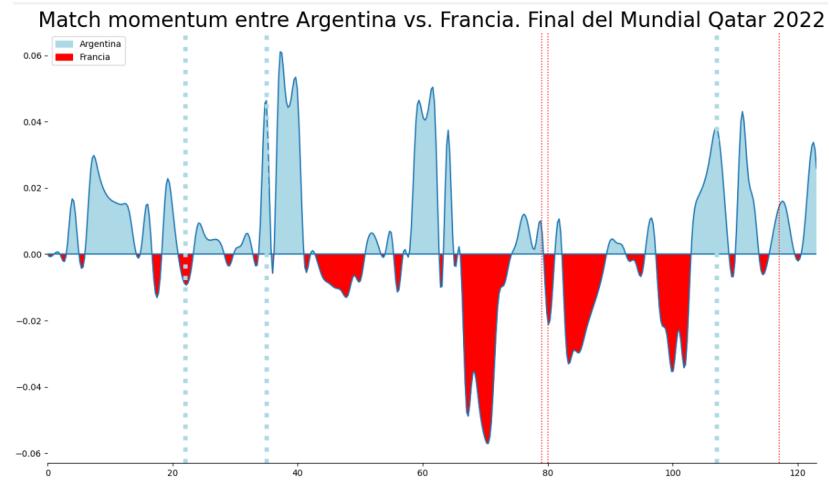


Figura 5.12: Ejemplo de gráfico *match momentum* generado en Python.

- **Portería:** con los datos de los disparos (acabaran en gol o no) de un partido se puede crear un gráfico que representa una portería dividida en secciones y muestra los lugares en los que un equipo disparó más o tuvo mayor probabilidad de gol.

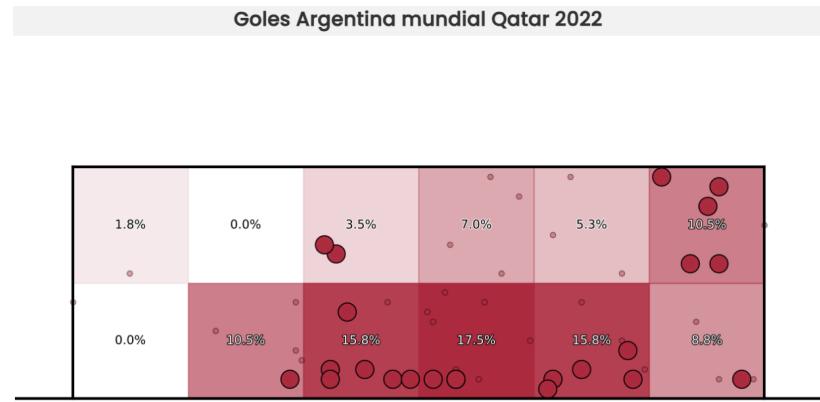


Figura 5.13: Ejemplo de gráfico de una portería que representa los disparos que tuvo Argentina en el mundial 2022.

## Integración de ImageAI

Siguiendo con el desarrollo del *back-end*, cabe destacar la implementación de imageAI, la cual no fue complicada gracias a la documentación de la librería. Una vez conseguí el vídeo procesado por el algoritmo, surgió el problema al intentar enviarlo al front-end.

Este error consistía en que el vídeo perdía las propiedades de su codec al ser procesado por el algoritmo. Para solucionar este error, utilicé la herramienta *Ffmpeg*, la cual se puede instalar en cualquier sistema operativo y se puede configurar en las variables de entorno del sistema para que pueda ser accesible a través de cualquier script en Python. Con esta herramienta, seguí el siguiente proceso para obtener el vídeo final:

- Procesar vídeo elegido
- Guardar resultado del procesamiento con imageAI en un fichero temporal en la carpeta 'uploads'.
- Utilizar *Ffmpeg* con el fichero temporal para generar el fichero de vídeo final.
- Obtener el fichero de vídeo final con el codec correcto y guardararlo en la carpeta uploads.

Posteriormente, realicé la conexión de Angular con Flask de esta funcionalidad. Desde Angular, un usuario hace un POST de un fichero '.mp4' a Flask. Este fichero es procesado en el back y devuelto al front para su visualización posterior. El usuario puede a través del reproductor utilizado descargar el vídeo.

## Arquitectura en local

La figura 5.14 muestra un esquema de la arquitectura de la aplicación a nivel local:

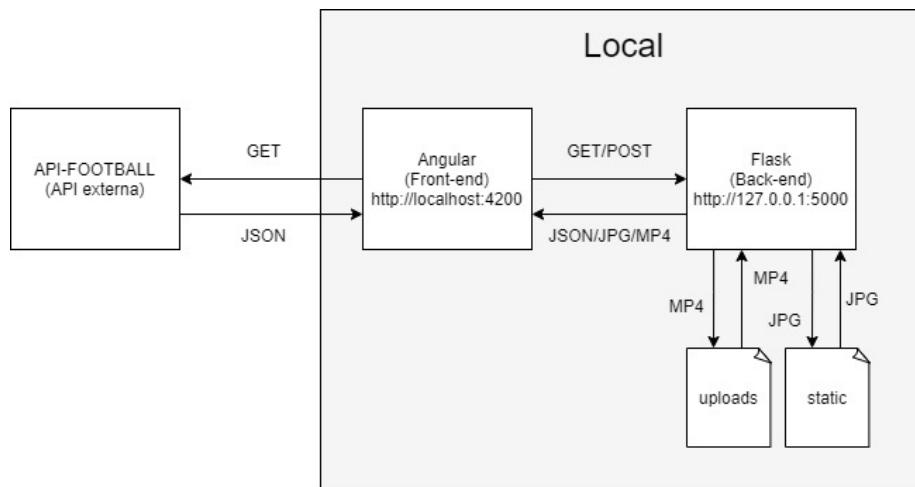


Figura 5.14: Arquitectura de la aplicación en local

En el esquema podemos ver como la aplicación en local se compone de Angular y Flask, los cuáles se pasan datos a través de peticiones GET y POST. Desde Angular se hacen peticiones GET a API-FOOTBALL que nos devuelve en ella respuesta un JSON. Finalmente, he representado las carpetas uploads y static que utilizo en el back para guardar los ficheros que se muestran en front-end.

## 5.5. Despliegue de la aplicación

Una vez la aplicación desarrolló completamente las funcionalidades pasó a desplegarse a producción para su uso. El *front-end* en Angular se desplegó en [Netlify](#), mientras que el *back-end* en Flask se desplegó en [Render](#). El despliegue en Netlify no supuso mayores problemas que seguir los pasos indicados. En cambio, el despliegue en Render tuvo algunos problemas debido a conflictos de dependencias en el fichero 'requirements.txt' que Render utilizaba para configurar el proyecto. Este error fue solucionado actualizando las dependencias que daban errores. Cabe destacar que la funcionalidad de los vídeos que consiste en que un usuario haga un POST al *back-end* de un vídeo para obtener de vuelta el mismo procesado por el algoritmo de imageAI no funciona debido a que los recursos que proporciona la versión gratuita de Render son escasos y no soporta la transferencia de archivos de gran volumen, además de permitir bajos recursos en uso de CPU y memoria RAM. Por ello, se aportará junto con el proyecto una máquina virtual con

instrucciones para poder utilizar esta funcionalidad y toda la aplicación en local.

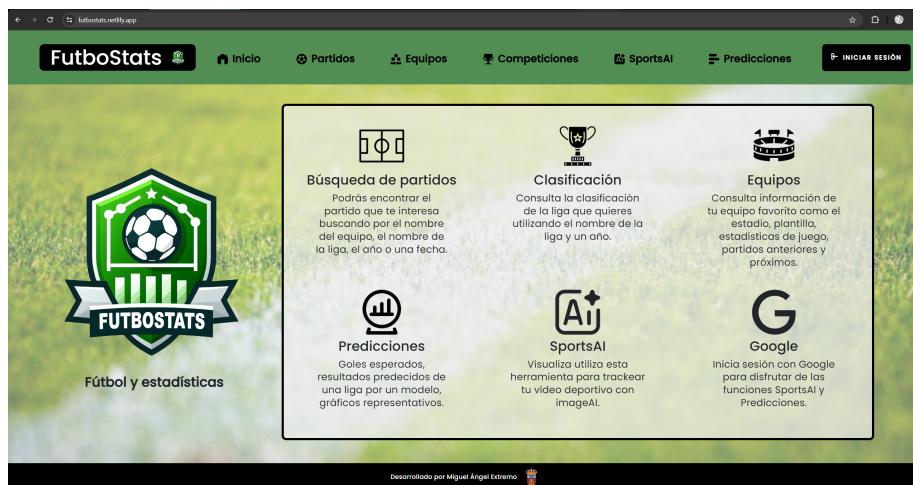


Figura 5.15: FutboStats desplegada en Netlify y Render

A continuación, muestro un esquema de la arquitectura de la aplicación a nivel de producción:

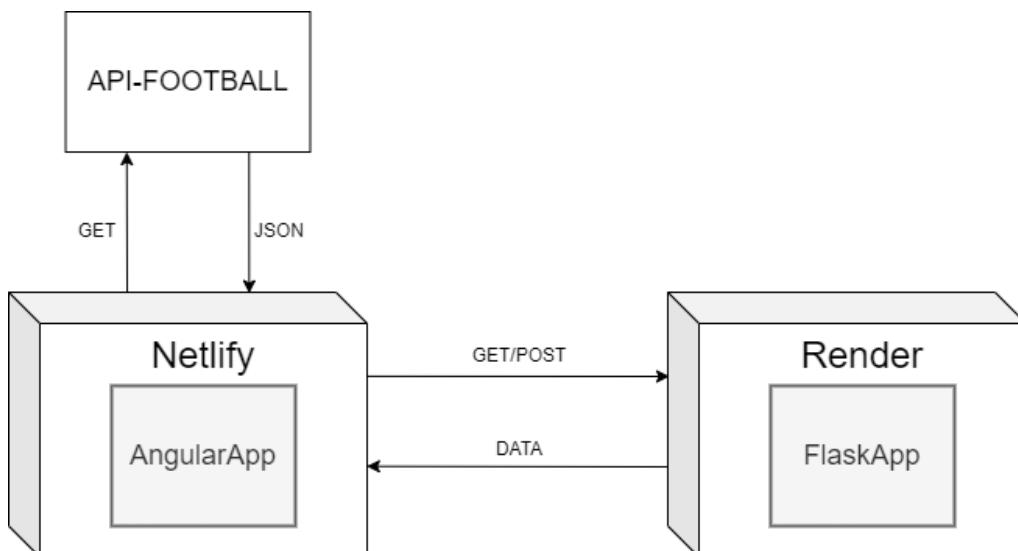


Figura 5.16: Arquitectura de la aplicación en producción.

## 5.6. Testing

Las pruebas que realicé en mi aplicación fueron utilizando la herramienta Postman.

He utilizado esta herramienta para probar los endpoints de las APIs utilizadas en el proyecto (API-FOOTBALL y API en Flask) antes de implementar los servicios que contendrían estas llamadas en Angular.

Utilicé Postman por las siguientes razones:

1. Postman ofrece una interfaz de usuario amigable que facilita la creación, configuración y prueba de solicitudes HTTP sin necesidad de escribir código adicional. Esto permite a los desarrolladores centrarse en la lógica de la API.
2. Es sencillo configurar todos los aspectos de una solicitud HTTP, como el método (GET, POST, PUT, DELETE), los headers (*api-key*, *api-host*), parámetros de consulta, el cuerpo de la solicitud, autenticación. Esto permite replicar exactamente las condiciones que la API esperaría en producción.
3. Postman permite probar rápidamente diferentes endpoints de una API y validar las respuestas. Se pueden ver las respuestas de la API en formato JSON, XML o texto plano, lo que facilita la comprobación de que los datos devueltos son correctos y están en el formato esperado.
4. Soporta la creación de entornos (desarrollo, prueba, producción), cada uno con sus propias variables de entorno como URL base, claves API.

### Ejemplo de uso de Postman: API-FOOTBALL

Las solicitudes a la API-FOOTBALL se estructuran en formato JSON y contienen parámetros como el nombre del equipo, nombre de la liga o fecha del partido. Estas solicitudes requieren incluir una clave de API llamada (*API key*) y un host de API llamado (*API host*) en los encabezados para autenticación. Cada *endpoint* de la API tiene parámetros establecidos, algunos de los cuales son opcionales. Para acceder a API-FOOTBALL, es necesario incluir en cada solicitud los siguientes encabezados:

- **API key:** Clave única proporcionada al registrarse en API-FOOTBALL, utilizada para autenticar las solicitudes.
- **API host:** Identificador del host de la API, que especifica el servidor al cual se envían las solicitudes.

Cada *endpoint* de la API-FOOTBALL acepta diferentes parámetros, que pueden ser obligatorios u opcionales. Algunos de los parámetros comunes incluyen:

- **Identificador del equipo:** identificador del equipo para el cual se desean obtener datos.
- **Identificador de la liga:** identificador de la liga para la cual se quieren obtener estadísticas o clasificaciones.
- **Fecha:** fecha del partido para el cual se desean obtener los resultados o detalles.
- **Season:** año de la temporada por la que se desea filtrar la información.

The screenshot shows a Postman request configuration for a GET request to <https://api-football-v1.p.rapidapi.com/v3/standings?league=140&season=2023>. The Headers tab is selected, showing three checked fields: Connection (keep-alive), X-RapidAPI-Key (your-api-key), and X-RapidAPI-Host (api-football-v1.p.rapidapi.com). The Body tab is selected, showing a JSON response structure. The response body is a JSON object with the following structure:

```

1  "get": "standings",
2   "parameters": {
3     "league": "140",
4     "season": "2023"
5   },
6   "errors": [],
7   "results": 1,
8   "paging": {
9     "current": 1,
10    "total": 1
11  },
12  "response": [
13    {
14      "league": {
15        "name": "La Liga"
16      }
17    }
18  ]

```

The status bar at the bottom indicates a successful 200 OK response with a time of 185 ms and a size of 11.05 KB.

Figura 5.17: Ejemplo de petición a API-FOOTBALL

Como podemos ver en la figura 5.17 se realiza una petición GET al endpoint '/standings' para obtener una clasificación en concreto. Le pasamos como parámetros el identificador de liga 140 (es el identificador de La Liga española) y la season 2023 (año de la clasificación). Para que la petición tenga éxito tenemos que establecer en los headers el campo 'X-RapidAPI-Key' con la Api-key que da API-FOOTBALL al registrarse y el campo

'X-RapidAPI-Host' con el host correspondiente que en mi caso es 'api-football-v1.p.rapidapi.com'. Como podemos observar en el *body* tenemos el resultado en JSON de la petición GET y el status de la petición (200 OK).

## Estructura de solicitudes y respuestas de la API en Flask

La API diseñada en Flask contiene varios endpoints. Algunos devuelven un JSON con la información del modelo de goles esperados, información de las tablas de los puntos esperados. En cambio, otros devuelven gráficos generados en Python utilizando los datos de *StatsbombPy*.

HTTP [http://127.0.0.1:5000/get\\_prediction\\_data](http://127.0.0.1:5000/get_prediction_data)

GET [http://127.0.0.1:5000/get\\_prediction\\_data](http://127.0.0.1:5000/get_prediction_data)

Params Authorization Headers (6) Body Pre-request Script

Query Params

Key
Key

Body Cookies Headers (10) Test Results

Pretty Raw Preview Visualize JSON ▾

```
1 "exactitud": 0.9034352744030163,
2 "precision": 0.6938775510204082,
3 "sensibilidad": 0.1969111969111969,
4 "especificidad": 0.9894266917293233,
5 "confusion_matrix": [
6     "0": {
7         "0": 4211,
8         "1": 416
9     },
10    "1": {
11        "0": 45,
12        "1": 102
13    }
14 ],
15 "goles Esperados": [
16     {"0": 0.0000000000000002}
17 ]
```

Figura 5.18: Ejemplo de petición a la API de Flask

Como podemos ver en la figura 5.18, se realiza una petición a un *endpoint* que devuelve un JSON con la información del modelo de goles esperados que se mostrará en el *front-end* en forma de tarjetas.

---

## 6. Trabajos relacionados

---

En este apartado, se van a comentar algunas de las referencias, trabajos y aplicaciones que he tenido en cuenta para desarrollar esta aplicación.

### Proyectos

Inicialmente, busqué proyectos que utilizarán la tecnología de imageAI para investigar y aprender más del funcionamiento acerca de esta herramienta. Tomé de referencia de partida un Trabajo de fin de grado de la Universidad de Sevilla llamado 'Detección de objetos en imágenes utilizando técnicas de aprendizaje profundo (Deep Learning)' [5].

En este trabajo de fin de grado, se ha utilizado ImageAI para el procesamiento de vídeos deportivos. Analiza los distintos algoritmos que proporciona ImageAI(YoloV3, Retinanet, TinyYOLOv3) y saca conclusiones a través de los resultados generados.

Mi proyecto utiliza imageAI y lo implementa en una aplicación para que un usuario común pueda utilizar esta herramienta sin ninguna complicación. Además, a diferencia de este trabajo de fin de grado, FutboStats conforma una aplicación completa de fútbol y estadísticas que brinda al usuario datos en tiempo y herramientas novedosas que se pueden aplicar en el fútbol como es ImageAI.

### Aplicaciones

Para desarrollar mi aplicación, tomé como referencia la aplicación móvil llamada 'One Football' [10].

OneFootball es una popular aplicación móvil dedicada a ofrecer noticias,

estadísticas y contenido multimedia sobre fútbol. Ofrece una amplia cobertura de ligas y competiciones de todo el mundo, incluyendo partidos en vivo, resultados, tablas de clasificaciones y actualizaciones de los equipos y jugadores.

Mi proyecto además de ser una web aplicación web implementa gráficos novedosos diseñados y creados en Python para representar estadísticas de partidos. Además, integra un modelo de goles esperados y de puntos esperados para informar al usuario de estadísticas más completas acerca de un equipo o una liga.

	TFG US	One Football	FutboStats
<b>Uso de ImageAI</b>	Sí	No	Sí
<b>ImageAI en web</b>	No	No	Sí
<b>Consulta de datos</b>	No	Sí	Sí
<b>Goles esperados (xG)</b>	No	No	Sí
<b>Puntos esperados (xPoints)</b>	No	No	Sí
<b>Estadísticas con Python</b>	No	No	Sí
<b>Aplicación web</b>	No	Sí	Sí
<b>Adaptación a móvil</b>	No	Sí	Sí

Tabla 6.1: Comparativa de funcionalidades entre el TFG de la Universidad de Sevilla, One Football y FutboStats

---

## 7. Conclusiones y Líneas de trabajo futuras

---

En este apartado se exponen las conclusiones del proyecto y las líneas de trabajo futuras mediante las cuales puede ir escalando el proyecto.

### Conclusiones

Tras finalizar el desarrollo del proyecto propuesto he sacado las siguientes conclusiones:

- He cumplido el objetivo inicial del proyecto gratamente. Se ha desarrollado una aplicación web desde cero e interactiva para cualquier usuario que desee consultar datos y estadísticas variadas sobre fútbol.
- Utilizar Angular para el desarrollo *front-end* ha supuesto ventajas e inconvenientes. No han surgido problemas con las dependencias en Angular utilizadas y ha supuesto, es un *framework* escalable bien estructurado, se pueden reutilizar los componentes creado fácilmente. Algunas de sus desventajas se basan en su curva de aprendizaje al iniciar el proyecto, ya que, ha supuesto un reto aprender Angular en un periodo reducido de tiempo, además de las extensas funcionalidad que contiene el *framework*.
- El proyecto ha abarcado conocimientos sobre librerías de Python utilizadas durante el grado como pueden ser Pandas, Matplotlib, SciPy, modelos de regresión, etc.
- El proyecto ha abarcado una gran cantidad de conocimientos que desconocía en el inicio. Ha supuesto un reto aprender Angular y

aplicarlo en un proyecto real tan pronto, pero gracias al esfuerzo y dedicación he conseguido desarrollar con éxito una aplicación usable por cualquier usuario.

- En este proyecto he utilizado un gran número de herramientas y tecnologías como ImageAI que me han ayudado a implementar funcionalidades nuevas a la aplicación y con ello mejorar el producto final que se ofrece al usuario.
- Se ha desarrollado el proyecto en el plazo establecido de manera satisfactoria.
- Gracias a esta aplicación he aprendido a utilizar la documentación de una API y como aplicarlo en un proyecto real. Además, he aprendido a utilizar la herramienta Postman, la cual es frecuentemente usada en el mundo laboral para realizar llamadas reales a los endpoints de una API y comprobar si la respuesta es la esperada o no. Esto suponía ahorrar tiempo a la hora de implementar correctamente las funcionalidades en la aplicación.

## Líneas de trabajo futuras

A continuación, se van a detallar algunas de las mejoras y líneas de trabajo futuras que tendrá este proyecto:

- Implementación de nuevas funcionalidades con API-FOOTBALL. Debido a la gran cantidad de datos en tiempo real que proporciona API-Football, se implementarán nuevas funcionalidades en la aplicación que mejoran la experiencia al usuario y enriquezcan la aplicación.
- Creación de un pequeño buzón de sugerencias al usuario para que nos haga llegar un *feedback* y como podríamos mejorar la aplicación.
- Actualizar las estadísticas en la parte de predicciones regularmente con datos de partidos y competiciones recientes.
- Internacionalizar la aplicación implementando traducciones para que sea posible utilizar la aplicación en varios idiomas.

Estas son algunas de las mejoras que se integrarán en la aplicación en el futuro, haciendo crecer a la aplicación para que llegue a un mayor número de usuarios.

---

## Bibliografía

---

- [1] Amazon Web Services, Inc. ¿qué es una api de restful? <https://aws.amazon.com/es/what-is/restful-api/>, 2023. Último acceso: 23 de mayo de 2024.
- [2] Alejandro Arroyo. ‘puntos esperados’ (xp): los que dan crédito al merecimiento. <https://www.driblab.com/es/actualidad/puntos-esperados-xp-la-metrica-que-da-credito-al-merecimiento/>, 2021. [Internet; accedido 17-mayo-2024].
- [3] Guillermo Baches. Goles esperados (xg) : Cómo han revolucionado la manera de analizar el rendimiento de los equipos y jugadores de fútbol actual. <https://guillembaches.medium.com/goles-esperados-xg-c%C3%B3mo-han-revolucionado-la-manera-de-analizar-el-rendimiento-de-los-equipo-cc9acc0aa548>, 2023. [Internet; accedido 17-mayo-2024].
- [4] Esri. Detección de objetos. <https://pro.arcgis.com/es/pro-app/latest/tool-reference/image-analyst/object-detection.htm>, 2022. [Internet; accedido 18-mayo-2024].
- [5] Daniel Estévez Trigo. Detección de objetos en imágenes utilizando técnicas de aprendizaje profundo (deep learning). <https://idus.us.es/handle/11441/127376>, 2021. [Internet; accedido por primera vez 20-febrero-2024].
- [6] API Football. Api football. <https://www.api-football.com/>, 2024. [Internet; accedido 20-mayo-2024].

- [7] glenn jocher. Yolov3, yolov3-ultralytics, y yolov3u. <https://docs.ultralytics.com/models/yolov3/>, 2024. [Internet; accedido 18-mayo-2024].
- [8] Fernando Herrera. Angular: De cero a experto. <https://www.udemy.com/course/angular-fernando-herrera/>, Última actualización: 3/2024. [Internet; accedido 20-mayo-2024].
- [9] Moses Olafenwa. Official english documentation for imageai. <https://imageai.readthedocs.io/en/latest/index.html>, 2022. [Internet; accedido 18-mayo-2024].
- [10] OneFootball. Onefootball: Todo sobre fútbol. <https://onefootball.com/es/inicio>, 2024. [Internet; accedido 21-mayo-2024].
- [11] RapidApi. Rapidapi. <https://rapidapi.com>, 2024. Último acceso: 23 de mayo de 2024.
- [12] StatsBomb. Statsbomb. <https://statsbomb.com/es/>, 2024. Último acceso: 23 de mayo de 2024.
- [13] StatsBomb. ¿qué son los goles esperados? <https://statsbomb.com/es/metricas-de-futbol/que-es-el-xg/#:~:text=En%20t%C3%A9rminos%20b%C3%A1sicos%2C%20el%20xG,escala%20de%200%20a%201.>, 2024. [Internet; accedido 17-mayo-2024].
- [14] Wikipedia. Distribución de poisson. [https://es.wikipedia.org/wiki/Distribuci%C3%B3n\\_de\\_Poisson](https://es.wikipedia.org/wiki/Distribuci%C3%B3n_de_Poisson), 2024. [Internet; accedido 17-mayo-2024].
- [15] Wikipedia. Kanban. <https://es.wikipedia.org/wiki/Kanban>, 2024. [Internet; accedido 15-mayo-2024].
- [16] Wikipedia. Regresión logística. [https://es.wikipedia.org/wiki/Regresi%C3%B3n\\_log%C3%ADstica](https://es.wikipedia.org/wiki/Regresi%C3%B3n_log%C3%ADstica), 2024. [Internet; accedido 17-mayo-2024].
- [17] Wikipedia. Scrum (desarrollo de software). [https://es.wikipedia.org/wiki/Scrum\\_\(desarrollo\\_de\\_software\)](https://es.wikipedia.org/wiki/Scrum_(desarrollo_de_software)), 2024. [Internet; accedido 15-mayo-2024].