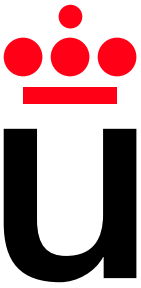


Universidad
Rey Juan Carlos

Escuela Técnica Superior
de Ingeniería Informática

**Grado en Ingeniería Informática**

**Curso 2023-2024**

**Trabajo Fin de Grado**

# DESARROLLO DE UN SISTEMA DE SOPORTE A LA DECISIÓN PARA JUEGOS DE GESTIÓN DEPORTIVA

**Autor: Antonio Cuadrado Álvarez**

**Tutor: Eduardo García Pardo**
**Sergio Cavero Díaz**

# Agradecimientos

A mis compañeros de la universidad, por su ayuda técnica cuando era necesaria y por soportarme cuando me encontraba agobiado.

A mis amigos de Fuensalida, por la ayuda y motivarme a seguir adelante.

A mi familia, por el gran apoyo y saber comprenderme en los momentos más difíciles.

A Ángela, por ser la gran amiga que es y ayudarme a no hacer una montaña de un grano de arena.

A Natalia, mi novia, ya que su cariño y su apoyo incondicional han sido fundamentales para seguir adelante.

Y, por último, pero no menos importante, a mis tutores, Eduardo García Pardo y Sergio Cavero Díaz, por haberme permitido hacer este Trabajo Fin de Grado con ellos y no haberse rendido conmigo.

# Resumen

El fútbol, a lo largo del tiempo, ha crecido en importancia dentro del mundo deportivo y, con él, los juegos de *fantasy manager*, los cuales se basan en la gestión de un equipo ficticio con los jugadores reales de una competición, con el objetivo de obtener la máxima puntuación. Este crecimiento en el fútbol ha resultado en el uso de Inteligencia Artificial para diferentes herramientas, algo que aún no se aplica de gran manera a los juegos de *fantasy manager*.

Este Trabajo Fin de Grado (TFG) surge como respuesta a esta ausencia de herramientas que, mediante el uso de técnicas algorítmicas, pongan a disposición de los usuarios de juegos *fantasy manager* un asistente que les ayude a tomar ciertas decisiones dentro de estos juegos.

En concreto, en este TFG y mediante el uso de los datos de fútbol, se pretende crear un sistema de apoyo para el usuario de los *fantasy manager* que mediante una predicción de la puntuación de los jugadores determine quiénes son los mejores once jugadores a utilizar. Para ello se va a modelar el problema como un problema de optimización y este se va a abordar mediante técnicas clásicas, en concreto, se propone la utilización de un algoritmo enumerativo, denominado *backtracking* complementado con el uso de la técnica de podas en el árbol de exploración.

Además, se lleva a cabo un análisis de las soluciones encontradas para determinar si el tiempo empleado en la búsqueda de la solución óptima es adecuado para el tipo de herramienta a desarrollar.

Por último, en este TFG, se ha desarrollado una interfaz gráfica sencilla que permite la ejecución del algoritmo implementado con cierto grado de parametrización, como por ejemplo el tiempo máximo de ejecución.

**Palabras clave**: *backtracking*, mánager deportivo, algoritmia, problema de optimización, interfaz gráfica.

# Índice de contenidos

**Índice de tablas** **VIII**

**Índice de figuras** **X**

**Acrónimos** **X**

**1. Introducción** **1**
1.1. Contextualización y conceptos previos . . . . . 1
1.1.1. Breve historia del fútbol . . . . . 1
1.1.2. Juegos *fantasy manager* . . . . . 3
1.1.3. Aplicaciones de inteligencia artificial en el contexto del fútbol y juegos *fantasy manager* . . . . . 5
1.2. Motivación . . . . . 6
1.3. Estructura del proyecto . . . . . 7

**2. Objetivos** **8**
2.1. Objetivo general . . . . . 8
2.2. Objetivos específicos . . . . . 8

**3. Metodología** **10**
3.1. Metodologías de desarrollo . . . . . 10
3.2. Metodología escogida . . . . . 13
3.3. Metodología investigación . . . . . 19

**4. Preparación de los datos** **21**
4.1. Normalización de los datos . . . . . 21
4.2. Análisis de la correlación . . . . . 23
4.3. Asignación de puntos . . . . . 25

**5. Formalización del problema y propuesta algorítmica** **28**
5.1. Formalización del problema . . . . . 28
5.2. Resolución del problema . . . . . 31

**6. Resultados** **36**

6.1. Análisis resultados . . . . . 36
6.2. Interfaces aplicación usuario . . . . . 41

**7. Conclusiones y trabajos futuros** **46**
7.1. Conclusiones . . . . . 46
7.2. Trabajos futuros . . . . . 47

**Bibliografía** **47**

# Índice de tablas

3.1. Historia de Usuario sobre normalización de datos totales de los jugadores. . . . . 14
3.2. Historia de Usuario sobre normalización de datos por partido de los jugadores. . . . . 15
3.3. Historia de Usuario sobre correlación de datos sin normalizar. . . 15
3.4. Historia de Usuario sobre correlación de datos normalizados. . . . 16
3.5. Historia de Usuario sobre sistema de asignación de puntos. . . . . 16
3.6. Historia de Usuario sobre comprobación del sistema de asignación. 17
3.7. Historia de Usuario sobre el cálculo de la alineación óptima. . . . 17

4.1. Correlación de datos no normalizados de los 5 delanteros con más puntos. . . . . 24
4.2. Correlación de datos normalizados de los 5 delanteros con más puntos. . . . . 25
4.3. Tabla mostrando la correlación de los puntos asignados por el sistema creado y los puntos Biwenger. . . . . 27

# Índice de figuras

1.1. Ebenezer Cobb Morley. . . . . 2
1.2. Copa de la Coronación (Actual Copa del Rey). . . . . 2
1.3. Riccardo Albini. . . . . 3
1.4. Imágenes de logos de las diferentes aplicaciones. . . . . 4
1.5. Imágenes de alineaciones en las diferentes aplicaciones. . . . . 4
1.6. Imágenes de mercados en las diferentes aplicaciones. . . . . 5

3.1. Diagrama de Gantt de las historias de usuario. . . . . 18
3.2. Diagrama de flujo del proceso de metodología de investigación. . . 20

4.1. Ejemplo de algunos de los datos antes de ser normalizados. . . . . 22
4.2. Ejemplo de algunos de los datos después de ser normalizados. . . 23

5.1. Ejemplo de recorrido de algoritmo genérico de *backtracking*. . . . . 31
5.2. Ejemplo de árbol de decisión para el algoritmo con poda. . . . . 34
5.3. Ejemplo de árbol de decisión para el algoritmo sin poda. . . . . 35

6.1. Gráfica con los tiempos para la alineación 4-3-3. . . . . 38
6.2. Gráfica con los tiempos para la alineación 5-4-1. . . . . 38
6.3. Gráfica con los tiempos para la alineación 4-3-3 cambiando los presupuestos. . . . . 39
6.4. Gráfica con los tiempos para la alineación 5-4-1 cambiando los presupuestos. . . . . 39
6.5. Gráfica que compara la calidad de los puntos obtenidos según el tiempo. . . . . 40
6.6. Gráfica que compara los resultados obtenidos en diferentes jornadas y con diferentes opciones. . . . . 41
6.7. Interfaz de la aplicación para introducir los datos. . . . . 42
6.8. Diferentes alineaciones a usar en la aplicación. . . . . 42
6.9. Interfaz que muestra la solución obtenida. . . . . 43
6.10. Interfaz que muestra los parámetros del primer caso de uso. . . . 44
6.11. Interfaz que muestra la solución obtenida en el primer caso de uso. 44
6.12. Interfaz que muestra los parámetros del segundo caso de uso. . . . 45
6.13. Interfaz que muestra la solución obtenida en el segundo caso de uso. 45

# Acrónimos

**CSV** *Comma Separated Values*. 24

**FIFA** *Fédération Internationale de Football Association*. 1

**IA** Inteligencia Artificial. 5, 6

**JSON** *JavaScript Object Notation*. 23

**LLM** *Large Language Models*. 5

**RUP** *Rational Unified Process*. 11

**TFG** Trabajo Fin de Grado. 1, 6–8, 47, V

**VAR** *Video Assisted Referee*. 5

# 1 Introducción

Este capítulo se divide en tres secciones: la Sección 1.1 introduce los conceptos básicos relacionados con el fútbol, los juegos *fantasy manager* y la inteligencia artificial y sus usos, la Sección 1.2 presenta la motivación del proyecto y, por último, la Sección 1.3 recoge la estructura del trabajo.

## 1.1. Contextualización y conceptos previos

Esta sección presenta los conceptos y definiciones necesarios para comprender esta memoria. El Trabajo Fin de Grado (TFG) se centra en tres áreas principales: fútbol, juegos *fantasy manager* y la inteligencia artificial.

### 1.1.1. Breve historia del fútbol

En cuanto al origen del fútbol, aunque haya pruebas que indican que anteriormente ya se practicaban deportes de pelota similares a este en sitios como China o la actual Sudamérica, el origen oficial del fútbol tal y como se conoce ahora data de 1863 con la creación de la *Football Association* en Inglaterra [1]. Esta fue creada por Ebenezer Cobb Morley y cuya foto se puede ver en la Figura 1.1, conocido como el padre del fútbol, posteriormente, el primer partido oficial se disputó el 19 de diciembre de ese mismo año. Desde entonces el fútbol ha ido evolucionando con el paso del tiempo y ciertos acontecimientos importantes como la creación de la FIFA (`https://www.fifa.com/es/`) acrónimo de *Fédération Internationale de Football Association*, que es la máxima autoridad del fútbol a

Figura 1.1: Ebenezer Cobb Morley.

escala global. Fundada en 1904, esta institución se encarga de supervisar y dirigir las federaciones nacionales de este deporte, unificando las reglas y organizando campeonatos internacionales como la famosísima Copa Mundial desde 1930 [2] [3].

En España, este deporte llegó de la mano de inmigrantes ingleses en Huelva donde se disputaron los primeros partidos cerca de 1870, para después crear el primer equipo reconocido legalmente de España, el Cricket y Foot-Ball (Club) de Madrid, aunque el decano fue el actual Real Club Recreativo de Huelva. Al igual que en el fútbol mundial, en España también hubo ciertos acontecimientos importantes, como la realización en 1902 del Concurso Madrid de Foot-Ball Association, lo que dio lugar a la actual Copa Del Rey, cuyo trofeo inicial se puede ver en la Figura 1.2, la creación en 1929 del Campeonato Nacional de Liga Profesional o el uso de dorsales en la temporada 1948-1949 [4].

Figura 1.2: Copa de la Coronación (Actual Copa del Rey).

### 1.1.2. Juegos *fantasy manager*

Los juegos *fantasy manager* de fútbol fueron creados en 1990 en Italia por *Riccardo Albini* quien aparece en la Figura 1.3, aunque esto no fue una idea original suya, ya que tomo inspiración de juegos creados previamente en Estados Unidos, como el sitio web para hockey de fantasía creado en 1995 por *Molson Breweries* [5].

Aunque este tampoco fue el primer juego *fantasy manager* originalmente creado, ya que el que se tiene en cuenta como el origen de los juegos *fantasy manager* fue el golf de fantasía en los años 50 creado por Wilfred Winkenbach, quien también creo el baloncesto de fantasía en 1960 o el fútbol americano de fantasía en 1962 [5].

Figura 1.3: Riccardo Albini.

La principal llegada de estos juegos a España se realizó principalmente por parte de Comunio (`https://www.comunio.es`) un juego *fantasy manager* de fútbol creado en Alemania en el que se podían crear competiciones para los usuarios con ligas como La Liga (España), Premier League (Inglaterra), Bundesliga (Alemania), etc. Todo esto provocó la creación de otros juegos *fantasy manager* en España como Biwenger o La Liga Fantasy [5].

Entre los más importantes juegos *fantasy manager* que se pueden encontrar en España se encuentran, el ya mencionado Comunio, Biwenger (`https://biwenger.as.com/`) o LaLiga Fantasy (`https://laligafantasy.relevo.com/`).

Estas son algunas de sus diferencias mostradas en las imagenes:

Las Figuras 1.4a, 1.4b y 1.4c presentan los diferentes logos para cada una de las aplicaciones.

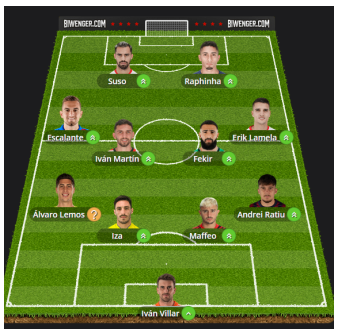

(a) Logo de Biwenger.


(b) Logo de Comunio.

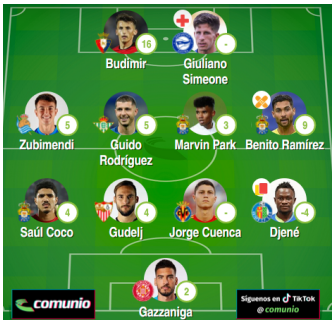

(c) Logo de LaLiga Fantasy.

Figura 1.4: Imágenes de logos de las diferentes aplicaciones.

Las Figuras 1.5a, 1.5b y 1.5c muestran cómo se ven las alineaciones de los diferentes juegos, las cuales deben tener obligatoriamente un portero, pero el resto de número de jugadores por posición puede ir variando, siendo las más comunes 4-4-3, 4-4-2, 3-4-3, etc. Concretamente la Figura 1.5a se ha obtenido de la aplicación Biwenger, la Figura 1.5b se ha obtenido de la aplicación Comunio y la Figura 1.5c se ha obtenido de la aplicación La Liga Fantasy.




(a) Alineación en Biwenger.

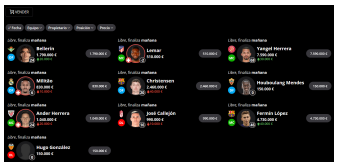



(b) Alineación en Comunio.


(c) Alineación en LaLiga Fantasy.

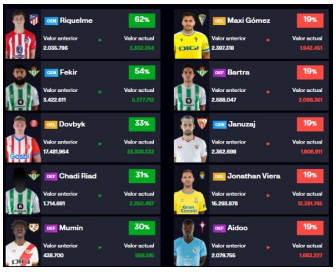
Figura 1.5: Imágenes de alineaciones en las diferentes aplicaciones.

Las Figuras 1.6a, 1.6b y 1.6c muestran cómo se ven los mercados de los diferentes juegos. En ellos se debe pujar un precio por el jugador deseado, y pasadas 24 horas el usuario que haya pujado la cantidad más alta se lleva al jugador. Concretamente la Figura 1.6a se ha obtenido de la aplicación Biwenger, la Figura 1.6b se ha obtenido de la aplicación Comunio y la Figura 1.6c se ha obtenido de la aplicación La Liga Fantasy.



(a) Mercado en Biwenger.

(b) Mercado en Comunio.



(c) Mercado en LaLiga Fantasy.

Figura 1.6: Imágenes de mercados en las diferentes aplicaciones.

### 1.1.3. Aplicaciones de inteligencia artificial en el contexto del fútbol y juegos *fantasy manager*

El origen de la Inteligencia Artificial (IA) se data en los años 50. Alan Turing, realizó la pregunta de si las maquinas podían pensar, y así se definió la denominada prueba de Turing, en la que una persona introducía unas preguntas en un ordenador y tenía que determinar si las respuestas eran de una persona o de un ordenador. Sin embargo, esta prueba no fue determinante en el momento que se planteó.

Algunos de los avances más significativos desde entonces fueron los siguientes: en 1952, Arthur Samuel fue el creador de un *software* que era capaz de jugar al ajedrez por voluntad propia; en los años 70 surgen los primeros *Large Language Models* (LLM), las cuales son maquinas que mediante una entrada de texto predicen la respuesta que corresponde a ese texto, mediante Eliza, considerada la primera *chatbot*; en los 90, surgen los agentes inteligentes, los cuales podían recibir percepciones de su entorno, procesar estas percepciones y actuar según la manera adecuada o en 2014, en la universidad de Reading, un ordenador paso de manera exitosa la prueba de Turing por primera vez [6].

A día de hoy, existen modelos LLM muy populares con los que es posible mantener conversaciones de manera fluida como si de un humano se tratara, por ejemplo, ChatGPT (`https://chatgpt.com/`).

En el lado del fútbol, una de las más importantes creaciones relacionadas con el uso de la inteligencia artificial es el origen del VAR (del inglés *Video Assisted Referee*), lo cual por ejemplo, mediante el uso de cámaras especializadas y puntos corporales de los jugadores calcula las diferentes posiciones para determinar si es un fuera de juego o no, entre otras aplicaciones, provocó que ya no se dependiese solamente del ojo humano a la hora de tomar decisiones, sin embargo, esta no es su única aplicación.

Una de las aplicaciones de la IA al fútbol que se pueden encontrar es el desarrollo de algoritmos para analizar el rendimiento de un jugador mediante diferentes datos recogidos de este. El análisis de estos datos permitiría determinar si el jugador es interesante o no para un equipo determinado.

Por otro lado, también se tiene la predicción de resultados, donde basándose en los datos históricos de un equipo se pueden realizar algoritmos que calculan cuáles son los puntos fuertes y débiles de este, cuál es su desempeño contra ciertos equipos, cuál es la probabilidad de anotar gol según cada situación, etc.

Otra aplicación que se puede encontrar en el mismo ámbito es la mejora en los entrenamientos ya que mediante un algoritmo que prediga cuáles son los puntos débiles de un jugador permitiría determinar qué reforzar de cara a un partido. También se puede utilizar la inteligencia artificial para el descubrimiento de talentos, ya que, con los datos de los jugadores, la inteligencia artificial puede ayudar a los ojeadores a valorar el rendimiento de estos y así fichar potenciales estrellas [7].

Con respecto a los juegos *fantasy manager* algunos de sus usos son, por ejemplo, algoritmos que ayudan a decidir que jugador elegir para poner en el 11 titular o a elegir que jugador fichar. Entre las aplicaciones que más destacan se encuentra, por un lado, Automanager [8] que tiene como objetivo ayudar a los jugadores de los juegos *fantasy manager* más importantes de España a tomar decisiones como qué jugador fichar o que jugador usar, lo cual se hace mediante un algoritmo que calcula las posibilidades de éxito de un jugador enfrentándolo a otros.

Otra aplicación que usa la inteligencia artificial para este tipo de juegos es Olocip [9], aplicación que se ha integrado en Biwenger con el mismo objetivo de calcular el rendimiento de los jugadores.

Dentro de todas las aplicaciones de la inteligencia artificial, este TFG se centra en crear un sistema que mediante el uso de ciertos algoritmos sea capaz de ayudar al jugador a tomar las decisiones necesarias en estos juegos.

## 1.2. Motivación

La inteligencia artificial ha experimentado un crecimiento significativo en los últimos años gracias a aplicaciones generativas como ChatGPT. Sin embargo, en el ámbito del fútbol, hay una falta de sistemas que utilicen una técnica algorítmica enumerativa para la toma de decisiones. Por lo tanto, la motivación principal para este TFG es el desarrollo un sistema que utilice una técnica algorítmica enumerativa como apoyo en la toma de decisiones tanto en el fútbol como en los juegos *fantasy manager*.

Además, este TFG es un requisito fundamental para la finalización de los

estudios en el Doble Grado de Ingeniería Informática e Ingeniería del Software. La culminación de la formación académica recibida se combina con la pasión del autor por el fútbol y el deseo de contribuir al avance de la ayuda a la toma de decisiones en este contexto.

Por otro lado, el mundo del fútbol es algo de gran interés para el autor de este TFG lo cual le motiva a investigar más sobre este tema y aprender sobre los conceptos más técnicos de este deporte.

Por último, resulta un aliciente tener la oportunidad de aprender más sobre temas como las técnicas algorítmicas enumerativas aplicadas al deporte, y explorar cómo esta técnicas pueden transformar la forma en que se toman decisiones en el fútbol.

## 1.3. Estructura del proyecto

En el Capítulo 1 se ha realizado una breve introducción a los conceptos que se tratan a lo largo de esta memoria, se presenta el contexto y la motivación del trabajo, además se introduce información necesaria para facilitar la comprensión de los siguientes capítulos. En el Capítulo 2, se detallan los objetivos planteados en este TFG, tanto principales como secundarios. A continuación, en el Capítulo 3, se presenta el proceso de la metodología que se ha seguido durante el desarrollo del proyecto. En el Capítulo 4, se explica el proceso que se ha seguido para normalizar los datos desde su obtención en la base de datos hasta su uso. En el Capítulo 5, se realiza un planteamiento del problema a resolver y la resolución de este. Seguidamente, en el Capítulo 6, se presentan los resultados del trabajo. Por último, en el Capítulo 7, se recogen las conclusiones y se plantean diferentes caminos de continuación para trabajos futuros.

# 2
# Objetivos

En este capítulo, se presentan los objetivos establecidos para este TFG. La Sección 2.1 describe el objetivo principal, mientras que en la Sección 2.2 se definen una serie de objetivos específicos que son necesarios para alcanzar dicho objetivo principal.

## 2.1. Objetivo general

El objetivo principal de este trabajo consiste en desarrollar una aplicación que brinde apoyo a los jugadores de juegos *fantasy manager* a la hora de tomar la decisión de qué jugadores poner en la alineación mediante un sistema basado en algoritmos de búsqueda enumerativos.

## 2.2. Objetivos específicos

Para alcanzar el objetivo principal anteriormente especificado es necesario alcanzar una serie de objetivos específicos que se enumeran a continuación:

1. Investigar sobre la historia del fútbol y de los juegos *fantasy manager*.
2. Investigar sobre el uso de la inteligencia artificial y otras técnicas algorítmicas en el fútbol y en los juegos *fantasy manager*.

3. Preparar los datos para que todos estos se encuentren en un formato correcto.

4. Formalizar el problema como un problema de optimización.

5. Desarrollar un algoritmo que implemente este problema de optimización capaz de encontrar soluciones de calidad en un tiempo razonable.

6. Analizar las diferentes soluciones que se obtengan del algoritmo.

7. Realizar una aplicación para ordenador que facilite el uso de estos algoritmos de ayuda en la decisión al jugador.

# 3

# Metodología

En este tercer capítulo se presentan las diferentes metodologías de desarrollo existentes (Sección 3.1) y la utilizada finalmente en la Sección 3.2. Además, en la Sección 3.3 se presenta la metodología de investigación que ha guiado parte del proceso de construcción de este trabajo.

## 3.1. Metodologías de desarrollo

Las metodologías de desarrollo en programación se refieren al conjunto de procedimientos y estrategias que se siguen para organizar el trabajo del equipo de desarrollo. Estas metodologías buscan optimizar la eficacia y la productividad del equipo, asegurando que el proyecto avance de manera ordenada y sistemática. Esto incluye la planificación de las tareas, la asignación de roles y responsabilidades, la gestión del tiempo y los recursos, y la implementación de mecanismos de control de calidad y revisión.

Dentro de las distintas metodologías se pueden distinguir dos tipos: las tradicionales / clásicas y las ágiles. Las metodologías clásicas destacan por ser bastante rígidas a la hora de definir las necesidades y ser poco flexibles a la hora de introducir cambios, habiendo incluso casos donde es imposible introducir cambios. Algunas de estas metodologías clásicas son las siguientes:

- Cascada: esta metodología funciona por etapas de tal manera que cuando se pasa a una etapa no se puede volver a la previa, por lo cual se recomienda

revisar toda la etapa antes de pasar a la siguiente [10].

Estas etapas son: análisis, diseño, implementación, prueba y entrega.

Una gran desventaja de esta metodología es que si en alguna de las fases se comete algún error no es posible volver atrás.

- *Rational Unified Process* (RUP): está dividida en 4 fases, la primera de ellas se conoce como inicio y busca establecer los requisitos con el cliente, la siguiente es la fase de elaboración, que tiene como objetivo refinar los requisitos definidos previamente y seleccionar una arquitectura para el proyecto, algo que es de gran importancia en esta metodología [11].

  Después, se encuentra la fase de elaboración, donde se empieza a desarrollar el sistema. Finalmente se encuentra la fase de transición, donde se pasa el proyecto de su desarrollo a su entrega al cliente.

- Espiral: está dividida en ciclos, donde dentro de cada ciclo podemos encontrar las siguientes fases:

  - Determinación de objetivos, donde se establecen los objetivos a cumplir durante el ciclo.
  - Evaluación de alternativas, donde se revisan los objetivos de la fase anterior y las diferentes alternativas para cada uno.
  - Desarrollo, donde se implementan las soluciones elegidas para objetivo.
  - Planificación del próximo ciclo, donde se revisa el ciclo actual, y se prepara el siguiente [12].

- Incremental y prototipado: estas metodologías son similares en cierta parte ya que ambas funcionan mediante la introducción de pequeñas funcionalidades, en este caso en la metodología incremental se hace desde cero, mientras que en la de prototipado como su nombre indica se hace a través de un prototipo inicial por lo cual se puede probar algo desde un principio [13].

- Iterativo: su funcionamiento está basado en pequeñas iteraciones, donde dentro de cada iteración se cumplen las fases del modelo de cascada.

  Algunas de sus ventajas son que permite entregas tempranas, así como adaptabilidad ante cualquiera cambio que pueda surgir, aunque puede ser costoso de gestionar [14].

Por otro lado, existen las denominadas metodologías ágiles, las cuales al contrario que las clásicas se caracterizan por su flexibilidad y su agilidad, junto a su funcionamiento incremental. Algunos tipos de metodología ágil son los siguientes [15]:

- Scrum: el funcionamiento de esta metodología se caracteriza por unas etapas, las cuales varían su duración según las necesidades del equipo. Estas etapas se conocen como *sprint*. Algunas de las actividades que se realizan dentro de cada sprint son las siguientes:

  1. *Sprint planning*: en este apartado se suele planear todo el *sprint* estableciendo algunas cosas básicas como las tareas que se necesitan realizar y quien va a realizar cada tarea o la duración de este.
  2. Ejecución: este apartado consiste en la realización de las tareas que le corresponden a cada uno que han sido establecidas en el apartado previo.
  3. *Daily meeting*: el objetivo de este apartado consiste en reuniones normalmente a primera hora de la mañana, en las que cada miembro del equipo cuenta al resto del equipo las tareas que realizo el día anterior y las tareas que tiene planeadas para ese día, así junto a posibles problemas que se haya podido encontrar, con el objetivo de mejorar la relación del equipo ya que todos los miembros saben en qué punto esta cada uno.
  4. *Sprint review*: en este último apartado lo que se realiza es una reunión en la que se revisa todo lo que se ha hecho durante el *sprint* al igual que lo que no se ha podido hacer.

- Lean: esta metodología ágil tiene el objetivo de que equipos pequeños puedan realizar cualquier tarea, cosa que consigue mediante la reducción de costes, tiempo y riesgos, dándole importancia al valor de lo que se realiza.

  Algunos de los beneficios de esta metodología son que se tiene menos costes que con otras metodologías, mejor calidad en el producto entregado o mayor moral de los empleados, lo cual está relacionado con el beneficio anterior.

  La manera de implementar esta metodología es cumpliendo con los siguientes principios:

  1. Eliminar desperdicio
  2. Generar calidad
  3. Crear conocimiento
  4. Generar compromiso
  5. Entregar rápido
  6. Respeto por las personas
  7. Optimizar

- Extreme Programming: el funcionamiento de esta metodología se basa en mejorar las relaciones entre los miembros del equipo y cumplir 12 principios base:

1. Diseño sencillo
2. Testing
3. Refactorización
4. Codificación con estándares
5. Propiedad colectiva del código
6. Programación en parejas
7. Integración continua
8. Entregas semanales
9. Integración con el cliente
10. Cliente *in situ*
11. Entregas frecuentes
12. Planificación

- Kanban: es aquella metodología cuyo funcionamiento se basa en la mejora constante y tareas extraídas de una lista de tareas pendientes.

  Para su implementación se usan los que se conocen como tableros Kanban, los cuales permiten visualizar el trabajo. Estos tableros están divididos en tres columnas según el estado de la tarea, *To do*, *Doing* y *Done*, que se traduciría como por hacer, haciendo y hecho.

  Cuenta con los siguientes principios básicos los cuales hay que seguir a la hora de usar esta metodología:

  1. Comienza con lo que sabes
  2. Busca una evolución incremental
  3. Respeta el proceso y los roles
  4. Apoya los actos de liderazgo

## 3.2. Metodología escogida

Para este proyecto no se ha utilizado una metodología en concreto, sino que se han utilizado herramientas de diferentes metodologías, como el tablero Kanban para separar las tareas según su estado.

Estas tareas se han estructurado a partir de las historias de usuario. Las historias de usuario son una técnica utilizada en el desarrollo de *software* que permite definir las funcionalidades del sistema desde la perspectiva del usuario final [16].

Cada historia de usuario describe la necesidad de un usuario en el sistema, y se acompaña de sus correspondientes subtareas. Este enfoque permite centrar el desarrollo en las necesidades y expectativas reales de los usuarios, asegurando que el sistema final sea útil y fácil de usar.

También se han seguido diferentes *sprints* donde se ha ido presentando a los tutores el progreso con entregas parciales.

Las historias de usuario definidas cuentan con un solo tipo de usuario, el usuario, que es la persona que usa la aplicación final.

Las Tablas 3.1, 3.2 3.3 3.4 3.5 3.6 y 3.7 presentan las diferentes historias de usuario:

| Historia de Usuario |
|---|
| Número 1 |
| Nombre: normalización de datos de los jugadores |
| Usuario |
| **Tarjeta de la HU**<br>Como usuario<br>Quiero que los datos de los jugadores se encuentren todos normalizados en un mismo formato<br>Para poder usarlos correctamente en un futuro |
| **Criterio de aceptación**<br>La datos deben encontrarse normalizados |
| **Conversación:**<br>**Desarrollador**: ¿Cómo quieres tener normalizados los datos?<br>**Usuario**: En un rango de valores de 0 a 1.<br>**Desarrollador**: ¿Qué datos quieres tener normalizados?<br>**Usuario**: Todos los relacionados con los totales de jugadores. |

Tabla 3.1: Historia de Usuario sobre normalización de datos totales de los jugadores.

Todas estas historias de usuario como se ha explicado previamente representan las tareas que se han realizado y que han ido pasando por los diferentes estados de *To do*, *Doing* y *Done* [15].

La Figura 3.1 representa las historias de usuario que se han trabajado en cada *sprint* mediante un diagrama de Gantt.

Aquellas historias que no aparecen en el diagrama es porque no se han llegado a implementar.

| Historia de Usuario |
| --- |
| Número 2 |
| Nombre: normalización de datos por partido de los jugadores |
| Usuario |
| **Tarjeta de la HU**<br>Como usuario<br>Quiero que los datos por partido de los jugadores se encuentren todos normalizados en un mismo formato<br>Para poder usarlos correctamente en un futuro |
| **Criterio de aceptación**<br>La datos deben encontrarse normalizados |
| **Conversación:**<br>**Desarrollador**: ¿Cómo quieres tener normalizados los datos?<br>**Usuario**: En un rango de valores de 0 a 1.<br>**Desarrollador**: ¿Qué datos quieres tener normalizados?<br>**Usuario**: Todos los relacionados con los datos por partido de los jugadores. |

Tabla 3.2: Historia de Usuario sobre normalización de datos por partido de los jugadores.

| Historia de Usuario |
| --- |
| Número 3 |
| Nombre: correlación de datos sin normalizar |
| Usuario |
| **Tarjeta de la HU**<br>Como usuario<br>Quiero poder visualizar una correlación entre puntos y estadísticas sin normalizar de los jugadores<br>Para tomar mejores decisiones |
| **Criterio de aceptación**<br>Se debe poder visualizar la correlación de los datos |
| **Conversación:**<br>**Desarrollador**: ¿Qué estadísticas se deben usar para realizar la correlación?<br>**Usuario**: Todas las posibles que no estén normalizadas.<br>**Desarrollador**: ¿Cómo quieres visualizar la correlación de los datos?<br>**Usuario**: Mediante una tabla y un rango de colores. |

Tabla 3.3: Historia de Usuario sobre correlación de datos sin normalizar.

| Historia de Usuario |
| --- |
| Número 4 |
| Nombre: correlación de datos normalizados |
| Usuario |
| **Tarjeta de la HU**<br>Como usuario<br>Quiero poder visualizar una correlación entre puntos y estadísticas normalizadas de los jugadores<br>Para tomar mejores decisiones |
| **Criterio de aceptación**<br>Se debe poder visualizar la correlación de los datos |
| **Conversación:**<br>**Desarrollador**: ¿Qué estadísticas se deben usar para realizar la correlación?<br>**Usuario**: Todas las posibles que estén normalizadas.<br>**Desarrollador**: ¿Cómo quieres visualizar la correlación de los datos?<br>**Usuario**: Mediante una tabla y un rango de colores. |

Tabla 3.4: Historia de Usuario sobre correlación de datos normalizados.

| Historia de Usuario |
| --- |
| Número 5 |
| Nombre: sistema asignación puntos |
| Usuario |
| **Tarjeta de la HU**<br>Como usuario<br>Quiero que haya un sistema que asigne una puntuación en cada partido de los jugadores según su rendimiento<br>Para poder realizar un algoritmo que los use en un futuro |
| **Criterio de aceptación**<br>Se debe asignar una puntuación a cada jugador |
| **Conversación:**<br>**Desarrollador**: ¿Cómo quieres asignar los puntos a los jugadores?<br>**Usuario**: Según el rendimiento en el partido. |

Tabla 3.5: Historia de Usuario sobre sistema de asignación de puntos.

| Historia de Usuario |
| --- |
| Número 6 |
| Nombre: comprobación sistema asignación |
| Usuario |
| **Tarjeta de la HU**<br>Como usuario<br>Quiero que se compruebe si el sistema de asignación de puntuación es válido<br>Para asegurar que su uso es correcto |
| **Criterio de aceptación**<br>Se debe poder comprobar la asignación de los puntos |
| **Conversación:**<br>**Desarrollador**: ¿Cómo quieres comprobar la asignación de los puntos?<br>**Usuario**: Mediante un cálculo de la correlación de los puntos. |

Tabla 3.6: Historia de Usuario sobre comprobación del sistema de asignación.

| Historia de Usuario |
| --- |
| Número 7 |
| Nombre: cálculo alineación óptima |
| Usuario |
| **Tarjeta de la HU**<br>Como usuario<br>Quiero que haya un sistema que, según una alineación a seguir, y una jornada me devuelva la alineación óptima en esa jornada<br>Para maximizar los puntos que obtengo en juegos *fantasy* |
| **Criterio de aceptación**<br>Se debe poder calcular una alineación óptima |
| **Conversación:**<br>**Desarrollador**: ¿Cómo quieres calcular la alineación óptima?<br>**Usuario**: Mediante el máximo de puntos que se predigan para la jornada.<br>**Desarrollador**: ¿Cómo quieres que se predigan los puntos?<br>**Usuario**: Mediante la media de los obtenidos en jornadas anteriores. |

Tabla 3.7: Historia de Usuario sobre el cálculo de la alineación óptima.

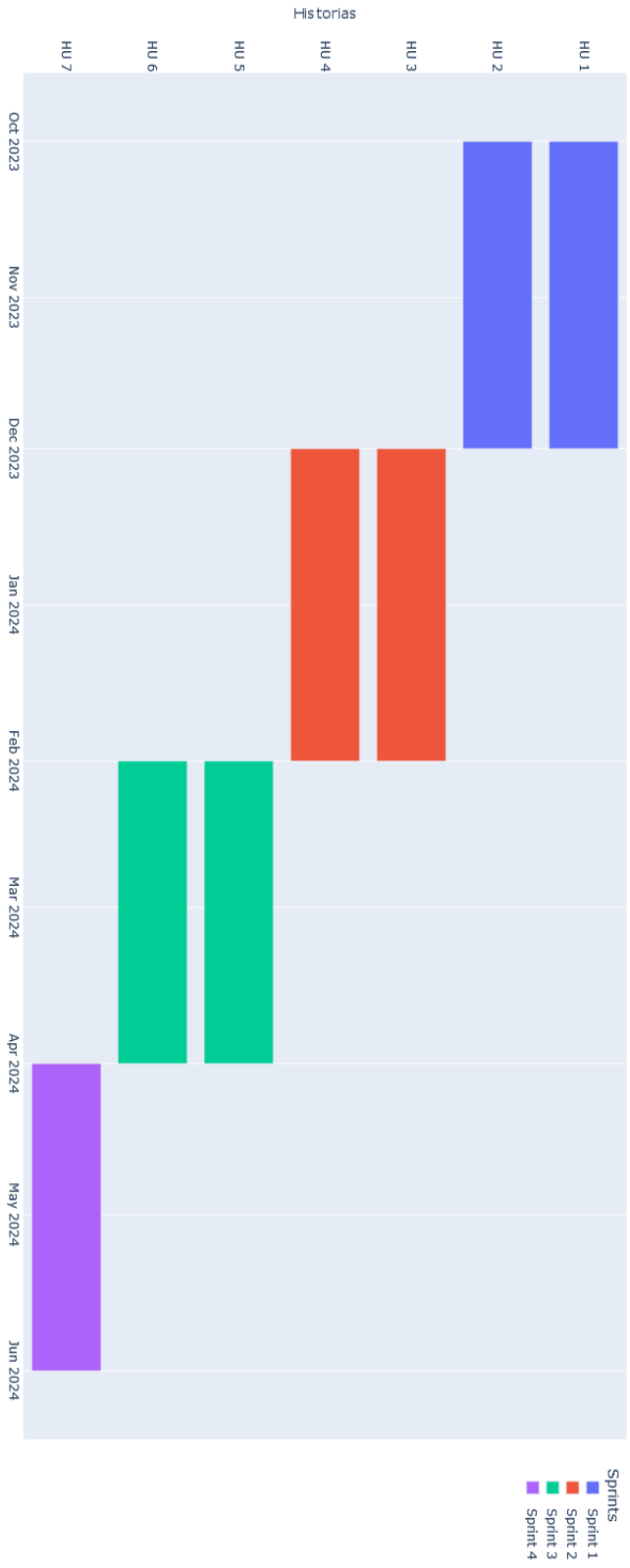



Figura 3.1: Diagrama de Gantt de las historias de usuario.

## 3.3. Metodología investigación

En este proyecto, no es suficiente con solo una metodología de desarrollo, sino que también es necesario una metodología de investigación para respaldar las decisiones e hipótesis que permiten abordar el problema de optimización planteado.

Mediante el diagrama en la Figura 3.2 se pueden apreciar los pasos que se han seguido mediante esta metodología de investigación. El primer paso consiste en realizar una observación llegando a la conclusión de que seleccionar el mejor once de fútbol para una jornada es un problema de optimización complejo debido a múltiples factores, como habilidades individuales, tácticas, lesiones y condiciones del partido.

Lo cual da paso al siguiente paso, con el que surge la pregunta de cómo se puede diseñar un algoritmo enumerativo que seleccione el mejor once de fútbol de manera eficiente y efectiva.

Después se llega a la parte de hipótesis donde se concluye que usando técnicas algorítmicas enumerativas se puede resolver el problema.

La parte de predicciones determina que si se entrena al algoritmo con datos históricos tanto de jugadores como de sus partidos se puede resolver el problema.

A continuación, en la parte de experimentación se realizan diferentes pasos:

- Se recolectan datos de partidos anteriores, incluyendo estadísticas de jugadores y resultados.
- Se diseña un algoritmo para abordar el problema de optimización planteado utilizando estos datos.
- Se evalúa el rendimiento del modelo utilizando métricas como puntos del resultado y tiempo en obtener un resultado.
- Se ajusta el algoritmo según los resultados obtenidos y se valida con datos de jornadas futuras.

Finalmente, se llega a la parte de conclusiones, donde a través de todo lo obtenido en la parte de experimentación se concluye que el algoritmo búsqueda enumerativo es capaz de seleccionar el mejor once.

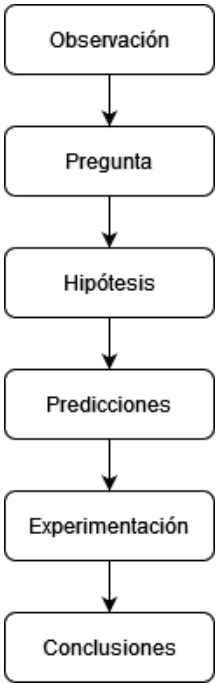



Figura 3.2: Diagrama de flujo del proceso de metodología de investigación.

# 4

# Preparación de los datos

En este capítulo se explican los pasos a seguir en el procesamiento y tratamiento de los datos para prepararlos para ser usados por el algoritmo enumerativo que se presenta en el siguiente capítulo.

Concretamente, en la Sección 4.1 se recoge la normalización que se ha realizado sobre los datos para que estén en un mismo formato, en la Sección 4.2 se recoge la correlación realizada a estos datos para ver cómo se comportan entre ellos y en la Sección 4.3 se presenta el sistema que se ha creado para asignar puntos por partido a los jugadores.

## 4.1. Normalización de los datos

La primera parte del proceso preparación de los datos para solucionar el problema y crear un sistema que ayude en la toma de decisiones, ha sido antes de realizar esta solución, preparar los datos, empezando por la normalización de los datos.

La normalización es una técnica que se aplica a menudo en el procesamiento de datos. Su objetivo es cambiar los valores de las columnas numéricas de un conjunto de datos para usar una escala común, sin distorsionar las diferencias en los intervalos de valores ni perder información.

En otras palabras, convierte los datos a una escala predeterminada, generalmente entre 0 y 1, facilitando su comparación y análisis. Al normalizar, se evita

redundancia, inconsistencias y errores en la gestión de datos [17].

Los datos que se han utilizado en este caso para normalizar han sido los datos reales y por partido de los jugadores que provienen de la extracción realizada en el anterior trabajo [18].

Sin embargo, no todos los datos han sido normalizados, solo aquellos que se pueden considerar como estadísticas, por ejemplo, pases, tiros o tarjetas amarillas.

La manera de realizar este proceso ha sido mediante Python [19] y la librería NumPy [20], el primer paso ha sido recorrer ambas tablas, en *scripts* diferentes, y generar una lista por cada columna que hubiese en la tabla, almacenando en estas todos los datos. Una vez se tienen las listas con todos los datos, el siguiente paso es ya la normalización en sí, donde utilizando solo las listas con los datos que se quieren normalizar se realiza el siguiente cálculo:

$$listaNormalizada = (listaOriginal - minimo)/(maximo - minimo)$$

Una vez se tienen las listas con los datos normalizados, se recorrer la lista que contiene los identificadores (id) primarios correspondientes a cada jugador y cómo se les identifica para cada tabla y según la posición de este id en la lista, se obtienen los elementos del resto de listas en la misma posición, ya que cuando se generaron al ser insertados a la vez ocupan la misma posición, y generando un nuevo objeto que insertar a una nueva tabla de la base de datos en MongoDB [21].

Las Figuras 4.1 y 4.2 representan los datos antes y después de ser normalizados respectivamente.

```
totalTackles: 3
blockTackles: 0
interceptionTackles: 0
totalDuels: 8
duelsWon: 4
dribblesAttempts: 1
dribblesSuccess: 1
dribblesPast: 1
```

Figura 4.1: Ejemplo de algunos de los datos antes de ser normalizados.

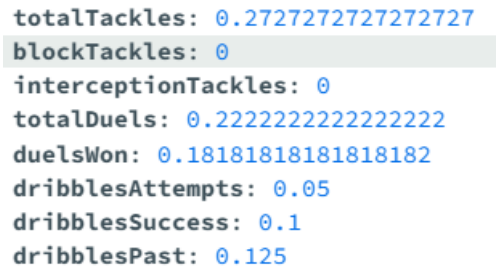



Figura 4.2: Ejemplo de algunos de los datos después de ser normalizados.

## 4.2. Análisis de la correlación

Una vez los datos han sido normalizados, el siguiente paso es realizar un cálculo de la correlación que tienen estos datos entre sí. Esta correlación se realiza principalmente entre las estadísticas y los puntos de los jugadores y la media de estos puntos por partido y por minuto jugado. El objetivo es comprobar qué estadísticas tener en cuenta cuando se realice el siguiente paso, el cuál consta en asignar una puntuación por partido a los jugadores. Este proceso también se realiza con Python, pero en este caso se utiliza la librería Pandas [22].

El primer paso que realizar es similar en parte al proceso anterior, ya que hay que recorrer toda la tabla con los datos reales de los jugadores y almacenarlos en una lista diferente para cada tipo de dato. Sin embargo, no se van a recorrer todos los jugadores ya que se realiza un filtro previo por posición de jugadores, ya que en un delantero sus estadísticas no se relacionan igual que en un defensa. Estos jugadores se ordenan de más a menos según el número de puntos que tengan en Biwenger, una vez se han almacenado los datos en las listas se crea un objeto de tipo JSON (del inglés *JavaScript Object Notation)* [23] donde el valor de cada clave es cada una de estas listas con los datos.

Una vez se tiene el objeto JSON mencionado anteriormente se crea un *dataframe* mediante la siguiente función de Pandas:

$$dataframe = pd.DataFrame(objetoJSON)$$

siendo *pd* una abreviatura de la librería Pandas y $DataFrame$ una función de esta que genera el *dataframe* mencionado antes. Posteriormente, una vez se tiene el *dataframe* el siguiente paso es generar la matriz de correlación que va a decir cómo se relacionan los datos entre ellos, lo cual se consigue mediante la siguiente función:

$$matrix = dataframe.corr()$$

Una vez se consigue esta matriz el siguiente paso es exportarla a formato CSV, del inglés *Comma Separated Values* donde se va a poder visualizar mejor la relación entre los datos. Estos son posible mediante esta función de la matriz:

$$matriz.to\_csv(\text{"}archivoSalida.csv\text{"}, sep = \text{";"})$$

Sin embargo, como solo interesa la relación de las estadísticas del jugador con los puntos, el resto de las columnas de la tabla se borran, quedando como resultado una tabla como la mostrada en la Tabla 4.1, donde se presenta la correlación de las estadísticas con los puntos de los 5 delanteros con más puntos:

| | points | pointsByMatches | pointsByMinutes |
|---|---|---|---|
| assistsByMatches | 0,85 | 0,9 | 0,89 |
| assistsByMinutes | 0,84 | 0,86 | 0,92 |
| blocksByMatches | 0,96 | 0,84 | 0,82 |
| blocksByMinutes | 0,95 | 0,83 | 0,82 |
| dribblesAttemptsByMatches | 0,06 | 0,36 | 0,15 |
| dribblesAttemptsByMinutes | 0,05 | 0,33 | 0,19 |
| dribblesSuccessByMatches | 0,11 | 0,4 | 0,26 |
| dribblesSuccessByMinutes | 0,1 | 0,37 | 0,3 |
| duelsWonByMatches | 0,18 | 0,47 | 0,18 |
| duelsWonByMinutes | 0,21 | 0,5 | 0,29 |
| foulsCommittedByMatches | -0,09 | 0,18 | -0,22 |
| foulsCommittedByMinutes | -0,08 | 0,19 | -0,19 |
| foulsDrawnByMatches | -0,01 | 0,25 | -0,07 |
| foulsDrawnByMinutes | -0,03 | 0,24 | -0,05 |
| goalsByMatches | -0,2 | -0,23 | -0,39 |
| goalsByMinutes | -0,17 | -0,23 | -0,33 |
| interceptionsByMatches | 0,94 | 0,8 | 0,93 |
| interceptionsByMinutes | 0,91 | 0,77 | 0,93 |
| keyPassesByMatches | 0,65 | 0,53 | 0,51 |
| keyPassesByMinutes | 0,66 | 0,49 | 0,58 |
| passesAccuracyByMatches | 0,64 | 0,67 | 0,84 |
| passesAccuracyByMinutes | 0,51 | 0,48 | 0,77 |

Tabla 4.1: Correlación de datos no normalizados de los 5 delanteros con más puntos.

En esta Tabla 4.1 se encuentran tres columnas que representan respectivamente los puntos totales, los puntos por partido, y los puntos por minuto de los jugadores. En cuanto a las filas, se encuentran todas las estadísticas de los jugadores tanto por minutos como por partidos.

Sin embargo, se puede ver que hay cosas que con un conocimiento de fútbol pueden darse por predecibles, mientras que otras no, por ejemplo, en los goles se

muestra una correlación negativa. Esto se debe a que, aunque un delantero meta muchos goles debido a un bajo rendimiento en otras estadísticas puede obtener una puntuación que afecte a la correlación con los goles.

Por lo tanto, se va a repetir este proceso, pero en este caso con los datos normalizados de los jugadores, siendo el único cambio que tabla se utiliza y devolviendo la Tabla 4.2.

Tras realizar la nueva correlación se puede ver en la Tabla 4.2, aparte de que cuenta con las mismas columnas y filas que la Tabla 4.1, que en gran parte sigue mostrándose igual, algo que se debe tener en cuenta para la siguiente parte del proceso, ya que, aunque este sistema pueda ser efectiva gran parte de las veces no siempre lo es del todo.

| | points | pointsByMatches | pointsByMinutes |
|---|---|---|---|
| assistsByMatches | 0,85 | 0,90 | 0,89 |
| assistsByMinutes | 0,84 | 0,86 | 0,92 |
| blocksByMatches | 0,96 | 0,84 | 0,82 |
| blocksByMinutes | 0,95 | 0,83 | 0,82 |
| dribblesAttemptsByMatches | 0,06 | 0,36 | 0,15 |
| dribblesAttemptsByMinutes | 0,05 | 0,33 | 0,19 |
| dribblesSuccessByMatches | 0,11 | 0,40 | 0,26 |
| dribblesSuccessByMinutes | 0,10 | 0,37 | 0,30 |
| duelsWonByMatches | 0,18 | 0,47 | 0,18 |
| duelsWonByMinutes | 0,21 | 0,50 | 0,29 |
| foulsCommittedByMatches | -0,09 | 0,18 | -0,22 |
| foulsCommittedByMinutes | -0,03 | 0,24 | -0,05 |
| foulsDrawnByMatches | -0,01 | 0,25 | -0,07 |
| foulsDrawnByMinutes | -0,03 | 0,24 | -0,05 |
| goalsByMatches | -0,20 | -0,23 | -0,39 |
| goalsByMinutes | -0,17 | -0,23 | -0,33 |
| interceptionsByMatches | 0,94 | 0,80 | 0,93 |
| interceptionsByMinutes | 0,91 | 0,77 | 0,93 |
| keyPassesByMatches | 0,65 | 0,53 | 0,51 |
| keyPassesByMinutes | 0,66 | 0,49 | 0,58 |
| passesAccuracyByMatches | 0,64 | 0,67 | 0,84 |
| passesAccuracyByMinutes | 0,51 | 0,48 | 0,77 |

Tabla 4.2: Correlación de datos normalizados de los 5 delanteros con más puntos.

## 4.3. Asignación de puntos

Para la resolución final del problema que se plantea, es necesario crear un sistema que asigne una puntuación a los jugadores por partido, ya que con esta

puntuación se va a crear el sistema que calcule la alineación a usar.

El primer paso se basa en obtener para cada uno de los jugadores los partidos que ha jugado y sus estadísticas en cada partido y, en función de las estadísticas que tuvo, se multiplica a cada una por un valor predeterminado y la suma de todo resulta en la puntuación en ese partido.

La manera de determinar qué estadísticas se usan y por qué valor multiplican se obtiene por una parte de la correlación realizada previamente, pero también, del conocimiento que se tiene sobre el deporte, ya que como se ha hablado anteriormente, en la correlación salía que los goles de un delantero tenían correlación negativa con los puntos, algo que no tiene sentido del todo, ya que los goles es algo primordial para un delantero. Por lo tanto, teniendo en cuenta esto se realizan 4 funciones que asignan los puntos a un jugador en un partido según su posición:

- Porteros:

$$puntos = 0{,}3 * paradas + 0{,}1 * penaltisParados + 0{,}05 * pasesTotales + 0{,}03 * duelosTotales + 0{,}07 * duelosGanados - 0{,}08 * golesConcedidos - 0{,}05 * tarjetasAmarillas - 0{,}06 * tarjetasRojas - 0{,}05 * penaltisRealizados - 0{,}03 * penaltisFallados$$

- Defensas:

$$puntos = 0{,}05 * entradasTotales + 0{,}07 * entradasBloqueo + 0{,}07 * intercepciones + 0{,}04 * duelosTotales + 0{,}075 * duelosGanados + 0{,}03 * pasesTotales + 0{,}05 * pasesClave - 0{,}05 * golesConcedidos - 0{,}05 * tarjetasAmarillas - 0{,}1 * tarjetasRojas - 0{,}05 * penaltisRealizados$$

- Mediocentros:

$$puntos = 0{,}05 * pasesTotales + 0{,}08 * pasesClave + 0{,}1 * asistencias + 0{,}03 * duelosTotales + 0{,}05 * duelosGanados + 0{,}1 * intercepciones + 0{,}03 * entradasTotales + 0{,}05 * entradasBloqueo + 0{,}05 * faltasRecibidas + 0{,}05 * regatesExitosos - 0{,}05 * faltasRealizadas - 0{,}03 * tarjetasAmarillas - 0{,}05 * tarjetasRojas - 0{,}02 * penaltisRealizados$$

- Delanteros:

$$puntos = 0{,}2 * golesTotales + 0{,}1 * asistencias + 0{,}05 * tirosPorteria + 0{,}07 * pasesClave + 0{,}1 * regatesExitosos + 0{,}06 * penaltisMarcados + 0{,}03 * faltasRecibidas + 0{,}05 * penaltisGanados - 0{,}05 * fuerasDeJuego - 0{,}03 * tarjetasAmarillas - 0{,}05 * tarjetasRojas - 0{,}05 * penaltisFallados - 0{,}03 * penaltisRealizados$$

Tras realizar la asignación de puntos por partido a cada jugador se debe comprobar si el sistema es correcto, esto se hace mediante una comprobación de la correlación de los puntos totales y sus medias con los asignados por el sistema. La manera de hacer esto es igual a cómo se ha hecho para las Tablas 4.1 y 4.2, resultando en la Tabla 4.3, en la cual se aprecia la correlación positiva entre los puntos del sistema creado y los puntos de Biwenger por lo cual este sistema se puede usar para la resolución del problema.

En la Tabla 4.3 las columnas que se encuentran hacen referencia por orden a los puntos totales de Biwenger, la media de puntos en Biwenger de cada jugador, los puntos asignados a cada jugador y la media de puntos asignados.

| | biwengerP | biwengerPBM | customP | customPBM |
|---|---|---|---|---|
| biwengerP | 1,00 | 0,50 | 0,83 | 0,76 |
| biwengerPBM | 0,50 | 1,00 | 0,33 | 0,37 |
| customP | 0,83 | 0,33 | 1,00 | 0,94 |
| customPBM | 0,76 | 0,37 | 0,94 | 1,00 |

Tabla 4.3: Tabla mostrando la correlación de los puntos asignados por el sistema creado y los puntos Biwenger.

# 5

# Formalización del problema y propuesta algorítmica

En este capítulo, se modela el problema a resolver como un problema de optimización (Sección 5.1). Posteriormente, en la Sección 5.2, se presenta la propuesta algorítmica desarrollada para abordar dicho problema.

## 5.1. Formalización del problema

La optimización es un campo de las matemáticas aplicadas que se centra en la búsqueda de la solución óptima dentro de un conjunto más amplio de soluciones factibles. De esta manera, mejora la toma de decisiones proporcionando información que puede no ser intuitiva inicialmente. Un problema de optimización está compuesto por una o varias funciones objetivo, que representan lo que se busca optimizar, ya sea intentando maximizar o minimizar el valor de la función objetivo. Además, presenta una serie de restricciones que la solución óptima debe satisfacer.

El problema que se quiere solucionar es calcular la que se considera la alineación óptima para una jornada, para ello hay que tener en cuenta ciertos aspectos como la jornada en la que se encuentra el usuario o el límite de presupuesto.

Una alineación óptima es aquella que obtiene más puntos en una jornada.

En lenguaje matemático se podría expresar de la siguiente manera, primero

se definen los conjuntos y parámetros a utilizar:

- $I$: conjunto de jugadores indexados por $i$
- $J$: conjunto de jornadas indexadas por $j$

A continuación, se presentan las variables utilizadas:

- $p_{ij}$: puntos del jugador $i$ en la jornada $j$
- $x_i$: variable binaria con valor 1 si el jugador $i$ es seleccionado, y 0 en caso contrario
- $k_i^G$: variable binaria con valor 1 si el jugador $i$ es portero, y 0 en caso contrario
- $k_i^D$: variable binaria con valor 1 si el jugador $i$ es defensa, y 0 en caso contrario
- $k_i^M$: variable binaria con valor 1 si el jugador $i$ es mediocentro, y 0 en caso contrario
- $k_i^A$: variable binaria con valor 1 si el jugador $i$ es atacante, y 0 en caso contrario
- $c_i$: coste del jugador $i$
- $W$: límite de coste
- $N_D$: número de defensas
- $N_M$: número de mediocentros
- $N_A$: número de atacantes

El objetivo del problema de optimización planteado es maximizar la suma de puntos de los jugadores $i$ escogidos en el 11 para una jornada en concreto, $j$.

En la Ecuación 5.1 resultante $p_{ij}$ representa los puntos del jugador $i$ en la jornada $j$ y $x_i$ representa si el jugador $i$ es seleccionado o no.

$$\text{máx} \sum_{i,j=1}^{n} p_{ij} x_i \tag{5.1}$$

Sin embargo, para poder encontrar una solución el problema debe cumplir unas restricciones, las cuales se representan a continuación:

La restricción en la Ecuación 5.2 representa que la suma de porteros en la solución sea 1, siendo $k_i^G$ 1 si el jugador $i$ es portero y 0 en caso contrario, mientras que $x_i$ representa si el jugador $i$ es seleccionado.

La restricción en la Ecuación 5.3 representa que la suma de defensas en la solución es igual al número de defensas, siendo $k_i^D$ 1 si el jugador $i$ es defensa y 0 en caso contrario, mientras que $x_i$ representa si el jugador $i$ es seleccionado.

La restricción en la Ecuación 5.4 representa que la suma de mediocentros en la solución es igual al número de mediocentros, siendo $k_i^M$ 1 si el jugador $i$ es mediocentro y 0 en caso contrario, mientras que $x_i$ representa si el jugador $i$ es seleccionado.

La restricción en la Ecuación 5.5 representa que la suma de atacantes en la solución es igual al número de atacantes, siendo $k_i^A$ 1 si el jugador $i$ es atacante y 0 en caso contrario, mientras que $x_i$ representa si el jugador $i$ es seleccionado.

La restricción en la Ecuación 5.6 representa que la suma total de defensas, mediocentros y delanteros en la solución debe ser 10.

Y finalmente, la Ecuación 5.7 limita que la suma de los precios de los $i$ jugadores seleccionados no debe superar el límite de precio establecido, representado por $W$.

$$\sum_{i=1}^{n} k_i^G x_i = 1 \tag{5.2}$$

$$\sum_{i=1}^{n} k_i^D x_i = N_D \tag{5.3}$$

$$\sum_{i=1}^{n} k_i^M x_i = N_M \tag{5.4}$$

$$\sum_{i=1}^{n} k_i^A x_i = N_A \tag{5.5}$$

$$\sum_{i=1}^{n} k_i^D x_i + \sum_{i=1}^{n} k_i^M x_i + \sum_{i=1}^{n} k_i^A x_i = 10 \tag{5.6}$$

$$\sum_{i=1}^{n} c_i x_i <= W \tag{5.7}$$

## 5.2. Resolución del problema

Una vez formalizado el problema de optimización, este se puede resolver empleando distintas técnicas algorítmicas. Entre ellas se pueden diferenciar las técnicas exactas (aquellas que encuentran y certifican la solución óptima) y técnicas aproximadas, aquellas que no son capaces de determinar si una solución encontrada es óptima o no.

En la parte algorítmica, en este TFG, lo que se ha realizado para resolver el problema, es implementar en Python un algoritmo exacto a través de la técnica de *backtracking* [24]. Este tipo de algoritmos, se denominan algoritmos enumerativos, ya que contemplan todas las soluciones posibles mediante llamadas a sí mismo, lo que se conoce como recursividad, [25] y retrocediendo hacia atrás para conseguir explorar todas las opciones.

En la Figura 5.1 [26] se muestra una representación genérica del algoritmo de *backtracking*. En esta figura, se observa que el algoritmo construye un árbol de búsqueda, partiendo de una solución vacía en la raíz y acabando en una solución completa en las hojas del árbol. Cada círculo azul, por lo tanto, representa un nodo del árbol de búsqueda y con ello una solución (ya sea completa o parcial). Este algoritmo va bajando hasta llegar a los nodos conocidos como "hojas", para después retroceder, también cuenta con la existencia de podas lo cual es una estrategia de búsqueda avanzada que evita contemplar soluciones poco prometedoras.

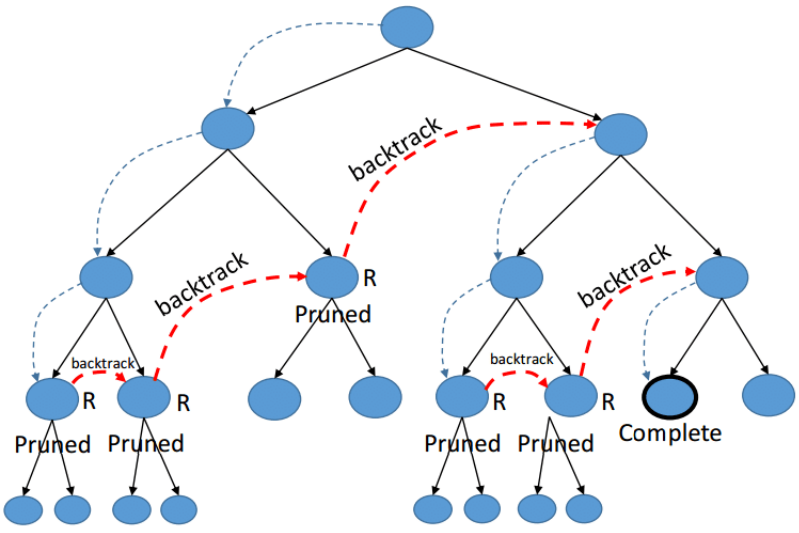



Figura 5.1: Ejemplo de recorrido de algoritmo genérico de *backtracking*.

A las restricciones del problema se han añadido unas nuevas, las cuales se conoces como podas, con el objetivo de reducir el tiempo de ejecución del problema.

Para el problema se necesitan los siguientes parámetros:

- Lista de jugadores ordenados de más a menos puntos hasta la jornada límite
- Lista con los precios de cada jugador en el mismo orden
- Lista con los puntos de cada jugador
- Lista con la posición de cada jugador
- Diccionario con el máximo de jugadores por posición según la alineación a usar
- El precio máximo a no sobrepasar
- Índice en la lista del jugador que se tiene en cuenta. Se inicializa en 0
- Solución actual. Se inicializa como una lista vacía
- Suma de precios actual. Se inicializa en 0
- Suma de puntos actual. Se inicializa en 0
- Numero de jugadores por posición actual. Se inicializa en 0 para cada solución.

El primer paso que realiza el Algoritmo 1, una vez se entra a él, es si el precio que se tiene en ese momento es mayor al límite establecido, retrocede a una llamada anterior, siendo esta la primera poda realizada, si no lo supera, el algoritmo considera si la solución que tiene en ese momento tiene 11 jugadores o si se está considerando el último jugador de la lista. En el caso de que, si la solución actual con la que llega tenga 11 jugadores, la suma de precios no supere el límite y la suma de puntos sea mayor a la que se tiene actualmente, se realiza el cambio en la mejor solución encontrada hasta el momento, dado que se ha encontrado una nueva solución de mejor calidad y retrocede para considerar el resto de soluciones. En el caso de que se esté considerando el último jugador de la lista, el algoritmo empieza a retroceder.

Una estrategia que sigue el algoritmo para reducir su tiempo de ejecución, son las podas, esta es una estrategia avanzada de búsqueda, que se introduce en diferentes partes del algoritmo. Su manera de funcionar es como un filtro, ya que cuando se está considerando un jugador, si llega a una poda y no la cumple, el algoritmo retrocede al jugador anterior.

Si no cumple ninguna de las condiciones se avanza en el algoritmo y comprueba si la posición del jugador actual ha llegado al límite, si no ha llegado al límite, se considera la siguiente poda. Esta se complementa con la comprobación de si el jugador que se está considerando se puede añadir a la solución dado que no se

ha superado el límite de jugadores en su posición, y comprueba que el precio del jugador actual por sí solo no sea mayor al límite y que si se añadiese al precio actual esa suma no sea superior al límite. Si todo esto se cumple este jugador se añade a la solución actual, y se aumentan los precios y puntos y el número de jugadores en esa posición, y se pasa al siguiente jugador.

En el caso de volver de una llamada anterior en la que se ha contemplado un jugador, se reduce en uno la cantidad de jugadores usados en la posición del último jugador que se ha comprobado, dado que se elimina de la posible solución para comprobar el siguiente. La última poda que se comprueba en el algoritmo es si el último jugador que se ha comprobado en la solución era el último que se podía añadir y si cumplía los requisitos de precio y posiciones para considerar esa solución válida. En caso de cumplirlo se retrocede a una llamada anterior ya que se determina que no hay ningún jugador posterior a este último que pueda mejorar la posible solución ya que como los jugadores están ordenados por puntos, solo se van a encontrar jugadores con los mismos puntos o menos.

Si no se cumple la última poda, el algoritmo pasa a considerar al siguiente jugador para la solución, pero sin modificar ni precio, ni puntos y sin añadirlo todavía a la solución.

Una vez se han explorado todas las soluciones el algoritmo devuelve la mejor solución encontrada, es decir, la solución óptima.

**Algoritmo 1** Algoritmo de selección de jugadores con podas.

```
1: function SeleccionarJugadores(solucion_actual, solucion_optima, jugador_actual)
2:     if solucionValida(solucion_actual) then
3:         if puntos(solucion_actual) > puntos(solucion_optima) then
4:             solucion_optima ← solucion_actual
5:         end if
6:     end if
7:     if puedeAñadir(jugador_actual, solucion_actual) y pasaPodas(jugador_actual) then
8:         SeleccionarJugadores(solucion_actual + jugador_actual, siguiente(jugador_actual))
9:     end if
10:    if noPasaOtrasPodas(solucion_actual) then
11:        return
12:    end if
13:    SeleccionarJugadores(solucion_actual, siguiente(jugador_actual))
14:    return solucion_optima
15: end function
```

Aunque el Algoritmo 1, el cual cuenta con las podas, se ha realizado el Algo-

ritmo 2 representa el algoritmo resultante de eliminar las podas, con el objetivo de verificar que el funcionamiento del Algoritmo 1 con podas es mejor. El Algoritmo 2 tiene el mismo funcionamiento que el anterior con podas, salvo que al no tener las podas se consideran más soluciones válidas, tardando más en llegar a la solución óptima.

**Algoritmo 2** Algoritmo de selección de jugadores sin podas.

```
1:  function SeleccionarJugadores(solucion_actual, solucion_optima, jugador_actual)
2:      if solucionValida(solucion_actual) then
3:          if puntos(solucion_actual) > puntos(solucion_optima) then
4:              solucion_optima ← solucion_actual
5:          end if
6:      end if
7:      if puedeAñadir(jugador_actual, solucion_actual) then
8:          SeleccionarJugadores(solucion_actual + jugador_actual, solucion_optima, siguiente(jugador_actual))
9:      end if
10:     SeleccionarJugadores(solucion_actual, solucion_optima, siguiente(jugador_actual))
11:     return solucion_optima
12: end function
```

Las Figuras 5.2 y 5.3 representan un ejemplo de cómo sería el árbol de decisiones para los algoritmos con poda y sin poda respectivamente.

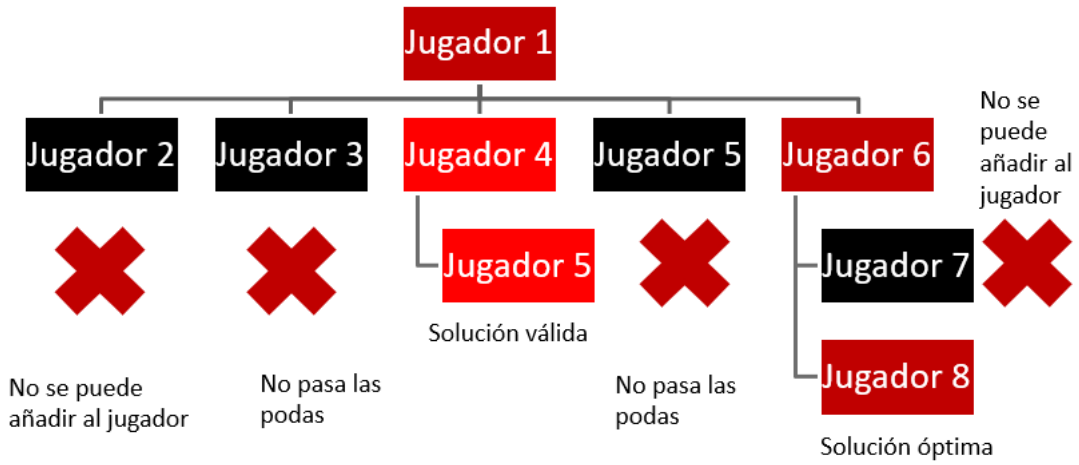



Figura 5.2: Ejemplo de árbol de decisión para el algoritmo con poda.

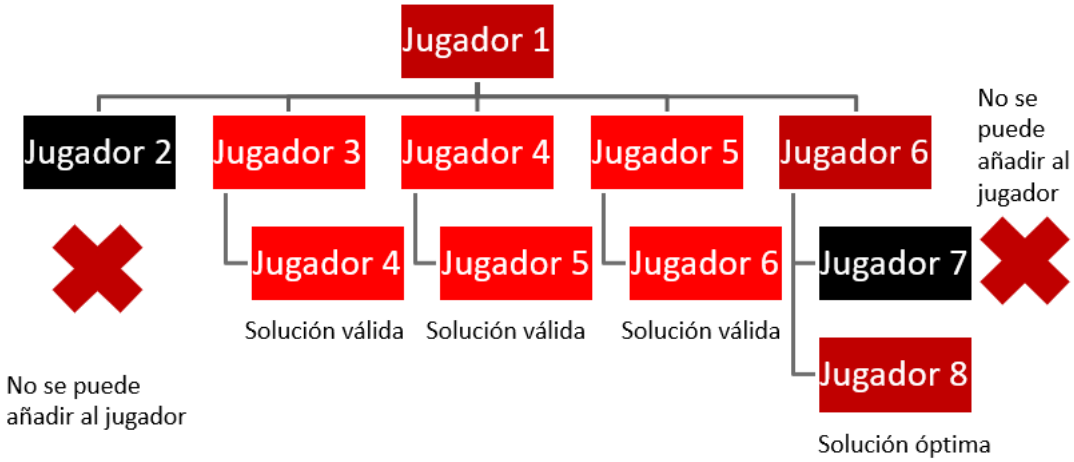



Figura 5.3: Ejemplo de árbol de decisión para el algoritmo sin poda.

Todo el código se encuentra en este repositorio de Github (`https://github.com/cuadantonio/TFG-AI_Assitant`).

# 6
# Resultados

En este capítulo se recogen los diferentes resultados, mostrando en la Sección 6.1 las gráficas de las diferentes pruebas realizadas con el algoritmo propuesto, y en la Sección 6.2 las interfaces que tiene la aplicación que se ha realizado.

## 6.1. Análisis resultados

Para garantizar el correcto funcionamiento del algoritmo con poda, se han realizado pruebas para evaluar tanto los tiempos de búsqueda de una solución válida como la calidad de dicha solución en función del tiempo.

Para la primera prueba, se seleccionaron dos alineaciones diferentes: 4-3-3 y 5-4-1. Se aumentó gradualmente el número de jugadores por posición, comenzando con el mínimo requerido para cada alineación (1 portero, 4 defensas, 3 mediocentros y 3 delanteros para 4-3-3; 1 portero, 5 defensas, 4 mediocentros y 1 delantero para 5-4-1). El objetivo fue determinar el tiempo necesario para obtener una solución en cada caso.

Como se ve en las Figuras 6.1 y 6.2, con el número de jugadores por posición ordenados por porteros, defensas, mediocentros y delanteros en el eje X y el tiempo que ha tardado en encontrar una solución en el eje Y, cuando el número de jugadores es más bajo, el tiempo tanto para el algoritmo con poda como sin poda es bastante similar, pero según se aumenta este número de jugadores, las diferencias de tiempos entre un algoritmo y otro son mayores al igual que el tiempo

que va aumentando considerablemente. En el siguiente aumento de jugadores el tiempo ya era superior a 1 hora.

Por lo cual se puede concluir, que cuantos más jugadores se incluyan, mayor será el tiempo que tarde la aplicación.

En la siguiente prueba, se repite el proceso de la anterior. Se toman dos alineaciones, 4-3-3 y 5-4-1, y se selecciona un número de jugadores que no haya tardado mucho en encontrar una solución en la prueba previa. Para el 4-3-3, se utilizan 3 porteros, 12 defensas, 9 mediocampistas y 9 delanteros. Para el 5-4-1, se emplean 3 porteros, 15 defensas, 12 mediocampistas y 3 delanteros. Luego, se varía el presupuesto pasado al algoritmo para observar nuevamente los tiempos de ejecución. El presupuesto se basa en la media de los precios de los jugadores multiplicada por 11, que es el número total de jugadores en una alineación.

En las Figuras 6.3 y 6.4, se representan los diferentes presupuestos en el eje X y en el eje Y el tiempo que ha tardado en encontrar una solución. Se puede comprobar que para ambos algoritmos cuando el presupuesto es mayor el tiempo aumenta debido al aumento de posibles soluciones, y disminuye cuando el presupuesto baja por el mismo motivo. También se observa que, para el algoritmo con poda, cuando el presupuesto disminuye, se aplican más podas que tiene implementadas, relacionadas con el precio y, por lo tanto, se reduce su tiempo considerablemente, mientras que el algoritmo sin poda al seguir considerando todas las soluciones no baja tanto su tiempo.

Para la tercera prueba, lo que se ha realizado es con algunos parámetros de anteriores pruebas como el número de jugadores, pero solo para una alineación, y el presupuesto, limitar el tiempo que se va ejecutando la aplicación y comprobar si cambia la calidad de las soluciones obtenidas en cada caso.

En la Figura 6.5, con el tiempo en minutos que ha estado ejecutando cada vez en el eje X y los puntos obtenidos en el eje Y, se puede observar que, con el paso del tiempo de ejecución del algoritmo, este va encontrado soluciones con más puntos, sin embargo, al llegar a los 40 minutos de ejecución el algoritmo no encuentra soluciones con más puntos.

Para la última prueba, se ha comprobado el funcionamiento del algoritmo, con casos más reales.

Para esta prueba se ha ejecutado el algoritmo con todos los jugadores y utilizando como sistema de predicción la media de puntos de los jugadores y un límite de presupuesto obtenido mediante la media de los precios de los jugadores, multiplicado primero por 11 ya que son los jugadores que se tienen y después por 2 para aumentar el presupuesto.

Los casos que se han realizado se han diferenciado en que en uno solo se tenían en cuenta las últimas 5 jornadas anteriores a la que se pasaba como actual, y en

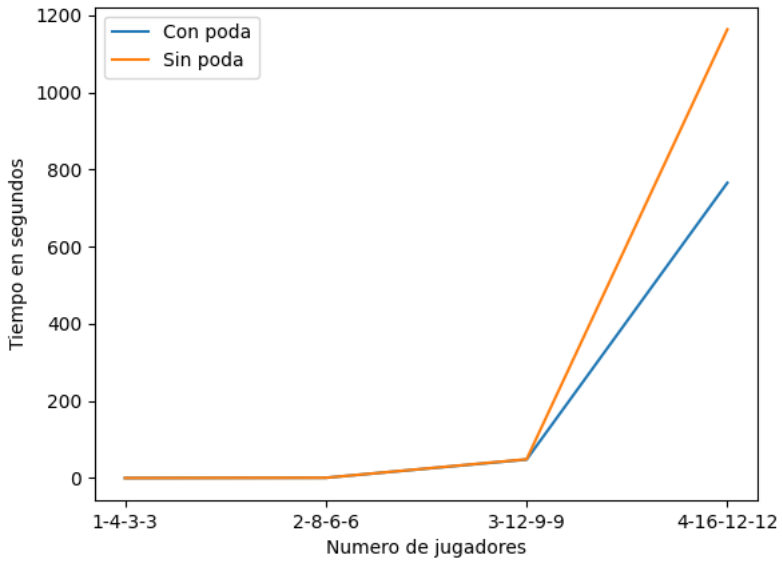



Figura 6.1: Gráfica con los tiempos para la alineación 4-3-3.

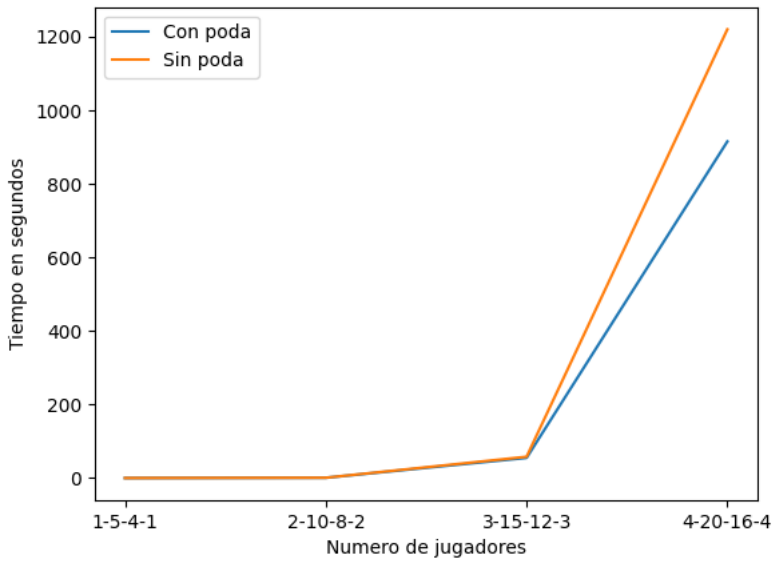



Figura 6.2: Gráfica con los tiempos para la alineación 5-4-1.

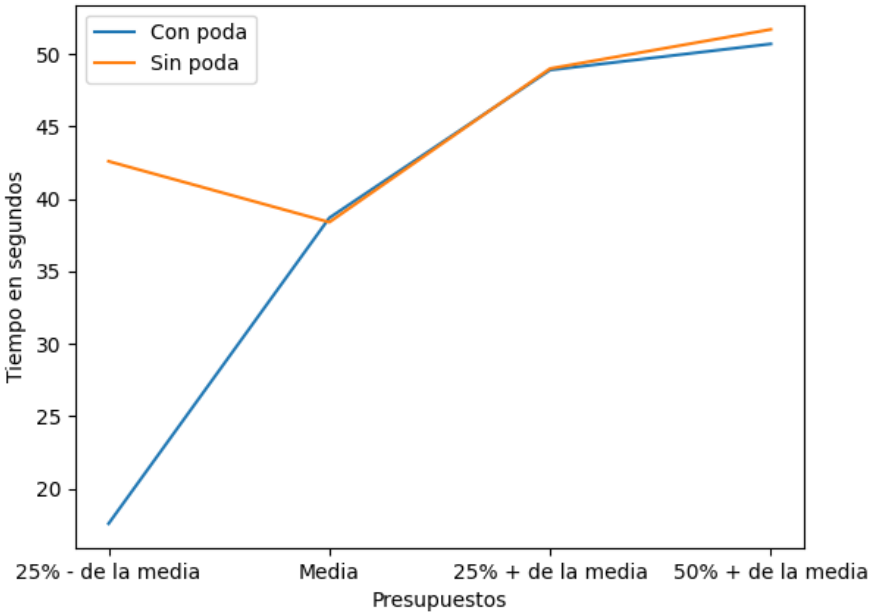



Figura 6.3: Gráfica con los tiempos para la alineación 4-3-3 cambiando los presupuestos.

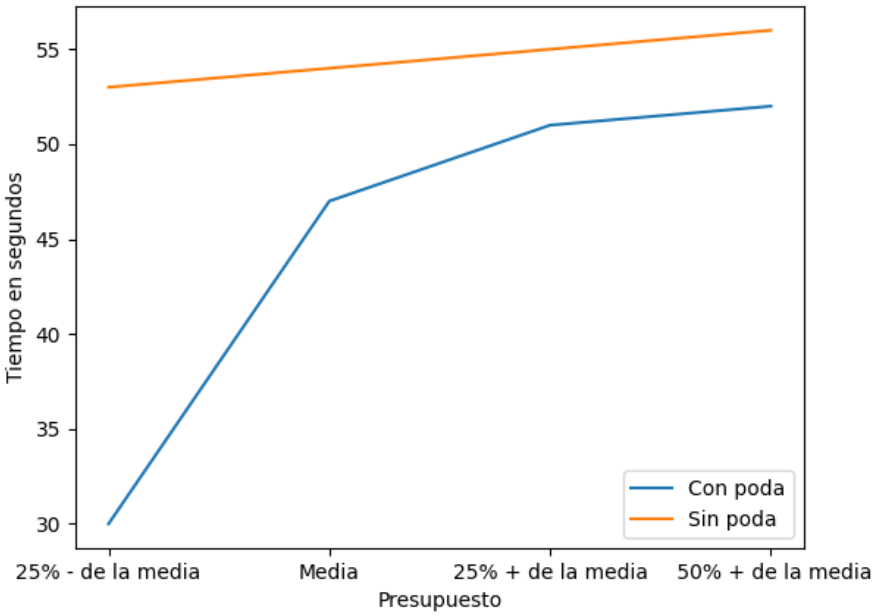



Figura 6.4: Gráfica con los tiempos para la alineación 5-4-1 cambiando los presupuestos.

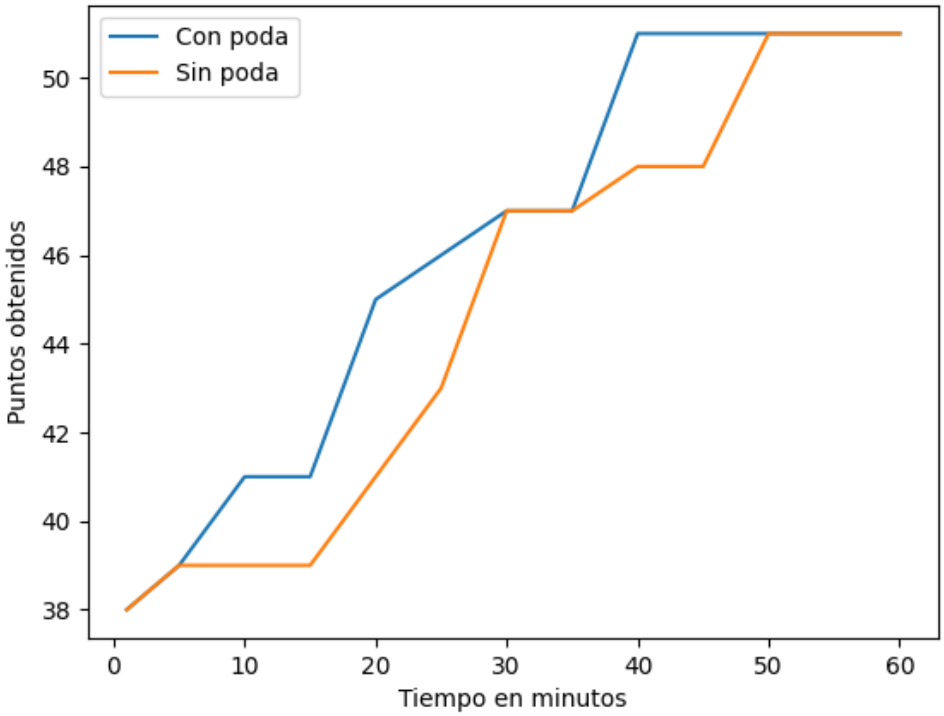



Figura 6.5: Gráfica que compara la calidad de los puntos obtenidos según el tiempo.

el otro caso eran todas las jornadas anteriores a la actual.

Los puntos obtenidos en cada jornada se han comparado después con el máximo de esa jornada.

En la Figura 6.6, con el eje X representando las diferentes jornadas que se han comprobado y en el eje Y los puntos obtenidos, se puede observar que, el algoritmo al realizar una predicción de puntos basada en la media obtenida por los jugadores hasta el momento, al principio se aleja más de los puntos máximos en esa jornada, ya que al principio el rendimiento de los jugadores puede variar, pero según pasan más las jornadas se acerca a los puntos máximos.

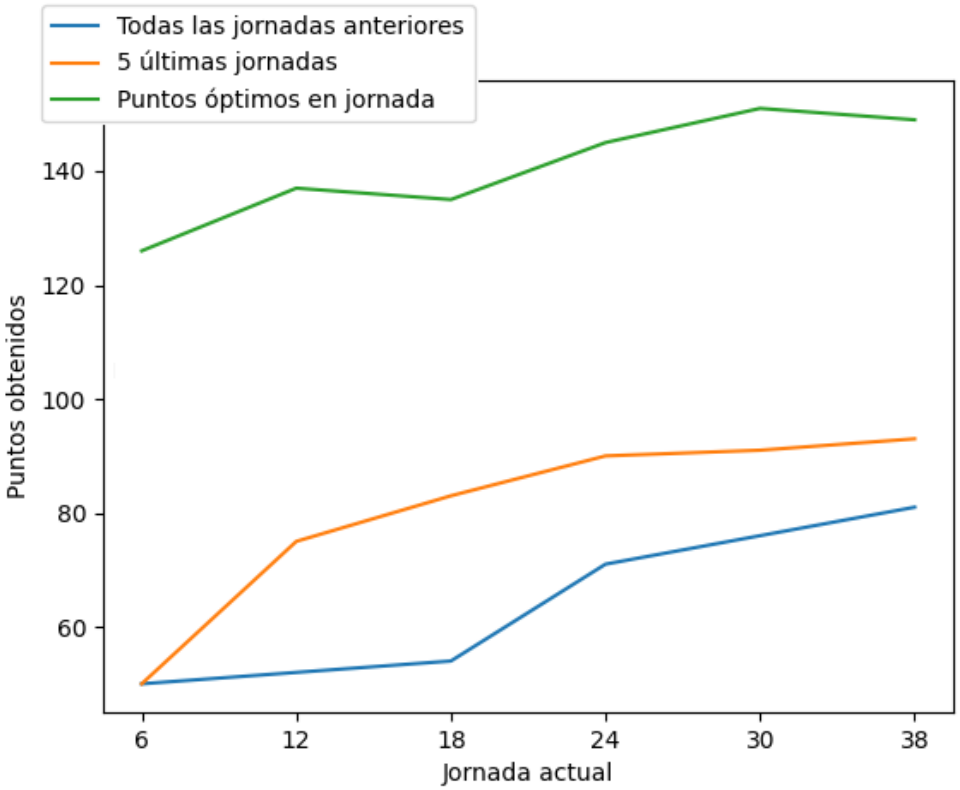



Figura 6.6: Gráfica que compara los resultados obtenidos en diferentes jornadas y con diferentes opciones.

## 6.2. Interfaces aplicación usuario

Para facilitar el uso del algoritmo a cualquier usuario se ha desarrollado una aplicación que implementa el uso de este. La aplicación consta de dos interfaces, la primera de ellas realizada con la librería de Python PySimpleGui (`https://www.pysimplegui.com/`) en la que el usuario va a introducir los parámetros

necesarios para la ejecución del algoritmo.

En la Figura 6.7 se puede comprobar los diferentes campos que el usuario debe rellenar para obtener una alineación.

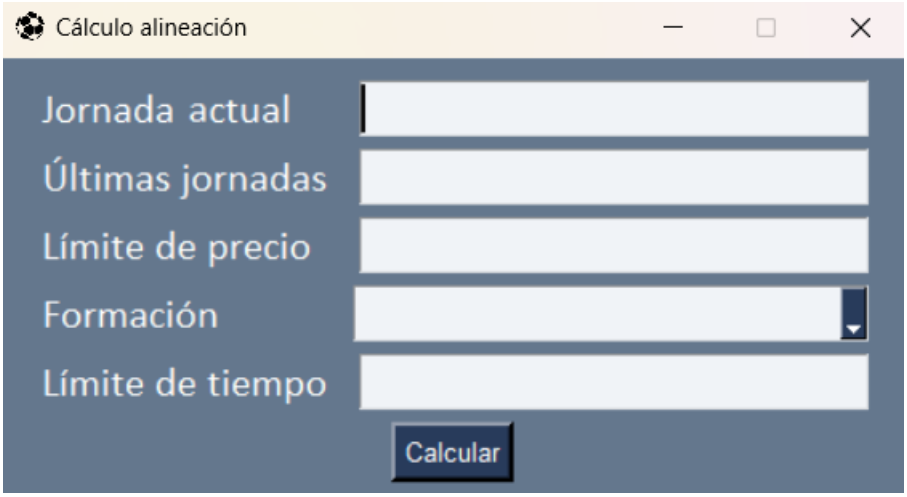



Figura 6.7: Interfaz de la aplicación para introducir los datos.

El primero de estos campos, denominado "Jornada actual" es la jornada en la que se encuentra el usuario que sirve para obtener los puntos de los jugadores hasta ese momento.

Después, se encuentra el campo de "Últimas jornadas", el cual determina si se quiere que el algoritmo tenga en cuenta todas las jornadas anteriores a la actual para realizar sus cálculos o solo un numero de jornadas anteriores a la actual. Si el campo se deja vacío se considera que el usuario quiere tener en cuenta todas las jornadas anteriores a la actual.

En el tercer campo, denominado "Límite de precio"se puede ver para rellenar el límite de precio que debe seguir la posible solución.

Siguiendo por el cuarto campo, denominado "Formación", se observa un desplegable que contiene las diferentes alineaciones que puede usar el algoritmo según sus restricciones. En la Figura 6.8 se pueden ver las diferentes alineaciones a usar.

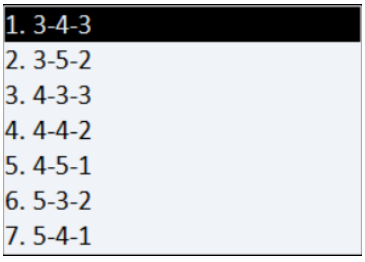



Figura 6.8: Diferentes alineaciones a usar en la aplicación.

Y finalmente, se encuentra el campo denominado "límite de tiempo" que indica al algoritmo cuanto tiempo puede estar ejecutando medido en minutos, de tal manera que si el algoritmo no tiene tiempo suficiente devuelve una solución, pero esta no es la óptima.

Una vez se rellenan todos estos campos, se pasan al algoritmo para obtener la solución, y con la librería de Python, Pygame (`https://www.pygame.org/news`) se muestra la solución encontrada al usuario en un formato diferente según la alineación seleccionada y con la predicción de puntos basada en la media de puntos obtenidos por los jugadores anteriormente.

En la Figura 6.9 se muestra un ejemplo de la interfaz la alineación considerada como óptima.


Figura 6.9: Interfaz que muestra la solución obtenida.

En las Figuras 6.10 y 6.11 se muestra el primer caso de uso de la aplicación con el algoritmo en el que solo se tienen en cuenta las 5 jornadas anteriores a la 10, el presupuesto es de 80 millones lo cual es un poco más alto que la media, y se deja ejecutar al algoritmo por 1 minuto.

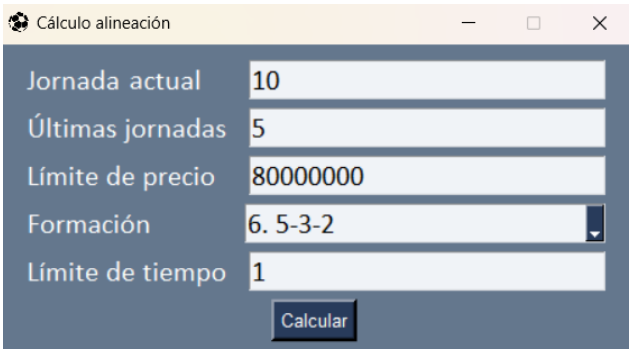



Figura 6.10: Interfaz que muestra los parámetros del primer caso de uso.


Figura 6.11: Interfaz que muestra la solución obtenida en el primer caso de uso.

En las Figuras 6.12 y 6.13 se muestra el segundo caso de uso de la aplicación con el algoritmo, pero ahora se tienen en cuenta todas las jornadas anteriores a la 17, el presupuesto es de 60 millones que se acerca a la media, y el algoritmo está ejecutándose por 1 minuto.

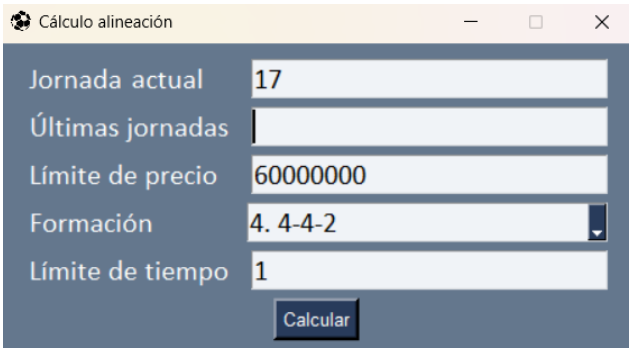



Figura 6.12: Interfaz que muestra los parámetros del segundo caso de uso.


Figura 6.13: Interfaz que muestra la solución obtenida en el segundo caso de uso.

# 7

# Conclusiones y trabajos futuros

Finalmente, en este capítulo se encuentran las conclusiones obtenidas al realizar este trabajo en la Sección 7.1 y sobre los posibles trabajos futuros a realizar en la Sección 7.2.

## 7.1. Conclusiones

Una vez realizado este trabajo se puede concluir que se ha creado un algoritmo enumerativo que sirve de apoyo al usuario para los juegos *fantasy manager*. Una vez creado el algoritmo que tiene como objetivo encontrar la solución del problema de optimización planteado se ha realizado una aplicación de escritorio para que el usuario pueda usar este algoritmo de forma sencilla y visualizar las diferentes soluciones que encuentre.

Posteriormente, se han realizado diferentes pruebas para comprobar el funcionamiento del algoritmo, observando tiempos de ejecución y calidad de las soluciones. Después, se ha cumplido con los objetivos específicos planteados al inicio del proyecto, siguiendo una metodología ágil basada en Kanban y utilizando herramientas y lenguajes adecuados para cada tarea.

De todo este proyecto se ha aprendido el proceso que sigue un problema de optimización, desde la investigación para ver que opción a seguir es la mejor, como el propio planteamiento del problema y la implementación para obtener la solución al problema.

Finalmente, se ha comprobado que, aunque el funcionamiento del algoritmo es correcto y obtiene una solución óptima, esta está basada en datos de jornadas anteriores. En este sentido, la propuesta presenta ciertas limitaciones, ya que la alineación sugerida únicamente es una predicción basada en información anterior, y puede haber casos que el algoritmo no contempla, como por ejemplo la lesión de un jugador de cara a la próxima jornada y que el algoritmo te lo recomiende por haberlo hecho muy bien en jornadas anteriores,aunque realmente no vaya a jugar.

## 7.2. Trabajos futuros

En esta sección, se exploran posibles direcciones para futuras extensiones basadas en los resultados y conclusiones de este TFG. Estas sugerencias pueden servir como punto de partida para investigaciones adicionales, contribuir al avance del campo, o ideas para la realización de otros TFG.

1. **Optimización de algoritmos**: se pueden investigar y mejorar los algoritmos utilizados en este proyecto. Concretamente se desea estudiar enfoques más eficientes o precisos que podrían aplicarse.

2. **Generalización a otros dominios**: evaluar cómo los métodos desarrollados en este TFG se aplican a otros dominios o problemas similares.

3. **Exploración de nuevas podas**: nuevas podas que ayuden a la búsqueda de la solución óptima.

4. **Validación externa**: realizar una validación externa utilizando algún sistema como podría ser la propia página de Biwenger con los resultados de cada jornada.

5. **Interpretación de resultados**: profundizar en la interpretación de los resultados. ¿Qué implicaciones prácticas tienen? ¿Cómo se pueden aplicar en situaciones del mundo real?

6. **Algoritmos heurísticos y metaheurísticos**: explorar el uso de otros tipos de algoritmos como *Genetic Algorithms* [27] o *Ant Colony Optimization* [28].

# Bibliografía


[1] G. Curry, *The Making of Association Football: Two Decades Which Created the Modern Game*. Cambridge Scholars Publishing, 2020.

[2] E. Blakemore, "¿dónde surgió el fútbol? esto dicen los arqueólogos," *National Geographic*, 2018. [Online]. Disponible en: https://www.nationalgeographic.es/historia/donde-surgio-el-futbol-esto-dicen-los-arqueologos

[3] A. Fernandez, "Origen del fútbol: Cómo y dónde empezó," *CeleBreak*, 2022. [Online]. Disponible en: https://celebreak.com/es/blog/origen-del-futbol/

[4] F. Martialay, *Implantación del profesionalismo y nacimiento de la Liga*. CIHEFE, 1996.

[5] D. Heitner, "The hyper growth of daily fantasy sports is going to change our culture and our laws," *Forbes: SportsMoney*, 2015.

[6] A. Kaplan, *Artificial intelligence, business and civilization: Our fate made in machines*. Routledge, 2022.

[7] U. M. School, "Inteligencia artificial fútbol," *Unisport Management School*, 2023. [Online]. Disponible en: https://unisport.es/inteligencia-artificial-futbol/

[8] P. T. García, "Automanager, la primera plataforma de inteligencia artificial del fantasy en españa," *La Opinión de Málaga*, 2022. [Online]. Disponible en: https://www.laopiniondemalaga.es/malaga/2022/02/18/automanager-primera-plataforma-inteligencia-artificial-app-malaga-62880863.html

[9] Olocip, "Biwenger aplicará la inteligencia artificial de olocip en el mundo fantasy," *Olocip*, 2021. [Online]. Disponible en: https://olocip.com/biwengerolocip/

[10] Z. Cataldi, F. Lage, R. Pessacq, and R. García Martínez, "Ingeniería de software educativo," in *Proceedings del V Congreso Internacional de Ingeniería Informática*, 1999, pp. 185–199.

[11] C. G. Aros, "Rup: Metodología en los sistemas y aplicaciones basadas en la web," *Avances: Investigacion en Ingeniería*, vol. 1, no. 8, pp. 83–87, 2008.

[12] R. Galo Fariño, "Modelo espiral de un proyecto de desarrollo de software," *Obtenido de http://www. ojovisual. net/galofarino/modeloespiral. pdf*, 2011.

[13] J. Z. Gamboa, "Evolución de las metodologías y modelos utilizados en el desarrollo de software," *INNOVA Research Journal*, vol. 3, no. 10, pp. 20–33, 2018.

[14] A. Canavese Arbona, "Diseño de identidades digitales: metodología iterativa para la creación y desarrollo de marcas," Ph.D. dissertation, Universitat Politècnica de València, 2023.

[15] S. Universidades, "Metodologías de desarrollo de software: ¿qué son?" *Santander Universidades*, December 2020. [Online]. Disponible en: https://www.becas-santander.com/es/blog/metodologias-desarrollo-software.html

[16] M. Cohn, *User stories applied: For agile software development.* Addison-Wesley Professional, 2004.

[17] Y. Dodge, *The Oxford dictionary of statistical terms.* Oxford University Press, USA, 2003.

[18] D. Cuadrado Álvarez, A., S. Cavero, and E. Pardo, "Desarrollo de una herramienta software para la extracción y análisis de datos deportivos," Trabajo Fin de Grado, E.T.S. de Ingeniería Informática, 6 2024.

[19] Python, *Python 3.12.3 documentation*, 2024. [Online]. Disponible en: https://docs.python.org/3/

[20] NumPy, *NumPy Documentation*, 2022. [Online]. Disponible en: https://numpy.org/doc/

[21] MongoDB, *MongoDB Documentation*, 2023. [Online]. Disponible en: https://www.mongodb.com/docs/

[22] Pandas, *pandas documentation*, 2024. [Online]. Disponible en: https://pandas.pydata.org/docs/

[23] IBM, *Formato JSON (JavaScript Object Notation)*, March 2021. [Online]. Disponible en: https://www.ibm.com/docs/es/integration-designer/8.5.5?topic=formats-javascript-object-notation-json-format

[24] P. Van Beek, "Backtracking search algorithms," in *Foundations of artificial intelligence.* Elsevier, 2006, vol. 2, pp. 85–134.

[25] El libro de Python, *Recursividad*, 2024. [Online]. Disponible en: https://ellibrodepython.com/recursividad

[26] I. Verenich, H. Nguyen, M. La Rosa, and M. Dumas, "White-box prediction of process performance indicators via flow analysis," in *Proceedings of the 2017 International Conference on Software and System Process*, 2017, pp. 85–94.

[27] S. Sivanandam, S. Deepa, S. Sivanandam, and S. Deepa, *Genetic algorithms.* Springer, 2008.

[28] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE computational intelligence magazine*, vol. 1, no. 4, pp. 28–39, 2006.