



Asistente para la Selección de Plantillas en EA SPORT FC Ultimate Team

Fecha: 02/7/2025

Tabla de contenidos

1. INTRODUCCIÓN	1
1.1. Objetivos	2
1.2. Planificación	2
1.3. Presupuesto	5
1.3.1. Inventario de Recursos	5
1.3.2. Costes Estimados	12
1.3.3. Fuentes de Ingresos	20
1.3.4. Viabilidad del Modelo de Negocio	22
1.4. Estructura de la memoria	23
2. ANTECEDENTES	25
2.1. Introducción al contexto	25
2.1.1. Conocimientos Básicos sobre Fútbol	25
2.1.2. FIFA	30
2.2. Toma de Decisión	38
2.2.1. Introducción a la Toma de Decisión	39
2.2.2. Toma de Decisión Multiatributo	41
2.2.3. TOPSIS (Technique for Order of Preference by Similarity to Ideal Solution)	42

2.2.4. DSS	45
2.3. Optimización por Colonias de Hormigas (ACO)	48
2.3.1. Sistema de Hormigas (AS)	49
2.3.2. Sistema de Colonias de Hormigas (ACS)	52
2.4. Asistentes Virtuales	53
2.4.1. Introducción a los Asistentes Virtuales	54
2.4.2. Historia de los Asistentes Virtuales	55
2.4.3. RAG	57
2.4.4. Agentic RAG	59
3. INVESTIGACIÓN DE USUARIO	65
3.1. Contenidos Populares	65
3.2. Aproximación al usuario	67
3.3. Herramientas utilizadas por los usuarios	67
3.3.1. Descripción de las herramientas	67
3.3.2. Limitaciones detectadas en las herramientas existentes	68
3.4. Modelado del Usuario	69
3.4.1. Introducción al Modelado del Usuario	69
3.4.2. Arquetipo 1: El Estudiante	70
3.4.3. Arquetipo 2: La Madre	71
3.4.4. Arquetipo 3: El Jugador Competitivo	72
3.4.5. Arquetipo 4: El Entrenador	73
4. DESARROLLO DEL PROYECTO	75
4.1. Descripción del Proyecto	75

4.2. Análisis del Problema	76
4.2.1. Identificación y Diagnóstico del Problema	76
4.2.2. Definición de los elementos del Problema	77
4.2.3. Definición de las Alternativas	78
4.2.4. Formalización de los elementos del problema para su procesa- miento	79
4.3. Especificación de Requisitos	80
4.3.1. Requisitos Funcionales	80
4.3.2. Requisitos No Funcionales	81
4.4. Análisis del Sistema	82
4.4.1. Arquitectura del Sistema	83
4.4.2. Casos de Uso	84
4.5. Diseño del MRAD	90
4.5.1. Conjunto de Scrapers	90
4.5.2. Base de Datos Distribuida	95
4.6. Diseño del MGP	100
4.6.1. Servidor Web	100
4.6.2. Servidor Back-end	104
4.6.3. Servidor Redis	107
4.7. Diseño del MRT	108
4.7.1. Servidores Worker	109
4.7.2. Base de Datos Local Reducida	112
4.8. Diseño del MA	113
4.8.1. Servidores RAG	113

4.8.2. Servidor de Almacenamiento de Documentos	119
4.8.3. Modelo de Lenguaje Natural	120
4.8.4. Colección de Documentos	121
4.9. Diseño del Sitio Web	122
4.9.1. Nombre de Dominio	122
4.9.2. Paleta de Colores	123
4.9.3. Fuentes Tipográficas	123
4.9.4. Etiquetado	124
4.9.5. Página Principal	125
4.9.6. Página de Creación de Plantillas	125
4.9.7. Ventana Modal de Comparación de Jugadores	129
4.10. Implementación	132
4.10.1. Herramientas utilizadas	132
4.10.2. TOPSIS	132
4.10.3. ACOS	134
5. ANÁLISIS DE RESULTADOS	141
5.1. Introducción	141
5.2. Consultar al Asistente	142
5.2.1. Consultar cuestiones sobre el juego	142
5.2.2. Crear un equipo Automáticamente	146
5.2.3. Evaluación de Plantillas	150
5.3. Creación de Equipos	153
5.3.1. Configurar Posiciones	154

6. CONCLUSIONES	161
6.1. Valoración Final del Proyecto	161
6.2. Trabajos Futuros	162
Bibliografía	V
A. MANUAL DE INSTALACIÓN	VI
A.1. Software Común	VI
A.1.1. Windows Subsystem for Linux (WSL)	VI
A.1.2. Visual Studio Code	VII
A.1.3. Node.js y npm	VII
A.1.4. Python	VIII
A.2. MRAD	VIII
A.2.1. Base de Datos Distribuida	VIII
A.2.2. Conjunto de Scrappers	XIII
A.3. MGP	XIII
A.3.1. Servidor Web	XIII
A.3.2. Servidor Back-end	XIV
A.3.3. Servidor Redis	XV
A.4. MRT	XVI
A.4.1. Base de Datos Local Reducida	XVI
A.4.2. Servidores Worker	XVII
A.5. MA	XVII
B. Manual de Usuario	XIX
B.1. Página Principal	XIX

B.2. Página de Creación de Plantillas	XIX
B.3. Ventana Modal de Comparación de Jugadores	XXIII
B.4. Ventana del Asistente	XXVIII

Lista de figuras

1.1. Presupuesto según número de usuarios	22
2.1. Fotografía de un campo de fútbol	26
2.2. Fotografía de una portería	27
2.3. Fotografía de un balón de fútbol	27
2.4. Fotografía de un árbitro	27
2.5. Fotografía de un equipo de fútbol	28
2.6. Formación 4-4-2.	29
2.7. Primer videojuego de la saga FIFA	31
2.8. EA Sports FC 25	31
2.9. Carta de la modalidad FC Ultimate Team	33
2.10. Estilos de juego	37
2.11. Estadísticas de una plantilla de Ultimate Team	39
2.12. Representación gráfica del funcionamiento de TOPSIS	42
2.13. Arquitectura genérica de un DSS	47
2.14. Representación gráfica del comportamiento de las hormigas [1]	49
2.15. Algoritmo ACO [2]	51
2.16. Logo de ChatGPT	54

2.17.Eliza en Emacs	55
2.18.Asistente Virtual de Microsoft Clippy	56
2.19.Logo de Siri	57
2.20.Esquema básico del contexto de un SRI	58
2.21.Funcionamiento de un RAG	59
2.22.Bucle principal de un Agentic RAG	63
3.1. Arquetipo 1: El estudiante	71
3.2. Arquetipo 2: La madre	72
3.3. Arquetipo 3: El Jugador Competitivo	73
3.4. Arquetipo 4: El Entrenador	74
4.1. Logo de FC Assistant	76
4.2. Estructura del problema de toma de decisión	77
4.3. Diagrama de interacción del sistema	84
4.4. Diagrama de caso de uso general	85
4.5. Diagrama de caso de uso: Crear un equipo	86
4.6. Diagrama de caso de uso: Configurar una posición	88
4.7. Diagrama de caso de uso: Consultar Asistente	90
4.8. Sitio web FUT.GG	91
4.9. Logo de Urban VPN	92
4.10.Logo de MongoDB	96
4.11.Arquitectura de la Base de Datos	97
4.12.Esquemas de las colecciones de la base de datos	98
4.13.Arquitectura de Angular	101

4.14. Diagrama del Funcionamiento Lógico del Servidor Web	103
4.15. Logo de NestJs	105
4.16. API que proporciona el Back-end	107
4.17. Funcionamiento de los Servidores RAGs	118
4.18. Sitio web 3 Sided Cube	123
4.19. Fuente tipográfica: Bangers	124
4.20. Fuente tipográfica: Mouse Memoirs	124
4.21. Wireframe de la página principal	126
4.22. Wireframe de la página de creación de plantillas	127
4.23. Wireframe de la ventana modal de comparación de jugadores	129
5.1. Plantilla generada automáticamente para jugar al contraataque con la formación 4-4-2 (20000 monedas)	146
5.2. Plantilla generada automáticamente para presionar desde arriba con la formación 4-4-2 (20000 monedas)	147
5.3. Plantilla generada automáticamente para mantener la posesión con la formación 4-4-2 (20000 monedas)	149
5.4. Comparación gráfica de Nico y Sávio para un sistema de posesión	154
5.5. Prioridades configuradas por el asistente para la posición de LM	155
5.6. Plantilla para jugar al contraataque con la formación 4-4-2 (27000 monedas) después de realizar modificaciones manuales	160
A.1. Consulta de la versión de Ubuntu instalada en WSL	VII
B.1. Página principal de FC Assistant	XX
B.2. Página de creación de plantillas de FC Assistant	XX
B.3. Gráfico de la química en FC Assistant	XXI

B.4. Representación gráfica de la plantilla en FC Assistant	XXI
B.5. Menú de opciones de una posición de plantilla en FC Assistant	XXII
B.6. Sección para construir automáticamente el equipo en FC Assistant	XXIII
B.7. Sección de ajustes en FC Assistant	XXIII
B.8. Ventana modal sin haber seleccionado un jugador en FC Assistant	XXIV
B.9. Ventana modal comparando jugadores en FC Assistant	XXIV
B.10. Ventana modal gráfico comparativo en FC Assistant	XXV
B.11. Añadir importancias en ventana modal para una posición en FC Assistant	XXVI
B.12. Importancias añadidas para una posición en FC Assistant	XXVI
B.13. Gráfico de importancias en FC Assistant	XXVII
B.14. Añadir filtros en la ventana modal para una posición en FC Assistant	XXVII
B.15. Ventana de conversación con el asistente en FC Assistant	XXVIII

Lista de tablas

1.1. Comparativa de precios entre almacenamiento HDD y SSD para Bunny Storage	7
1.2. Perfiles profesionales, niveles y salarios estimados según BOE	12
1.3. Comparativa de modelos en términos de latencia y costo total por 1000 consultas.	16
1.4. Distribución de personal, salarios mensuales y motivo de contratación relacionado con recursos hardware	18
1.5. Tabla Gastos Variables.	19
1.6. Tabla Gastos Mensuales.	19
1.7. Costo total de puesta en marcha	20
1.8. Paquetes de consultas y beneficios obtenidos	22
4.1. Coeficientes de similitud por normalización	111
4.2. Tecnologías utilizadas en cada componente del sistema	132
5.1. Prioridades tácticas por posición en esquema 4-4-2 para contraataque .	147
5.2. Prioridades tácticas por posición en esquema 4-4-2 con presión alta . .	148
5.3. Sentido de la asignación de prioridades en esquema 4-4-2 orientado a posesión	150
5.4. Evaluación resumida de la plantilla (4-4-2 Contraataque, 20000 monedas)	151
5.5. Evaluación resumida de la plantilla (4-4-2 Presión alta, 20000 monedas)	152

5.6. Evaluación resumida de la plantilla (4-4-2 Posesión, 20000 monedas) .	153
5.7. Comparativa técnica entre Nicholas Williams y Sávio Moreira (LM en formación 4-4-2)	154
5.8. Comparativa LM: Paula Fernández Jiménez vs Sávio Moreira de Oliveira	156
5.9. Comparativa LM: Thomas Lemar vs Sávio Moreira de Oliveira	157
5.10. Comparativa LM: Thomas Lemar vs Paula Fernández (Formación 4-4-2)	157
5.11. Comparativa LM: Lauren Hemp vs Thomas Lemar (Formación 4-4-2) .	158

Lista de algoritmos

- 1. TOPSIS 133
- 2. ACOS 135
- 3. Algoritmo de Búsqueda Local 137

Lista de listados de código

4.1. Función objetivo	138
A.1. Consulta de la versión instalada de Ubuntu en WSL	VII
A.2. Comandos para instalar Visual Studio Code en Ubuntu desde terminal	VII
A.3. Comando para ejecutar Visual Studio Code	VII
A.4. Instalación de Node.js y npm en Ubuntu	VIII
A.5. Comprobación de versiones instaladas de Node.js y npm	VIII
A.6. Comandos para instalar Python 3 y pip en Ubuntu	VIII
A.7. Verificación de versiones de Python 3 y pip	VIII
A.8. Comandos para instalar Docker en Ubuntu	IX
A.9. Descargar imagen oficial de MongoDB	IX
A.10.Crear red personalizada para Docker	IX
A.11.Crear servidores de configuración MongoDB	IX
A.12.Inicializar replica set para servidores de configuración	IX
A.13.Crear nodos de shards MongoDB	X
A.14.Inicializar replica sets de shards	X
A.15.Lanzar router Mongo para el cluster	XII
A.16.Añadir shards al router mongos	XII
A.17.Instalación de bibliotecas necesarias de Python	XIII
A.18.Ejecutar algoritmo de web scrapping	XIII
A.19.Instalación de Angular CLI	XIII
A.20.Instalación de dependencias del proyecto Angular	XIV
A.21.Compilación del proyecto Angular en modo producción	XIV
A.22.Instalación de NestJS CLI	XIV
A.23.Instalación de dependencias del servidor NestJS	XIV
A.24.Ejemplo de configuración de constantes para el servidor back-end	XIV
A.25.Compilación del proyecto NestJS	XV
A.26.Ejecución del servidor NestJS en producción	XV
A.27.Descarga de la imagen oficial de Redis desde Docker Hub	XV
A.28.Ejecución de un contenedor Redis en segundo plano	XV
A.29.Parada del contenedor Redis	XVI

A.30.Reinico del contenedor Redis	XVI
A.31.Ejemplo de ejecución del script de extracción de datos	XVII
A.32.Configuración del archivo <code>.env</code> para servidores worker	XVII
A.33.Instalación de dependencias para el servidor MA	XVIII
A.34.Ejemplo de ejecución del servidor MA	XVIII

Capítulo 1

INTRODUCCIÓN

En las últimas décadas, la industria de los videojuegos ha crecido de forma exponencial y se ha consolidado como una de las principales formas de entretenimiento a nivel mundial [3]. Dentro de este amplio contexto, los videojuegos deportivos, en particular las franquicias de fútbol, han logrado captar la atención de millones de jugadores [4], ofreciendo experiencias cada vez más realistas y sofisticadas. Estas propuestas se asemejan más a simuladores de fútbol que a videojuegos tradicionales [5, 6].

El principal referente en el ámbito de los videojuegos de fútbol es EA SPORTS FC 25, comúnmente conocido como FC 25, galardonado en los Game Awards 2024 como el mejor juego en la categoría de deportes [7].

Dentro de sus distintas modalidades, una de las más populares entre los aficionados al fútbol virtual es FC Ultimate Team¹, que ofrece a los jugadores la posibilidad de crear y gestionar sus propios equipos mediante la adquisición de cartas de futbolistas. Sin embargo, la complejidad que implica conformar plantillas óptimas, teniendo en cuenta aspectos como la química, las estadísticas individuales y la dinámica del mercado, puede suponer un desafío considerable, especialmente para quienes se inician en la franquicia.

El objetivo de este Trabajo de Fin de Grado es precisamente el desarrollo de un sistema de soporte a la decisión que facilite la creación de plantillas en FC Ultimate Team. La herramienta propuesta busca simplificar el proceso de selección y organización de jugadores, incluso para usuarios con conocimientos limitados sobre teoría futbolística.

Para alcanzar este propósito, se desarrollará un sitio web con una interfaz dinámi-

¹<https://www.ea.com/es/games/ea-sports-fc/fc-25/features/ultimate-team>

ca orientada a la resolución de problemas de toma de decisiones, diseñada para ser accesible al usuario promedio, y se complementará con un back-end que incorporará diversas soluciones basadas en inteligencia artificial (IA), que tienen la finalidad de apoyar, explicar, simplificar y automatizar este complejo proceso de toma de decisiones.

1.1. Objetivos

- Revisar los conceptos relacionados con la toma de decisiones inteligentes para seleccionar jugadores y con EA SPORTS FC Ultimate Team.
- Identificar los atributos de los jugadores en EA SPORTS FC Ultimate Team y analizar su importancia en el juego.
- Extraer información de los jugadores incluidos en EA SPORTS FC Ultimate Team.
- Diseñar e implementar un asistente que permita definir y resolver la selección de jugadores adecuados.
- Analizar los resultados obtenidos.
- Redactar una memoria que recoja todo el trabajo desarrollado.

1.2. Planificación

El desarrollo de un sistema de soporte a la decisión que logre asistir a los usuarios en la creación de plantillas de FC Ultimate Team, es un proyecto complejo y ambicioso que requiere flexibilidad para adaptarse a cambios e incertidumbres. Para gestionar esta complejidad, se ha adoptado una metodología ágil basada en *Scrum* [8], seleccionada por su enfoque iterativo, incremental y adaptable, que permite priorizar objetivos estratégicos y ajustar el plan según las necesidades reales del proyecto.

Adaptación de Scrum a un Proyecto Individual

Aunque Scrum está ideado originalmente para equipos multidisciplinares, su estructura puede adaptarse eficazmente a entornos de desarrollo individuales. En este TFG, los sprints mensuales han sido el marco principal de trabajo, permitiendo dividir

el proyecto en ciclos manejables, con entregables parciales funcionales y revisables. Los principios fundamentales que se han seguido son:

- Desarrollo incremental: se construyó el sistema por capas funcionales, que podían ampliarse sin comprometer la estabilidad del producto.
- Backlog priorizado: se estructuró un listado de tareas ordenado por su impacto, necesidad y viabilidad, separando claramente las funcionalidades imprescindibles de aquellas secundarias y destacables, utilizando técnicas como *MoSCoW* (*Must have, should have, could have, wont't have*) [9], garantizando así que los objetivos principales se completen primero.
- Evaluación continua: tras la finalización de cada iteración se realizó una revisión de los avances y se volvieron a planificar las tareas restantes si era necesario.

A continuación, se expone la planificación temporal del proyecto, desglosada en iteraciones y actividades principales.

Inicialización (15 de noviembre - 1 de enero)

- Fase de configuración: investigación sobre el contexto, búsqueda de herramientas hardware y software a utilizar, investigación sobre el usuario con el objetivo de conocer sus necesidades.
- Diseño e implementación de la base de datos.
- Diseño e implementación del algoritmo de web scrapping.

Iteración 1: Sitio Web (1 de enero - 1 febrero)

- Diseño básico del sitio web.
- Implementación básica del sitio web.

Iteración 2: Back-end (1 febrero - 1 marzo)

- Implementación básica del back-end.
- Implementación básica de los servidores workers.

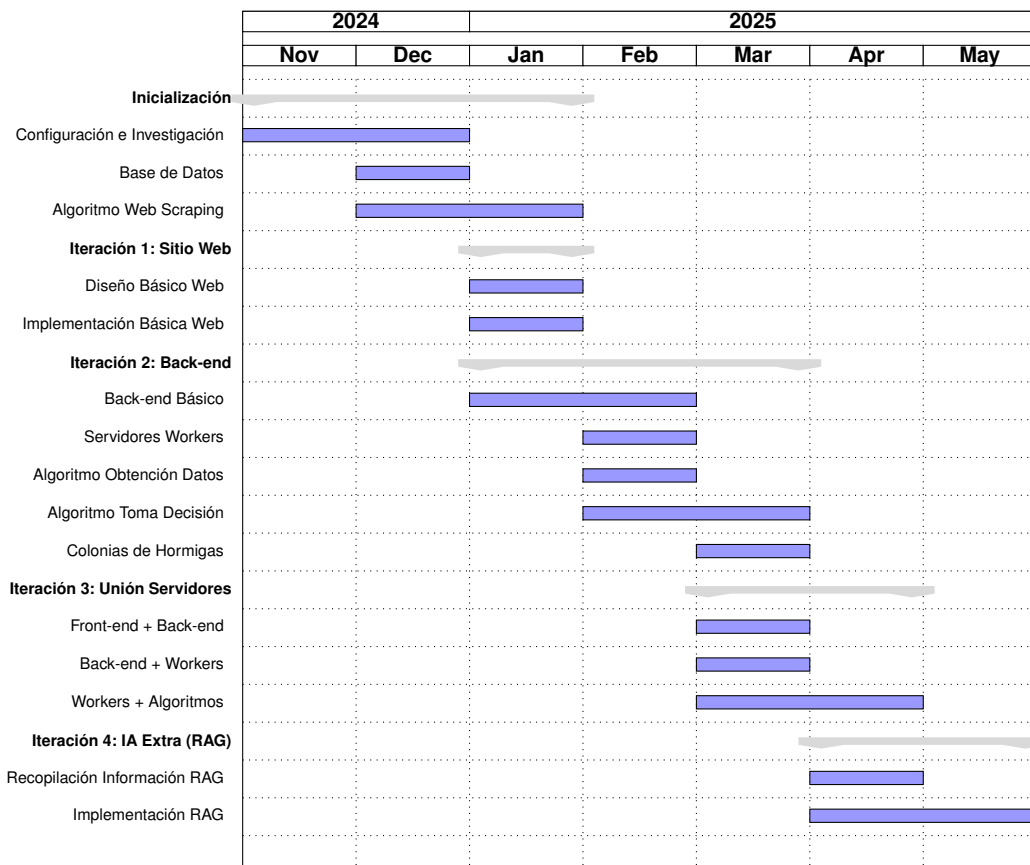
- Diseño e implementación del algoritmo de obtención de datos por los workers.
- Implementación del algoritmo de toma de decisión.
- Implementación del algoritmo de optimización.

Iteración 3: Unión de los Servidores (1 marzo - 1 abril)

- Unión del front-end con el back-end.
- Unión del back-end con los servidores workers.
- Unión de los servidores workers con los algoritmos.

Iteración 4: Incorporación de IA Extra (1 abril - 1 mayo)

- Búsqueda y recolección de Información para la implementación del RAG.
- Diseño e implementación del RAG.



1.3. Presupuesto

A continuación, se presenta un presupuesto detallado que incluye los recursos necesarios, la estimación de los costes asociados a dichos recursos, así como una propuesta de modelo de negocio en la que se describen las diferentes fuentes de ingresos.

Es importante tener en cuenta que algunos componentes contratados están presupuestados en dólares estadounidenses. Dado que los proveedores facturan en esta moneda, para simplificar el proceso, los cálculos intermedios se realizarán en dólares, efectuándose la conversión a euros únicamente al presentar los resultados finales.

Para evitar confusiones, los valores en dólares están expresados con notación europea (coma decimal y punto para los miles), por lo que deben interpretarse del mismo modo que los valores en euros, aunque corresponden a una moneda distinta.

1.3.1. Inventario de Recursos

Teniendo en mente el objetivo de elaborar un presupuesto integral, se ha confeccionado un inventario exhaustivo de los recursos indispensables para el desarrollo, implementación y mantenimiento del sistema.

Recursos Software

- **Frameworks y Herramientas de Desarrollo:** las siguientes herramientas de software son de código abierto, por lo que no generan costes asociados:
 - Angular².
 - NestJS³.
 - NetBeans⁴.
 - Visual Studio Code⁵.
- **Sistemas Operativos**
 - Distribuciones Linux de código abierto: Ubuntu⁶ 24.04.2 LTS, disponible de

²<https://angular.dev/>

³<https://nestjs.com/>

⁴<https://netbeans.apache.org/>

⁵<https://code.visualstudio.com/>

⁶<https://ubuntu.com/download>

forma gratuita y sin costes de licencia.

■ Plataformas y Servicios

• Servicios de almacenamiento y optimización de imágenes en la nube:

se contemplan soluciones tecnológicas como :

- **Bunny Storage**⁷: servicio de almacenamiento de objetos replicado globalmente. Permite cargar archivos a través de FTP, SFTP, API HTTP o mediante un gestor web, con replicación automática en múltiples regiones (hasta 15), lo que reduce significativamente la latencia y mejora la disponibilidad. Está especialmente diseñado para alojar contenido destinado a su distribución global de forma eficiente.
- **Bunny CDN**⁸: red global de distribución de contenido (CDN), con más de 100 ubicaciones en seis continentes. Facilita la entrega de recursos estáticos acercándolos a los usuarios finales, lo que se traduce en tiempos de carga optimizados y una mejor experiencia de navegación. Ofrece precios por tráfico consumido, sin tarifas adicionales por peticiones, y garantiza una latencia media en torno a los 25 ms.
- **Bunny Optimizer**⁹: herramienta especializada en la optimización y transformación dinámica de imágenes y recursos web. Sus funcionalidades incluyen el redimensionado, compresión y conversión automática a formatos optimizados como WebP (logrando reducciones de tamaño de hasta un 80 %).

Este servicio se empleará exclusivamente para almacenar las imágenes del sitio web. Dado que, por el momento, no se contempla su internacionalización, es suficiente con contratar únicamente el servicio de Bunny Storage.

El volumen de datos almacenados puede estimarse a partir de la Ecuación 1.1, donde el cálculo depende de:

- Número estimado de imágenes de perfil de usuarios y jugadores.
- Resolución, formato y profundidad de color de las imágenes, factores que determinan su tamaño en memoria.

$$V_{\text{DAlmacenados}} \text{ (GB)} = \frac{N_{\text{ITotal}} \times (A \times H) \times \left(\frac{B}{8}\right)}{1024^3} \quad (1.1)$$

⁷<https://bunny.net/storage/>

⁸<https://bunny.net/cdn/>

⁹<https://bunny.net/optimizer/>

$V_{\text{DAlmacenados}}$: Volumen estimado de datos almacenados (en gigabytes).

N_{ITotal} : Número total de imágenes almacenadas.

A : Ancho máximo de cada imagen (en píxeles).

H : Alto máximo de cada imagen (en píxeles).

B : Profundidad de color (en bits por píxel).

1024^3 : Conversión de bytes a gigabytes (1 GB = 1024^3 bytes).

- **Tipo de almacenamiento:** los diferentes tipos de almacenamiento y sus respectivos precios se pueden observar en la Tabla 1.1.

Tipo	Precio (\$/GB)	Notas
HDD	\$0,01/GB	Menor coste por GB. Latencia alta.
SSD	\$0,02/GB	Mayor velocidad de acceso.

Tabla. 1.1: Comparativa de precios entre almacenamiento HDD y SSD para Bunny Storage

Se contratará el servicio de almacenamiento SSD de Bunny Storage con el fin de garantizar una baja latencia en la recuperación de imágenes, con un coste unitario de \$0,02 por gigabyte almacenado. Por lo tanto, el coste total estimado destinado al almacenamiento de imágenes se puede calcular mediante la expresión mostrada en la Ecuación 1.2:

$$C_{\text{Alimagenes}}(\$) = V_{\text{DAlmacenados}} \times 0,02 \quad (1.2)$$

$C_{\text{Alimagenes}}$: Coste estimado dedicado al almacenamiento de imágenes (\$).

$V_{\text{DAlmacenados}}$: Volumen estimado de datos almacenados (GB).

0,02 : Coste por gigabyte de almacenamiento ocupado (\$/GB).

- **Número de regiones en las que se replican los datos:** Bunny Storage ofrece la posibilidad de replicar los datos en distintas regiones, como Madrid, Londres, Frankfurt o Praga, entre otras. No obstante, dado que el proyecto no contempla su internacionalización, se ha optado por contratar una única réplica en Madrid.
- **Mínimo de facturación mensual:** \$1.

Motores de almacenamiento vectorial: se utilizará Chromadb¹⁰, cuyo modelo de coste es por suscripción mensual y está condicionado por los siguientes factores:

- **Volumen de datos almacenados:** con un coste de \$0.33/GB, determinado por:
 - **Número y tamaño de vectores:** se considera un vector por documento y que cada vector tiene 384 dimensiones, con un tamaño de 4 bytes por dimensión, lo que da como resultado un tamaño total de 1536 bytes por vector. Estas cantidades se asumen por ser configuraciones habituales en modelos de embeddings. De este modo, el tamaño de cada vector puede estimarse mediante la Ecuación 1.3.

$$T_{\text{vector}}(\text{GB}) = \frac{D \times B}{F} = \frac{384 \times 4}{1024^3} = \frac{1536}{1024^3} = 1,43051 \times 10^{-6} \quad (1.3)$$

Donde:

T_{vector} : Tamaño de un vector en gigabytes (GB).

D : Número de dimensiones del vector (384).

B : Bytes por dimensión (4 bytes).

F : Factor de conversión de bytes a gigabytes (1024^3).

- **Número y tamaño de los metadatos de los documentos:** considerando el tamaño del texto del documento como parte del conjunto de metadatos, junto con los elementos adicionales asociados (identificador, fuente y fecha de creación), el tamaño total estimado de cada documento destinado al almacenamiento de metadatos se puede calcular mediante la Ecuación 1.4:

$$T_{\text{metadatos}}(\text{GB}) = \frac{N_{\text{caracteres}} + I + F + C}{1024^3} \quad (1.4)$$

Donde:

¹⁰<https://www.trychroma.com/home>

- $T_{\text{metadatos}}$: Tamaño total de los metadatos de un documento en gigabytes (GB).
- $N_{\text{caracteres}}$: Número de caracteres del texto del documento (1 byte/carácter, UTF-8).
- I : Tamaño del identificador del documento (36 bytes, UUID estándar).
- F : Tamaño de la fuente del documento (20 bytes).
- C : Tamaño de la fecha de creación (10 bytes).
- 1024^3 : Factor de conversión de bytes a gigabytes.

Por lo tanto, el tamaño total estimado de almacenamiento por documento se obtiene como la suma del tamaño ocupado por los metadatos y el correspondiente al vector de representación, tal y como se expresa en la Ecuación 1.5:

$$T_{\text{ADocumento}}(\text{GB}) = T_{\text{metadatos}} + T_{\text{vector}} \quad (1.5)$$

Donde:

- $T_{\text{ADocumento}}$: Tamaño total estimado de almacenamiento por documento (GB).
- $T_{\text{metadatos}}$: Tamaño de los metadatos asociados al documento (GB).
- T_{vector} : Tamaño del vector de representación del documento (GB).

- **Volumen de datos escritos:** el coste asociado a las operaciones de escritura es de \$2,50 por GB de datos escritos. Este coste se desglosa en dos componentes:
 - **Escritura inicial de la colección base:** corresponde al volumen de datos necesario para almacenar el conjunto inicial de documentos. El coste estimado se calcula mediante la Ecuación 1.6:

$$C_{\text{Escritura Inicial}}(\$) = N_{\text{Documentos Base}} \times T_{\text{ADocumento}} \times 2,50 \quad (1.6)$$

- **Actualización mensual estimada:** hace referencia al coste mensual derivado de la incorporación de nuevos documentos (por ejemplo, noticias relacionadas con FC 25). El cálculo se realiza según la Ecuación 1.7:

ción 1.7:

$$C_{\text{Escritura Mensual}}(\$) = N_{\text{Documentos Nuevos}} \times T_{\text{ADocumento}} \times 2,50 \quad (1.7)$$

Donde:

- $C_{\text{Escritura Inicial}}$: Coste estimado por la escritura inicial de la colección (\$).
- $C_{\text{Escritura Mensual}}$: Coste estimado por las escrituras mensuales (\$).
- $N_{\text{Documentos Base}}$: Número de documentos incluidos en la colección inicial.
- $N_{\text{Documentos Nuevos}}$: Número estimado de nuevos documentos añadidos mensualmente.
- $T_{\text{ADocumento}}$: Tamaño total estimado de almacenamiento por documento (GB).
- 2,50 : Coste por gigabyte de datos escritos (\$/GB).

- **Volumen de consultas:** el modelo de tarificación aplicado a las operaciones de consulta distingue entre coste por volumen de consultas enviadas (\$0.0075/GB) y coste por volumen de respuestas (\$0,09/GB).
- **API de modelos LLM (Large Language Model):** se empleará el servicio proporcionado por OpenRouter¹¹, cuya suscripción mensual presenta un coste variable en función de los siguientes factores:
 - Volumen de tokens procesados, tanto de entrada como de salida.
 - Variante del modelo empleada, que afecta el coste por token.
 - Uso de funcionalidades adicionales, tales como fine-tuning o generación de imágenes, que generan cargos extra.
- **Herramientas de gestión de proyecto:**
 - **GitHub**¹²: ofrece planes de pago con funcionalidades avanzadas, entre los cuales destacan:
 - **GitHub Pro:** repositorios privados ilimitados, mayor capacidad de almacenamiento y herramientas avanzadas.
 - **GitHub Team:** gestión colaborativa de equipos y control de accesos.
 - **GitHub Enterprise:** orientado a grandes organizaciones, con soporte dedicado y opciones personalizadas.

¹¹<https://openrouter.ai/>

¹²<https://github.com/>

- **Nifty**¹³: plataforma para gestión y planificación de tareas, cuyo coste depende del plan seleccionado y se ve influenciado por:
 - Número de usuarios o miembros del equipo.
 - Capacidad de almacenamiento requerida.
 - Número de proyectos activos gestionados simultáneamente.

Recursos Hardware

- **Servidor web**: responsable del alojamiento y distribución de la interfaz cliente.
- **Servidor back-end**: gestiona la lógica de negocio, proporciona datos al servidor web y actúa como intermediario con los distintos componentes del sistema.
- **Servidor Redis**: empleado como sistema de mensajería y coordinación entre el servidor back-end, los servidores workers y los servidores RAG.
- **Conjunto de servidores RAG**: encargados de la resolución de consultas mediante técnicas de recuperación aumentada por generación, utilizando como base un corpus documental específico.
- **Conjunto de servidores workers**: destinados al procesamiento paralelo y eficiente de tareas computacionalmente intensivas.

Recursos Humanos

El desarrollo e implementación del sistema requiere la participación de profesionales especializados en distintas áreas, con el objetivo de asegurar una adecuada ejecución, despliegue y mantenimiento de la solución. En la tabla 1.2 se describen los perfiles clave necesarios, así como sus retribuciones salariales, calculadas conforme a las referencias establecidas en el BOE [10].

¹³<https://niftypm.com/>

Perfil Profesional	Responsabilidades	Nivel y Salario Estimado (anual) / Fuente
Desarrolladores Front-end	Implementación y mantenimiento de la interfaz de usuario. Dominio de Angular.	Programador/a Nivel I 14.463,82 €
Desarrolladores Back-end	Desarrollo de la lógica del servidor, gestión de bases de datos, conexión con servicios externos. Recomendable experiencia en NestJS y MongoDB.	Analista Programador/a I 17.452,20 €
Especialistas en Bases de Datos	Diseño, optimización y mantenimiento de sistemas de almacenamiento relacional, NoSQL y motores vectoriales como ChromaDB.	Analista Programador/a I 17.452,20 €
Especialistas en IA	Mantenimiento, optimización de los servidores RAG.	Analista Programador/a I 17.452,20 €
Diseñadores UX/UI	Definición y mantenimiento de una experiencia de usuario coherente, intuitiva y accesible, alineada con los objetivos del proyecto.	Programador/a Nivel I 14.463,82 €
Diseñadores Gráficos	Creación de elementos visuales y gráficos del sistema (iconografía, ilustraciones, identidad corporativa), asegurando coherencia estética y funcionalidad visual.	Programador/a Nivel I 14.463,82 €
Editores Web	Gestión, actualización y optimización de contenidos web. Garantizar calidad, coherencia y presentación adecuada, además de correcta integración con el front-end.	Programador/a Nivel I 14.463,82 €

Tabla. 1.2: Perfiles profesionales, niveles y salarios estimados según BOE

1.3.2. Costes Estimados

Amortización Inicial

Se contempla una inversión inicial destinada a la adquisición de los equipos, servidores y servicios necesarios, cuya amortización se distribuye a lo largo de un periodo estimado de 120 meses (10 años). La amortización mensual se calcula mediante la

Ecuación 1.8.

$$A = \frac{C_{\text{Inicial}}}{T} \quad (1.8)$$

A : Amortización mensual estimada (€).

C_{Inicial} : Coste total de adquisición del hardware (€).

T : Periodo de amortización, expresado en meses.

Los costes estimados correspondientes a la inversión inicial asociada a los distintos recursos son los siguientes:

- **Servidor web:** 3.000 €
- **Servidor back-end:** 4.000 €
- **Servidor Redis:** 2.000 €
- **Servidores RAG** (mínimo 2 unidades): 6.000 € (3.000 € por unidad)
- **Servidores workers** (mínimo 2 unidades): 6.000 € (3.000 € por unidad)
- **Inversión inicial en la escritura de datos en la base de datos vectorial:** se han almacenado un total de 647 documentos, cada uno con una longitud aproximada de 500 caracteres. El coste estimado de esta operación calculado según la Ecuación 1.6, asciende a 0,0032 €, cantidad despreciable a efectos prácticos.

El coste total inicial, C_{inicial} , se calcula mediante la Ecuación 1.9 y su amortización mensual aproximada A se obtiene dividiendo dicho coste entre 120 meses, como se detalla en la Ecuación 1.10.

$$C_{\text{inicial}} = 3,000 + 4,000 + 2,000 + 6,000 + 6,000 = 21,000\text{€} \quad (1.9)$$

$$A = \frac{C_{\text{inicial}}}{120} = \frac{21,000 \text{€}}{120} = 175 \text{€/mes} \quad (1.10)$$

Costes mensuales asociados al almacenamiento de imágenes

Es posible estimar el coste de almacenamiento considerando un escenario con los valores máximos de recursos previstos que proporcione el suficiente margen de tiempo como para rectificar:

- Número de imágenes de jugadores: 20000.
- Número de imágenes de usuarios: 10000.
- Resolución máxima de las imágenes de los jugadores: 800 por 1118 píxeles.
- Resolución máxima de las imágenes de los usuarios: 400 por 400 píxeles.
- Profundidad de color: 32 bits.

El coste calculado bajo las suposiciones previamente establecidas, y empleando la Ecuación 1.2, es de \$2,05/mes.

Costes mensuales asociados a la escritura en Chromadb

Es importante considerar que:

- La colección inicial consta de 647 documentos.
- Cada documento contiene aproximadamente 500 caracteres.

En consecuencia, el coste mensual asociado al almacenamiento de los documentos, calculado mediante la Ecuación 1.5, en función del número total de documentos y del coste por gigabyte en la base de datos vectorial (0,33\$/GB), asciende aproximadamente a 0,0032\$. El incremento de este gasto debido a la incorporación de nuevos documentos podrá ser evaluado y tenido en cuenta en futuras revisiones presupuestarias.

Costo mensual derivado de las consultas al asistente

- **Costes derivados del uso de OpenRouter:** corresponde al gasto mensual estimado por el uso del modelo de lenguaje a través de OpenRouter. El coste se calcula mediante la Ecuación 1.11.

$$C_{\text{OpenRouter}}(\$) = \left(\frac{\text{Tokens}_{\text{Entrada}}}{10^6} \times P_{\text{entrada}} + \frac{\text{Tokens}_{\text{Salida}}}{10^6} \times P_{\text{salida}} \right) \times N_{\text{consultas}} \quad (1.11)$$

Donde:

$C_{\text{OpenRouter}}$: Coste mensual estimado por el uso de OpenRouter (\$).

$\text{Tokens}_{\text{Entrada}}$: Tokens de entrada procesados por consulta.

$\text{Tokens}_{\text{Salida}}$: Tokens de salida generados por consulta.

P_{entrada} : Precio por millón de tokens de entrada (\$).

P_{salida} : Precio por millón de tokens de salida (\$).

$N_{\text{consultas}}$: Número total de consultas al asistente en un mes.

10^6 : Factor de normalización (tokens a millones de tokens).

- **Costes derivados del uso de ChromaDB:** representa el coste mensual asociado a la búsqueda semántica de vectores en la base de datos ChromaDB. El cálculo se realiza mediante la Ecuación 1.12.

$$C_{\text{Chromadb}}(\$) = \left(\frac{4 \times \text{Tokens}_{\text{Entrada}}}{10^{12}} \times P_{\text{entrada}} + \frac{4 \times \text{Tokens}_{\text{Salida}}}{10^9} \times P_{\text{salida}} \right) \times N_{\text{consultas}} \quad (1.12)$$

Donde:

C_{Chromadb} : Coste mensual estimado por el uso de ChromaDB (\$).

$\text{Tokens}_{\text{Entrada}}$: Tokens de entrada utilizados por consulta.

$\text{Tokens}_{\text{Salida}}$: Tokens de salida devueltos por consulta.

P_{entrada} : Precio por TB de tokens de entrada (\$).

P_{salida} : Precio por GB de tokens de salida (\$).

$N_{\text{consultas}}$: Número total de consultas al sistema vectorial en un mes.

10^{12} : Conversión de tokens a terabytes (TB).

10^9 : Conversión de tokens a gigabytes (GB).

Es importante tener en cuenta que cada consulta realizada al asistente genera múltiples solicitudes tanto a la base de datos vectorial como a la API de OpenRouter.

En promedio, se estima que por cada interacción con el asistente se efectúan aproximadamente 20 peticiones a cada una de estas plataformas.

Por consiguiente, el coste estimado por consulta al asistente se calcula mediante la siguiente Ecuación 1.13.

$$C_{\text{Consulta}}(\$) = (C_{\text{OpenRouter}} + C_{\text{Chromadb}}) \times 20 \quad (1.13)$$

Donde:

C_{Consulta} : Coste total estimado de una consulta al asistente (en dólares).

$C_{\text{OpenRouter}}$: Coste medio de una petición a la API de OpenRouter (en dólares).

C_{Chromadb} : Coste medio de una petición a la base de datos vectorial Chromadb (en dólares).

20 : Número estimado de solicitudes realizadas por consulta al asistente.

Costo mensual por usuario

Modelo	Latencia	Coste Total/1000 consultas	Latencia Asistente
GPT-4o	0.53	\$20,0360003	10.6
GPT-4.1-mini	0.54	\$20,0360003	10.8
GPT-4.1 nano	0.36	\$5,0360003	7.2
GPT-4.1	0.56	\$100,0360003	11.2
GPT-3.5 Turbo	0.36	\$20,0360003	7.2
Google: Gemma 29B	0.38	\$4,0360003	7.6
Mistral 3B	0.19	\$0,8360003	3.8
Mistral 8B	0.25	\$2,0360003	5

Tabla. 1.3: Comparativa de modelos en términos de latencia y costo total por 1000 consultas.

El costo mensual por usuario corresponde al gasto estimado derivado del uso que este realice del asistente conversacional. Se establece un límite estándar de 1000 consultas mensuales por usuario, lo que equivale aproximadamente a 32 consultas diarias.

Para el cálculo de este coste, se asumen las siguientes condiciones promedio por consulta:

- **Tokens de entrada en OpenRouter:** 500 tokens por consulta.

- **Tokens de salida en OpenRouter:** 500 tokens por respuesta.
- **Tokens de entrada en Chromadb:** 500 tokens por consulta.
- **Tokens de salida en Chromadb:** 500 tokens por cada uno de los 10 documentos devueltos, resultando en un total de 5.000 tokens.

Bajo estas condiciones, el coste mensual estimado por usuario se determina mediante la Ecuación 1.14.

$$C_{\text{Usuario}}(\$) = C_{\text{Consulta}} \times 1,000 \quad (1.14)$$

Donde:

C_{Usuario} : Coste mensual estimado por usuario (\$).

C_{Consulta} : Coste medio de una consulta individual al asistente (\$).

1,000 : Número máximo de consultas mensuales permitidas por usuario.

El costo total por usuario varía en función del modelo de inteligencia artificial utilizado, como se muestra en la Tabla 1.3.

Se recomienda utilizar inicialmente el modelo *Mistral 3B*¹⁴ hasta completar la amortización inicial, dada su bajo coste aproximado de \$0,83 cada 1000 consultas por usuario. Posteriormente, se podrá considerar la actualización a un modelo de mayor costo, permitiendo así enfocar los ingresos en la recuperación de la inversión inicial.

¹⁴<https://openrouter.ai/mistralai/ministral-3b>

Coste empleados

Profesión	Número de Empleados	Salario Mensual (€)	Motivo
Desarrollador Front-end	3	1.205 €	Número suficiente para gestionar la interfaz cliente y agregar funcionalidades de forma continua.
Desarrollador Back-end	2	1.454 €	Empleados necesarios para cubrir el desarrollo y mantenimiento de la lógica de negocio en el servidor back-end.
Especialista en Bases de Datos	1	1.454 €	Profesional suficiente para diseñar y optimizar la gestión de datos, soportando las consultas de servidores back-end, workers y RAG.
Especialista en IA	1	1.454 €	Responsable de todo lo que requiera conocimientos de IA.
Diseñador UX/UI	1	1.205 €	Adecuado para definir la experiencia de usuario en el servidor web, asegurando coherencia y accesibilidad según los objetivos del proyecto.
Diseñador Gráfico	1	1.205 €	Suficiente para mantener la identidad visual en la interfaz alojada en el servidor web, garantizando coherencia estética y usabilidad.
Editor Web	1	1.205 €	Número adecuado para gestionar y actualizar contenidos integrados en la interfaz cliente, asegurando calidad y sincronización con el front-end.

Tabla. 1.4: Distribución de personal, salarios mensuales y motivo de contratación relacionado con recursos hardware

En la Tabla 1.4 se presenta un detalle de los perfiles profesionales requeridos para el proyecto, indicando el número estimado de empleados para cada puesto, el salario mensual aproximado y una breve descripción que justifica la necesidad de cada rol dentro del equipo de trabajo.

Por consiguiente, el gasto mensual en personal, conforme a la configuración detallada en la Tabla 1.4, es de 13.049,31€. Las retribuciones salariales, han sido extraídas de referencias establecidas en el BOE [10].

Resumen gastos variables

En la Tabla 1.5 se presenta un resumen de los gastos variables que deberán contemplarse. Estos gastos corresponden exclusivamente a las distintas consultas que los usuarios realicen al asistente.

Recurso	Gasto
Consultas Asistente	\$0,000836/Consulta = 0,00072€/Consulta

Tabla. 1.5: Tabla Gastos Variables.

Resumen gastos mensuales

Los gastos fijos que deberán considerarse mensualmente se recogen en la Tabla 1.6. En dicha tabla y en la Tabla 1.5 se ha realizado el cambio de moneda teniendo en cuenta que en ese momento un dólar estadounidense eran 0,87 euros.

Recurso	Gasto Mensual
Amortización	175€/mes
Empleados	13.049€/mes
Almacenamiento Imágenes	\$2,058744/mes = 1,7911€/mes
Almacenamiento Documentos	\$0,00316648/mes = 0,0028€/mes

Tabla. 1.6: Tabla Gastos Mensuales.

Costo total de puesta en marcha

El coste total de la puesta en marcha del proyecto se puede estimar como la suma del coste derivado de la amortización inicial y el importe correspondiente a las horas de trabajo del personal durante el periodo de desarrollo, cuya duración es de 24 semanas. Considerando que todos los días, a excepción de los fines de semana, son laborables, el total de días hábiles asciende a 120. A partir de estos datos, el coste total estimado se detalla en la Tabla 1.7.

Recurso	Coste Total (€)
Desarrollador Front-end (3 empleados)	14.265,69€
Desarrollador Back-end (2 empleados)	11.475,42€
Especialista en Bases de Datos (1 empleado)	5.737,71€
Especialista en IA (1 empleado)	5.737,71€
Diseñador UX/UI (1 empleado)	4.755,23€
Diseñador Gráfico (1 empleado)	4.755,23€
Editor Web (1 empleado)	4.755,23€
Servidor web	3.000,00€
Servidor back-end	4.000,00€
Servidor Redis	2.000,00€
Servidores RAG (2 unidades)	6.000,00€
Servidores workers (2 unidades)	6.000,00€
Total	72.482,2€

Tabla. 1.7: Costo total de puesta en marcha

1.3.3. Fuentes de Ingresos

El modelo de negocio¹⁵ propuesto se basa en un sistema híbrido de suscripción y pago por uso, concebido específicamente para ajustarse a las particularidades y demandas propias de la plataforma.

Esta estructura responde a la necesidad de evitar la inviabilidad económica que podría derivarse de un modelo de suscripción tradicional con acceso ilimitado, especialmente cuando los usuarios efectúan un elevado número de consultas a los servidores RAG. Cada consulta implica un coste inherente asociado al uso de modelos LLM externos y bases de datos vectoriales, lo que hace imprescindible un esquema que permita controlar y optimizar dichos gastos.

Con el fin de abordar esta problemática, se propone la creación de una moneda virtual propia, denominada *Assistant Coin*, la cual funcionará como medio exclusivo de pago para las consultas efectuadas a los sistemas RAG.

Por lo tanto, el modelo de negocio se estructura en tres niveles claramente diferenciados:

¹⁵Este esquema de monetización excede el ámbito del presente proyecto, pero se contempla su implementación en futuras iteraciones.

- **Modelos de Toma de Decisión:** acceso gratuito para todos los usuarios sin restricciones.
- **Creación Automática de Plantillas:** acceso ilimitado mediante suscripción mensual o anual.
- **Asistente (Consultas RAG):** el acceso se gestionará mediante el uso de *Assistant Coins*. Cada consulta tendrá un coste en esta moneda, calculado en función del precio real de acceso al modelo LLM a través de OpenRouter, incrementado por un margen de beneficio. La suscripción incluirá un paquete mensual de Assistant Coins que permitirá un uso razonable del asistente durante dicho periodo. En caso de superar este límite, los usuarios podrán adquirir paquetes adicionales adaptados a sus necesidades.

Definición del precio de la suscripción

El precio de suscripción debe ser asequible para el público objetivo. Se ha establecido un rango entre 5 y 9 dólares, ya que estas cantidades coinciden con los importes que los usuarios suelen estar dispuestos a pagar por suscripciones a plataformas de entretenimiento como Twitch¹⁶, Netflix¹⁷ o HBO¹⁸. Un precio superior podría resultar disuasorio, dado que el servicio está orientado a una actividad lúdica y no esencial.

El precio final de la suscripción se determina mediante la Ecuación 1.15, en la que se tienen en cuenta todos los costes asociados y un margen de beneficio.

$$P_{\text{suscripción}} = (C_{\text{Mensuales}} + (C_{\text{Consulta Asistente}} \times 1000)) \times (1 + B) \quad (1.15)$$

$P_{\text{suscripción}}$: Precio total de la suscripción mensual.

$C_{\text{Mensuales}}$: Cuota fija por mantenimiento y funcionalidades básicas.

$C_{\text{Consulta Asistente}}$: Coste por consulta al asistente

B : Margen de beneficio expresado en tanto por uno.

1,000 : Número máximo de consultas mensuales permitidas por usuario.

¹⁶<https://www.twitch.tv/>

¹⁷<https://www.netflix.com//>

¹⁸<https://play.max.com/>

Definición del precio para consultas adicionales al asistente

Se propone ofrecer diversos paquetes que incluyan diferentes cantidades de consultas adicionales disponibles para los usuarios. Un ejemplo ilustrativo de dichos paquetes se presenta en la Tabla 1.8.

Paquete	Consultas	% Beneficio	Precio (\$)	Beneficio (\$/€)
P. Principiante	500	10900 %	2,5	2,27/1,97
P. Entrenador	1000	690 %	5	4,32/3,75
P. Maestro del Juego	2150	310 %	7,5	5,57/4,84

Tabla. 1.8: Paquetes de consultas y beneficios obtenidos

1.3.4. Viabilidad del Modelo de Negocio

Finalmente se presenta una tabla que ilustra la viabilidad del modelo de negocio a largo plazo (ver Figura 1.1). Según el análisis, no se alcanzarán beneficios hasta contar con un mínimo de 4.000 usuarios suscritos, solo teniendo en cuenta las suscripciones como medio de obtención de ingresos. Sin embargo, a medida que la herramienta gane popularidad y crezca la base de suscriptores, los ingresos se incrementarán proporcionalmente. Esto permitirá reducir significativamente la proporción de la suscripción mensual destinada a cubrir gastos de personal y mantenimiento, incrementando así el margen de beneficio.



Figura 1.1: Presupuesto según número de usuarios

Estas estimaciones se han realizado considerando el uso de los recursos del sistema distribuido a pleno rendimiento, por lo que no se ha contemplado la posibilidad de sacrificar, en una fase inicial, cierta precisión en los métodos de optimización o asistencia. No obstante, esta estrategia podría permitir aplazar la necesidad de realizar reinversiones en infraestructura hardware, al reducir los tiempos de respuesta de los procesos y facilitar la atención a un mayor número de usuarios con los mismos recursos disponibles. Como consecuencia, se podrían obtener beneficios en una etapa más temprana del proyecto.

1.4. Estructura de la memoria

La presente memoria se estructura en varios capítulos que describen, de forma ordenada, los diferentes aspectos del proyecto desarrollado. En esta sección se ofrece una breve descripción del contenido de cada uno de ellos, así como de los anexos que complementan la información expuesta.

En primer lugar, el **Capítulo 2** presenta los conceptos previos necesarios para comprender el contexto, base y desarrollo del trabajo. Se abordan tanto los fundamentos relacionados con el fútbol y el modo *Ultimate Team*, como las principales herramientas y tecnologías informáticas utilizadas. Asimismo, se incluyen nociones toma de decisiones y asistentes virtuales.

A continuación, el **Capítulo 3** está dedicado a la investigación de usuarios. Primeramente, se realiza una estimación de los contenidos de interés potencial para los usuarios, apoyada en datos de analítica web. Posteriormente, se expone el estado del arte de las herramientas que utilizan los usuarios actualmente, identificando sus principales limitaciones. Finalmente, se lleva a cabo el modelado de usuarios mediante la definición de arquetipos de usuario.

El **Capítulo 4** presenta el desarrollo del proyecto. Comienza con una descripción general del trabajo realizado, seguida de un análisis detallado del problema relacionado con la creación de plantillas y la explicación de la solución propuesta. A continuación, se presentan la especificación de requisitos y el diseño del sistema desarrollado, abarcando todos los aspectos necesarios para dar respuesta al problema planteado.

En el **Capítulo 5** se presenta un análisis básico de resultados, en el que se expone una prueba de uso del sistema simulando un caso de uso real. De este modo, se muestran los resultados obtenidos y se evalúa el comportamiento del sistema en un escenario representativo.

Finalmente, el **Capítulo 6** está dedicado a las conclusiones y líneas de trabajo futuro. En esta sección se resumen los principales resultados obtenidos, se reflexiona sobre los retos afrontados durante el desarrollo del proyecto y se proponen posibles mejoras o ampliaciones orientadas a optimizar el rendimiento o aumentar las funcionalidades del sistema.

El documento se complementa con: el **Anexo 1**, que detalla las configuraciones necesarias para la instalación y puesta en marcha del sistema, ofreciendo una guía útil para usuarios y desarrolladores; el **Anexo 2**, que contiene el manual de usuario.

Capítulo 2

ANTECEDENTES

En este capítulo se abordan los fundamentos teóricos y contextuales que sirven de base para el desarrollo del presente proyecto. En primer lugar, se presenta una introducción al contexto general en el que se enmarca la propuesta, con el objetivo de situar al lector en la problemática abordada. A continuación, se describen en detalle las principales tecnologías y enfoques metodológicos sobre los que se apoya la solución planteada.

2.1. Introducción al contexto

Este trabajo se sitúa en la intersección entre el fútbol y los videojuegos de simulación deportiva, concretamente en el análisis estratégico de plantillas de Ultimate Team y selección de jugadores para las mismas.

2.1.1. Conocimientos Básicos sobre Fútbol

El fútbol es un deporte colectivo regulado por la International Football Association Board (IFAB)¹, en el que dos equipos, conformados cada uno por once jugadores, compiten con el propósito de anotar un mayor número de goles que su adversario en un tiempo previamente establecido. En esencia, su objetivo principal consiste en desplazar un balón esférico mediante diversas acciones técnicas — tales como pases, regates y remates — hasta introducirlo en la portería contraria, utilizando cualquier

¹<https://www.theifab.com/es/>

parte del cuerpo excepto los brazos y las manos, salvo en el caso del portero dentro de su propia área.

De acuerdo con Hernández Moren [11], el fútbol es un deporte de equipo que se caracteriza por una relación de colaboración-oposición, desarrollado en un espacio compartido y parcialmente indeterminado, con la participación simultánea de los jugadores de ambos conjuntos.

Desde una perspectiva académica, el fútbol se distingue por su alta complejidad táctica y la necesidad de tomar decisiones ágiles en un entorno dinámico y en constante evolución, lo que ha generado un considerable interés en diversas investigaciones enfocadas en el desarrollo de sistemas de apoyo a la toma de decisiones para entrenadores [12, 13].

Conceptos indispensables

Para comprender el desarrollo y las dinámicas de un partido de fútbol, es fundamental conocer los agentes y elementos que intervienen en su transcurso:

- **Campo de fútbol:** área delimitada donde se disputa el encuentro. Este área debe cumplir con las dimensiones establecidas en el reglamento oficial. Se puede observar un ejemplo en la Figura 2.1.



Figura 2.1: Fotografía de un campo de fútbol

- **Portería:** estructura rectangular situada en el centro de cada línea de gol, cuya función es constituir la zona en la que se anotan los goles. Podemos visualizar una fotografía de una portería en la Figura 2.2.
- **Balón:** objeto esférico utilizado durante el juego, cuyo objetivo es ser introducido en la portería adversaria para marcar goles. Podemos visualizar una imagen de este elemento en la Figura 2.3.



Figura 2.2: Fotografía de una portería



Figura 2.3: Fotografía de un balón de fútbol

- **Árbitro:** persona responsable de dirigir el partido y de velar por el cumplimiento estricto de las reglas durante el desarrollo del encuentro. Se puede observar un ejemplo en la Figura 2.4.

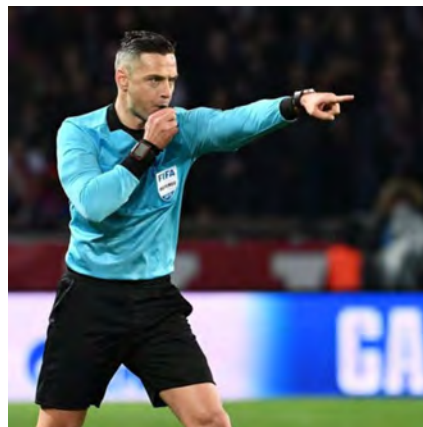


Figura 2.4: Fotografía de un árbitro

- **Equipo o plantilla:** conjunto de once jugadores que colaboran con el propósito común de marcar goles y defender su portería. Podemos visualizar una fotografía de un equipo de fútbol en la Figura 2.5.



Figura 2.5: Fotografía de un equipo de fútbol

Táctica

La **táctica** [14] en el fútbol se refiere a los mecanismos que un equipo ejecuta dentro del contexto de juego para explotar sus propias virtudes y neutralizar las del rival. Cada situación que se presenta durante un partido requiere decisiones tácticas por parte de los jugadores, tanto a nivel individual como colectivo. Por lo tanto, intuitivamente podemos dividir la táctica en dos niveles:

- **Táctica Individual:** define las acciones que un jugador realiza individualmente para resolver situaciones específicas del juego, como desmarques, coberturas, intercepciones...
- **Táctica Colectiva:** abarca las acciones coordinadas de varios jugadores o del equipo completo, como por ejemplo: el marcaje por zonas, el cómo y cuándo aplicar presión a los jugadores rivales...

Formación o Sistema

Un **sistema** [14] es un conjunto estructurado de elementos que se interrelacionan para alcanzar un objetivo. En el fútbol, de manera simplificada, un sistema de juego se refiere a la formación utilizada, esta define las siguientes cuestiones:

- La disposición de los jugadores en el campo.
- La táctica individual de cada jugador: al distribuir los once jugadores por el campo se les asigna a cada uno, roles implícitos en la posición en la que juegan.
- La táctica colectiva.

La Figura 2.6 representa gráficamente la formación 4-4-2.



Figura 2.6: Formación 4-4-2.

Posición

Teniendo en cuenta la definición anterior podemos inferir que la posición es un elemento conceptual de una formación, que se le asigna a un jugador específico para definir su posición relativa con respecto a los demás jugadores en el campo y en consecuencia su táctica individual.

Las posiciones principales existentes en el fútbol son:

- **Portero (GK):** es el único jugador que puede usar las manos dentro del área. Su función principal es evitar que el balón entre en su portería y organizar la defensa.
- **Defensa central (CB):** se ubica en el centro de la defensa. Su tarea es interceptar ataques, despejar balones y marcar a los delanteros rivales.
- **Lateral derecho (RB):** defensa situado en la banda derecha. Cubre los avances por su lado y apoya ofensivamente con centros y desbordes.
- **Lateral izquierdo (LB):** defensa en la banda izquierda, cumple las mismas funciones que el lateral derecho en su respectivo costado.
- **Mediocentro (CM):** se encarga de conectar defensa y ataque. Ayuda en la recuperación y organización del juego.
- **Extremo derecho (RW):** ataca por la banda derecha. Desborda, centra y genera ocasiones desde el costado.

- **Extremo izquierdo (LW):** ataca por la banda izquierda. Su rol es similar al del extremo derecho.
- **Delantero centro (ST):** principal responsable de marcar goles. Se mueve entre los centrales rivales y finaliza jugadas ofensivas.

Todas las abreviaturas están en inglés y se incluyen para ubicar las posiciones mostradas en la Figura 2.6.

2.1.2. FIFA

EA Sports FIFA, comúnmente denominado simplemente como "FIFA", fue una serie de videojuegos de fútbol publicada anualmente por Electronic Arts² bajo el sello de EA Sports, en colaboración con la Federación Internacional de Fútbol³ (FIFA). Esta serie, desarrollada originalmente en Japón, tuvo su origen en 1993 y se distinguió por ser el primer videojuego en obtener una licencia oficial otorgada por la FIFA.

Inicios y trayectoria del videojuego FIFA

A lo largo de su historia, las distintas ediciones de la saga FIFA han incorporado licencias exclusivas que representan equipos, ligas y selecciones nacionales de todo el mundo, tales como la Premier League, la Serie A, la Bundesliga y LaLiga. Además, se destacaron por incluir detalles auténticos como estadios reales, cánticos característicos de los aficionados y características realistas de los jugadores.

El primer título de la serie introdujo una perspectiva isométrica innovadora, que lo diferenció de la tradicional vista superior empleada por otros videojuegos deportivos de la época (ver Figura 2.7).

Desde 1995 hasta 2023, FIFA lanzó una nueva edición de forma anual y se convirtió en la primera franquicia de videojuegos deportivos en superar la cifra de mil millones de copias vendidas, consolidándose como un referente global en simulación deportiva.

Tras la finalización del contrato de licencia con la FIFA, la serie fue renombrada como EA Sports FC⁴, marcando el cierre de la icónica saga bajo su denominación original (ver Figura 2.8).

²<https://www.ea.com/es-es>

³<https://www.fifa.com/es>

⁴<https://www.ea.com/es-es/games/ea-sports-fc>



Figura 2.7: Primer videojuego de la saga FIFA



Figura 2.8: EA Sports FC 25

Modalidad de juego Ultimate Team

Ultimate Team⁵ es una modalidad del videojuego FC 25 que ofrece a los usuarios la posibilidad de coleccionar cartas virtuales de futbolistas profesionales pertenecientes a equipos de todo el mundo. Estas cartas pueden obtenerse, comprándolas utilizando una moneda virtual del juego. A partir de estas cartas, los usuarios pueden confeccionar su propia plantilla personalizada, la cual podrán utilizar para competir en distintas submodalidades dentro del mismo juego.

Las principales características de esta modalidad son:

- **Modo de juego en línea:** posibilita que el equipo creado por el jugador compita en tiempo real contra equipos de otros usuarios.
- **Las cartas del juego representan a jugadores profesionales.**
- **Presenta una submodalidad llamada Desafíos de Creación de Plantillas (SBCs):** esta submodalidad propone a los jugadores completar retos específicos

⁵<https://www.ea.com/es/games/ea-sports-fc/fc-25/features/ultimate-team>

mediante la construcción de plantillas que cumplan con ciertos criterios, tales como valoración media, química, nacionalidades o ligas. Al completar estos desafíos y enviar las plantillas, los jugadores pueden intercambiar cartas de su club por recompensas exclusivas, como sobres, monedas o cartas especiales.

- **Eventos y promociones:** a lo largo del año, se celebran eventos temáticos como el Equipo de la Semana (TOTW), el Equipo del Año (TOTY) y el Equipo de la Temporada (TOTS), que ofrecen cartas especiales y recompensas únicas.
- **Mercado de transferencias:** un sistema donde los usuarios pueden comprar y vender cartas utilizando monedas del juego.

Cartas de jugadores del FC Ultimate Team

Las cartas del juego representan a jugadores profesionales y poseen una serie de características definidas de la siguiente manera:

- **Tipo de carta:** se refiere al evento o promoción mediante el cual la carta fue incluida en el juego. Entre los tipos de carta más relevantes se encuentran los *héroes* y los *iconos*.
- **Atributos:** son las características que describen al jugador y pueden clasificarse en dos grandes categorías:
 - **Atributos numéricos:** representan características cuantificables del jugador. Dentro de esta categoría se distinguen dos tipos:
 - **Escalados:** atributos basados en características del jugador que influyen directamente en su desempeño. Estos valores están normalizados en un intervalo de $[0, 100]$. Ejemplos de estos pueden ser el ritmo, el tiro, la velocidad de sprint...
 - **No escalados:** atributos adicionales que no aparecen en el resumen visual de la carta. Entre ellos destacan:
 - ◇ **Habilidad con la pierna no dominante:** medida en un intervalo de $[0, 5]$.
 - ◇ **Movimientos de habilidad:** evaluados en un intervalo de $[0, 5]$.
 - ◇ **Altura:** valor entero expresado en centímetros.
 - **Atributos nominales:** características del jugador expresadas mediante cadenas de texto, que se agrupan intuitivamente en tres categorías:

- **Roles:** cada posición en el campo cuenta con roles específicos que se asignan a las cartas de los jugadores ubicados en dicha posición, influyendo en su comportamiento y en el cumplimiento de las responsabilidades propias del rol.
- **Estilos de juego:** asignados a las cartas de los jugadores, proporcionan ventajas específicas en determinadas situaciones de juego.
- **Características descriptivas:** propiedades que no afectan directamente el rendimiento, pero que son relevantes para la gestión del equipo. Incluyen aspectos como la nacionalidad, la liga, el equipo y las posiciones en las que el jugador puede actuar.

En la Figura 2.9 se presenta un ejemplo de carta correspondiente a la modalidad Ultimate Team, concretamente del jugador Rodrigo Hernández Cascante.



Figura 2.9: Carta de la modalidad FC Ultimate Team

Atributos Numéricos Escalados

Estos atributos⁶ cuantifican características de los jugadores dentro del intervalo $[0, 100]$ y tienen una influencia directa en su desempeño durante el juego [15]. Se clasifican en dos grandes grupos:

⁶Se incluyen los atributos tanto en español como en inglés, ya que con mayor frecuencia se encuentran denominados en inglés.

- **Atributos numéricos de los porteros:**
 - **Estiradas (Diving):** habilidad para detener disparos dirigidos con precisión.
 - **Posicionamiento (Positioning):** colocación óptima para afrontar remates a portería.
 - **Reflejos (Reflexes):** rapidez de reacción ante tiros cercanos.
 - **Saque (Kicking):** precisión y potencia en los despejes largos.
 - **Velocidad (Speed):** rapidez en salidas desde la portería.

- **Atributos numéricos del resto de jugadores:** estos se organizan en seis categorías principales, definidas como la media aritmética de los atributos que integran cada grupo:
 - **Ritmo (Pace):**
 - **Aceleración (Acceleration):** tiempo requerido para alcanzar la velocidad máxima.
 - **Velocidad de sprint (Sprint Speed):** velocidad máxima al correr sin balón, fundamental para superar rivales.
 - **Regate (Dribbling):**
 - **Agilidad (Agility):** capacidad para cambiar de dirección con rapidez, esencial en espacios reducidos.
 - **Balance (Balance):** estabilidad ante entradas o contactos físicos, clave para mantener el control del balón bajo presión.
 - **Reacciones (Reactions):** rapidez para responder a situaciones inesperadas.
 - **Control de balón (Ball Control):** habilidad para recibir y dominar el balón eficazmente.
 - **Regate (Dribbling):** destreza para superar oponentes en enfrentamientos individuales.
 - **Compostura (Composure):** indica la distancia a la que el jugador que posee el balón comienza a percibir la presión ejercida por un oponente. Influye en la probabilidad de que el jugador cometa un error al realizar acciones como disparos, pases o centros.
 - **Tiro (Shooting):**
 - **Posicionamiento ofensivo (Attacking Positioning):** movimientos inteligentes para recibir pases en zonas de ataque.
 - **Definición (Finishing):** precisión en los disparos dentro del área.

- **Potencia de disparo (Shot Power):** fuerza aplicada en los tiros.
- **Tiros de larga distancia (Long Shots):** precisión en disparos fuera del área.
- **Voleas (Volleys):** capacidad para ejecutar disparos en el aire.
- **Penales (Penalties):** efectividad en lanzamientos desde el punto penal-ti.
- **Pase (Passing):**
 - **Visión de juego (Vision):** habilidad para identificar y ejecutar pases clave.
 - **Centros (Crossing):** precisión en envíos al área desde las bandas.
 - **Pases cortos y largos (Short/Long Passing):** eficacia en pases de diferentes distancias.
- **Defensa (Defending):**
 - **Intercepciones (Interceptions):** capacidad para anticipar y cortar pases rivales.
 - **Conciencia defensiva (Defensive Awareness):** entendimiento táctico para mantener la posición correcta.
 - **Entradas de pie (Standing Tackle):** efectividad en la recuperación del balón mediante entradas normales.
 - **Entradas deslizantes (Sliding Tackle):** precisión y fuerza en las entradas para arrebatar el balón.
- **Físico (Physicality):**
 - **Saltos (Jumping):** capacidad para disputar balones aéreos mediante elevación.
 - **Resistencia (Stamina):** aptitud para mantener un rendimiento óptimo durante todo el encuentro.
 - **Fuerza (Strength):** habilidad para imponerse en duelos físicos.
 - **Agresividad (Aggression):** intensidad y determinación al disputar el balón.

Roles de los jugadores

Los roles son atributos nominales que asociamos a los jugadores para modelar su comportamiento. Tienen las siguientes características:

- Están asociados a posiciones.

- Modelan el comportamiento o movimiento de los jugadores cuando no tienen el balón.
- Determinan cómo se alinean los jugadores con y sin el balón.
- Determinan las responsabilidades principales del jugador.

Por último, cada jugador tendrá una familiaridad con el rol que le permitirá realizar sus tareas con mayor efectividad o eficiencia que otros jugadores. Existen tres niveles de familiaridad:

- **Solo se menciona el rol:** significa que el jugador puede desempeñar ese rol.
- **Se le añade un signo ‘+’:** significa que el jugador es competente en ese rol. Cada jugador en FC 25 tiene al menos un rol de este nivel.
- **Se le añaden dos signos ‘+’:** indica que el jugador es clase mundial en ese rol.

Estilos de Juego

Los estilos de juego en FC Ultimate Team son atributos nominales asignados a las cartas de jugadores que modifican su comportamiento en situaciones específicas durante los partidos. A diferencia de los roles, que definen responsabilidades posicionales, los estilos potencian habilidades particulares o añaden ventajas únicas a los jugadores. Hay dos niveles o tipos para cada estilo de juego:

- **Oro:** el jugador llevará a cabo de una mejor manera la acción que se menciona en el estilo de juego.
- **Plata:** el jugador llevará a cabo la acción que se menciona en el estilo de juego pero no con tanta fiabilidad como el de tipo oro.

Podemos ver los estilos de juego de la carta mostrada anteriormente en la Figura 2.9 en la Figura 2.10.

Química

La química es una variable cuantitativa que puede evaluarse tanto a nivel global de la plantilla en FC Ultimate Team como de forma individual para cada jugador. Aunque

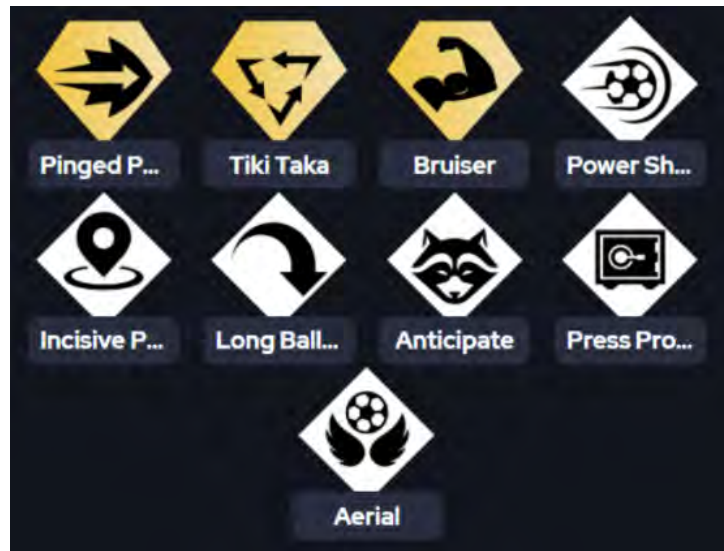


Figura 2.10: Estilos de juego

es posible calcularla para cada jugador, su propósito principal radica en contribuir al valor agregado que representa la química total del equipo. Esta métrica refleja la sinergia existente entre los jugadores y ejerce una influencia directa sobre su desempeño durante los encuentros.

La química de la plantilla se calcula como la suma de los puntos de química aportados por cada jugador, con un máximo total de 33 puntos. Cada jugador puede contribuir con hasta 3 puntos, los cuales se determinan mediante la suma — limitada a un máximo de tres — de los puntos derivados de su nacionalidad, liga o equipo. La asignación de estos puntos se establece de acuerdo con las siguientes reglas:

■ **Respecto a la nacionalidad:**

- Si hay dos jugadores con la misma nacionalidad todos los jugadores de esa nacionalidad adquieren un punto.
- Si hay cinco jugadores con la misma nacionalidad todos los jugadores de esa nacionalidad adquieren dos puntos.
- Si hay ocho jugadores con la misma nacionalidad todos los jugadores de esa nacionalidad adquieren tres puntos.

■ **Respecto a la liga:**

- Si hay tres jugadores de la misma liga todos los jugadores de esa liga adquieren un punto.
- Si hay cinco jugadores de la misma liga todos los jugadores de esa liga adquieren dos puntos.

- Si hay ocho jugadores de la misma liga todos los jugadores de esa liga adquieren tres puntos.
- **Respecto al equipo:**
 - Si hay dos jugadores del mismo equipo todos los jugadores de ese equipo adquieren un punto.
 - Si hay cuatro jugadores del mismo equipo todos los jugadores de ese equipo adquieren dos puntos.
 - Si hay siete jugadores del mismo equipo todos los jugadores de ese equipo adquieren tres puntos.
- **Si el tipo de carta de un jugador es *héroe*:** este se considerará como un jugador adicional en relación con las demás nacionalidades presentes en el equipo, dado que dicho jugador no posee nacionalidad asignada. Además, su contribución equivaldrá a la inclusión de dos jugadores en la plantilla correspondientes a la liga a la que pertenece.
- **Si el tipo de carta de un jugador es *icono*:** este se considerará como un jugador adicional respecto a las ligas presentes en el equipo, dado que dicho jugador no posee liga asignada. Adicionalmente, su valor será equivalente a la inclusión de dos jugadores en la plantilla pertenecientes a la nacionalidad del jugador.

En la Figura 2.11 se muestran las estadísticas que ofrece la interfaz de creación de plantillas del FC 25. En la parte superior se presenta la valoración general de la plantilla, mientras que en la sección inferior se encuentra el indicador de química. Finalmente, en la parte más baja, se incluye una representación detallada de la química, donde se especifica el número de jugadores adicionales necesarios de una misma liga, nacionalidad o club para que los integrantes de ese grupo obtengan un punto más en la química individual.

2.2. Toma de Decisión

Esta sección presenta una introducción al proceso de toma de decisiones, abordando posteriormente la toma de decisiones multiatributo, un modelo específico para la resolución de problemas toma de decisión, y finalizando con las particularidades relacionadas con la automatización de estos métodos para la implementación de sistemas que asistan a los usuarios en dicho proceso.

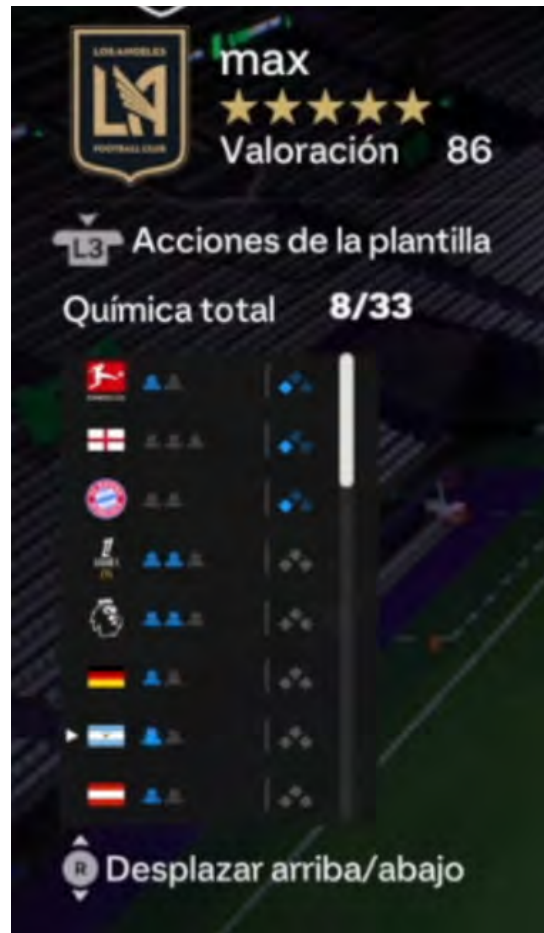


Figura 2.11: Estadísticas de una plantilla de Ultimate Team

2.2.1. Introducción a la Toma de Decisión

La toma de decisiones es un proceso fundamental en la vida de las personas y en el ámbito organizacional, donde las decisiones acertadas son clave para el éxito empresarial. Este proceso implica identificar, evaluar y seleccionar alternativas para resolver problemas o aprovechar oportunidades. Debido a su complejidad, la toma de decisiones requiere un enfoque estructurado que combine el juicio humano con herramientas tecnológicas, como los Sistemas de Soporte a las Decisiones (DSS) [16], diseñados para facilitar el proceso sin reemplazar al tomador de decisiones.

En un escenario clásico de toma de decisiones, se identifican cuatro componentes fundamentales:

- Uno o más objetivos a alcanzar.
- Un conjunto de alternativas o decisiones posibles.
- Un contexto definido por factores, criterios o estados de la naturaleza.

- Una evaluación de las consecuencias asociadas a cada combinación de alternativa y estado.

En general, el proceso de toma de decisiones se organiza en las siguientes etapas [17]:

1. **Identificación y diagnóstico del problema:** consiste en detectar el problema que se desea resolver, recolectando información relevante para comprender todos los elementos involucrados. Un análisis exhaustivo en esta fase garantiza una definición clara del problema.
2. **Generación de alternativas:** en esta etapa se identifican todas las posibles soluciones viables. La creatividad y el conocimiento experto son claves para no omitir opciones relevantes.
3. **Evaluación de alternativas:** cada alternativa es valorada considerando criterios tanto cuantitativos (por ejemplo, costes, tiempos) como cualitativos (como la calidad del entorno laboral o el riesgo político). El análisis se enfoca en identificar ventajas, inconvenientes y el impacto de cada opción sobre los objetivos planteados.
4. **Selección de la mejor alternativa:** una vez evaluadas las opciones, se procede a tomar la decisión más adecuada. Existen distintos enfoques como:
 - Maximizar: escoger la opción más beneficiosa.
 - Satisfacer: seleccionar la primera alternativa que cumpla con los requisitos mínimos.
 - Optimizar: buscar un equilibrio entre múltiples objetivos.
5. **Implementación de la decisión:** la decisión elegida debe ser llevada a la práctica. Es importante que quienes implementen la solución estén alineados con los objetivos, comprendan las razones de la elección y estén comprometidos con su correcta ejecución.
6. **Evaluación de la decisión:** esta última etapa implica analizar los resultados obtenidos tras la implementación. Si los resultados son positivos, puede considerarse la reutilización del enfoque en problemas similares. Si son negativos, se vuelve al diagnóstico inicial con información más enriquecida para tomar una mejor decisión en un nuevo ciclo.

Además de las etapas mencionadas, existen diversos factores que condicionan el proceso decisorio:

- **Información:** disponer de datos adecuados es fundamental para tomar decisiones acertadas.
- **Conocimientos:** la formación y el dominio del área en cuestión por parte de los decisores influyen directamente en la calidad del proceso.
- **Experiencia:** haber enfrentado situaciones similares permite anticipar problemas y detectar patrones útiles.
- **Análisis:** descomponer y estudiar los elementos del problema facilita una mejor comprensión de su naturaleza.
- **Juicio:** la capacidad para ponderar los aspectos relevantes de cada alternativa y seleccionar la más adecuada resulta esencial.

2.2.2. Toma de Decisión Multiatributo

Un problema de toma de decisión multiatributo, también denominado decisión multicriterio o MCDA (Multi-Criteria Decision Analysis), consiste en seleccionar una o varias alternativas entre un conjunto, considerando múltiples criterios o atributos que pueden presentar conflictos entre sí. Este tipo de problemas puede representarse de manera sencilla mediante una matriz.

Una matriz de decisión [18] A es una matriz de dimensiones $m \times n$, donde el elemento a_{ij} representa el desempeño de la alternativa i con respecto al criterio de decisión j , para $i = 1, 2, \dots, m$ y $j = 1, 2, \dots, n$. Se asume asimismo que el tomador de decisiones ha asignado pesos a cada criterio, denotados como w_j , los cuales reflejan la importancia relativa de los mismos. Podemos visualizar un ejemplo de una matriz de decisión en la Ecuación 2.1.

Sea A un conjunto finito de alternativas de decisión y C un conjunto finito de criterios u objetivos según los cuales se evalúa la conveniencia de cada alternativa. El objetivo consiste en identificar la alternativa óptima $a \in A$ que maximice el grado de deseabilidad en relación con todos los criterios relevantes $c \in C$.

Alts.	$C_1(w_1)$	$C_2(w_2)$	$C_3(w_3)$	\cdots	$C_n(w_n)$
A_1	a_{11}	a_{12}	a_{13}	\cdots	a_{1n}
A_2	a_{21}	a_{22}	a_{23}	\cdots	a_{2n}
\vdots	\vdots	\vdots	\vdots	\ddots	\vdots
A_m	a_{m1}	a_{m2}	a_{m3}	\cdots	a_{mn}

(2.1)

2.2.3. TOPSIS (Technique for Order of Preference by Similarity to Ideal Solution)

Los métodos de decisión multiatributo constituyen herramientas fundamentales para abordar MCDA. Si bien existe una amplia variedad de enfoques, en la presente subsección nos centraremos específicamente en el método TOPSIS.

El método TOPSIS [19] fue propuesto por Hwang y Yoon en 1981 como una técnica de ayuda a la toma de decisiones multicriterio. Su fundamento se basa en el concepto de solución de compromiso, según el cual la mejor alternativa es aquella que presenta la menor distancia respecto a la solución ideal positiva y la mayor distancia respecto a la solución ideal negativa. Este funcionamiento lo podemos ver representado en la Figura 2.12.

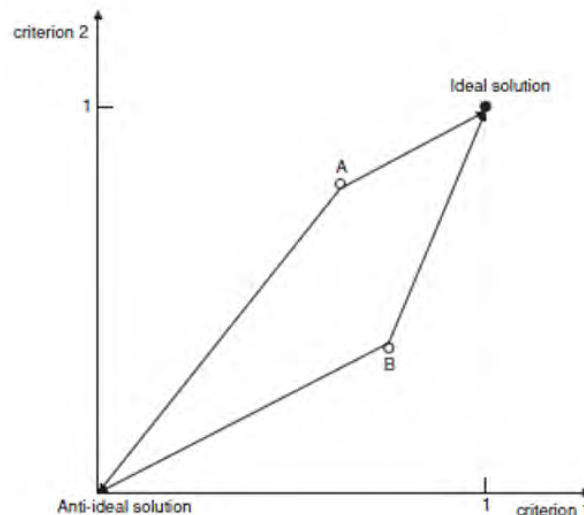


Figura 2.12: Representación gráfica del funcionamiento de TOPSIS

- **Solución Ideal Positiva (SIP):** conjunto formado por los mejores valores alcanzables para cada criterio considerado.
- **Solución Ideal Negativa (SIN):** conjunto formado por los peores valores posibles para cada criterio.

Principales características

- Requiere poca información subjetiva.
- Fácil de aplicar e interpretar.

- Útil en una gran variedad de dominios: selección de proveedores, evaluación de desempeño, planificación estratégica, etc.

Pasos del método TOPSIS

1. **Construcción de la matriz de decisión:** incluye todas las alternativas y los criterios relevantes.
2. **Normalización de la matriz:** las puntuaciones de diferentes criterios se normalizan para comparar diferentes unidades. Existen distintos métodos de normalización:

- **Normalización Distributiva:** es el método más utilizado. Consiste en dividir cada valor por la raíz cuadrada de la suma de los cuadrados de todos los valores del mismo criterio. Este cálculo se representa en la Ecuación 2.2.

$$r_{ia} = \frac{x_{ia}}{\sqrt{\sum_{a=1}^n x_{ia}^2}} \quad \text{para } a = 1, \dots, n \quad \text{y } i = 1, \dots, m. \quad (2.2)$$

- **Normalización Ideal:**

- Criterios de Beneficio (Maximización): se divide cada valor por el valor máximo del criterio correspondiente, como se muestra en las Ecuaciones 2.3 y 2.4.

$$r_{ai} = \frac{x_{ai}}{u_a^+} \quad \text{para } a = 1, \dots, n \quad \text{y } i = 1, \dots, m, \quad (2.3)$$

$$u_a^+ = \text{máx}(x_{ai}) \quad \text{para todos } a = 1, \dots, n. \quad (2.4)$$

- Criterios de Coste (Minimización): se divide cada valor por el valor mínimo del criterio correspondiente, como se muestra en las Ecuaciones 2.5 y 2.6:

$$r_{ai} = \frac{u_a^-}{x_{ai}} \quad \text{para } a = 1, \dots, n \quad \text{y } i = 1, \dots, m, \quad (2.5)$$

$$u_a^- = \text{mín}(x_{ai}) \quad \text{para todos } a = 1, \dots, n. \quad (2.6)$$

- **Normalización Min-Max:** transforma los valores originales de cada atributo al rango $[0, 1]$, utilizando la Ecuación 2.7.

$$r_{ai} = \frac{x_{ai} - \text{mín}(x_a)}{\text{máx}(x_a) - \text{mín}(x_a)} \quad \text{para } a = 1, \dots, n. \quad (2.7)$$

Aunque se trata de una técnica ampliamente utilizada, su aplicación en el contexto de TOPSIS presenta determinadas limitaciones. En primer lugar, tiende a distorsionar la diversidad original de los datos al comprimir su variabilidad, lo que puede afectar negativamente a su capacidad discriminativa. Además, homogeneiza las contribuciones relativas de los atributos, disminuyendo el peso de aquellos que presentan una mayor variabilidad inicial, lo que puede derivar en inconsistencias en el ranking final.

Por estos motivos, se recomienda emplear la normalización distributiva como alternativa, ya que permite preservar de forma más adecuada la diversidad informativa de los datos y contribuye a obtener resultados más robustos en el proceso de toma de decisiones [20].

3. **Ponderación de los criterios:** se aplican los pesos a los valores normalizados. Esta operación consiste en multiplicar cada valor normalizado por el peso correspondiente al criterio, tal como se muestra en la Ecuación 2.8, definiendo el vector de pesos con la siguiente Ecuación 2.9.

$$v_{ai} = w_i \cdot r_{ai} \quad (2.8)$$

$$W = \begin{bmatrix} w_1 & w_2 & \cdots & w_n \end{bmatrix} \quad (2.9)$$

4. **Determinación de la SIP y la SIN:** se puede afrontar de varias maneras:

- Se identifican los mejores y peores valores por criterio, definiendo la SIP y la SIN respectivamente.
- Se asumen unos puntos ideales y anti-ideales absolutos. De la misma forma en la que se expresan en las siguientes ecuaciones, de donde la Ecuación 2.10 define la SIP y la Ecuación 2.11 representa la SIN.

$$A^+ = (1, \dots, 1) \quad (2.10)$$

$$A^- = (0, \dots, 0) \quad (2.11)$$

- La SIP y SIN son definidas por el tomador de decisiones y deben estar entre los puntos absolutos. Es más difícil de gestionar.

5. **Cálculo de distancias euclidianas:** se mide la distancia de cada alternativa a la Solución Ideal Positiva (SIP) y a la Solución Ideal Negativa (SIN). La distancia a la SIP se calcula mediante la Ecuación 2.12, y la distancia a la SIN usando la Ecuación 2.13.

$$d_a^+ = \sqrt{\sum_i (v_i^+ - v_{ai})^2}, \quad a = 1, \dots, m \quad (2.12)$$

$$d_a^- = \sqrt{\sum_i (v_i^- - v_{ai})^2}, \quad a = 1, \dots, m \quad (2.13)$$

6. **Cálculo del coeficiente de cercanía relativa:** se calcula un índice de preferencia para cada alternativa. Este coeficiente se define mediante la Ecuación (2.14):

$$C_a = \frac{d_a^-}{d_a^+ + d_a^-} \quad (2.14)$$

7. **Ordenación de las alternativas:** se genera un ranking en función de la cercanía relativa.

2.2.4. DSS

Los DSS [21] son herramientas computarizadas diseñadas para asistir a los tomadores de decisiones en situaciones semiestructuradas o no estructuradas. Su objetivo principal es mejorar la calidad de las decisiones al proporcionar acceso a datos, modelos y herramientas analíticas, sin reemplazar el juicio humano. A continuación, se detallan sus características, componentes, clasificación y evolución histórica.

Definición y Características

Un DSS se define como un sistema interactivo y flexible que combina datos, modelos y conocimiento para apoyar el proceso de toma de decisiones. Sus características principales incluyen:

- **Interactividad:** permite a los usuarios controlar la secuencia de operaciones y ajustar parámetros según sea necesario.
- **Orientación a problemas semiestructurados:** ideal para situaciones donde no existe un procedimiento predefinido para la solución.
- **Uso de datos y modelos:** integra bases de datos y modelos cuantitativos para analizar alternativas.

- **Soporte a múltiples niveles organizacionales:** desde operaciones cotidianas hasta planeación estratégica.
- **Interfaz amigable:** facilita la comunicación entre el sistema y el usuario.

Componentes de un DSS

Los DSS están compuestos por varios subsistemas que trabajan de manera integrada, basados en el marco teórico de Bonczek, Holsapple y Whinston [18] (ver Figura 2.13):

- **Subsistema de datos:** gestiona la recuperación, almacenamiento y organización de datos relevantes. Incluye bases de datos y herramientas de consulta.
- **Subsistema de modelos:** contiene modelos analíticos (simulación, optimización, etc.) para evaluar alternativas.
- **Máquina de conocimiento:** proporciona inteligencia al sistema, combinando datos y modelos para generar soluciones.
- **Interfaz de usuario:** permite la interacción entre el usuario y el sistema, facilitando la visualización de resultados.
- **Usuario:** esencial para el éxito del DSS, ya que su experiencia y juicio complementan las capacidades del sistema.

Clasificación de los DSS

Los DSS pueden clasificarse según su componente dominante o su propósito específico. Una clasificación ampliamente aceptada es la propuesta por Power [22], que distingue cinco tipos principales:

- **Orientados a datos:** se centran en el manejo, almacenamiento y análisis de grandes volúmenes de información estructurada. Ejemplos destacados de esta categoría son los *Data Warehouses* [23] y los sistemas OLAP (*Online Analytical Processing*) [24].

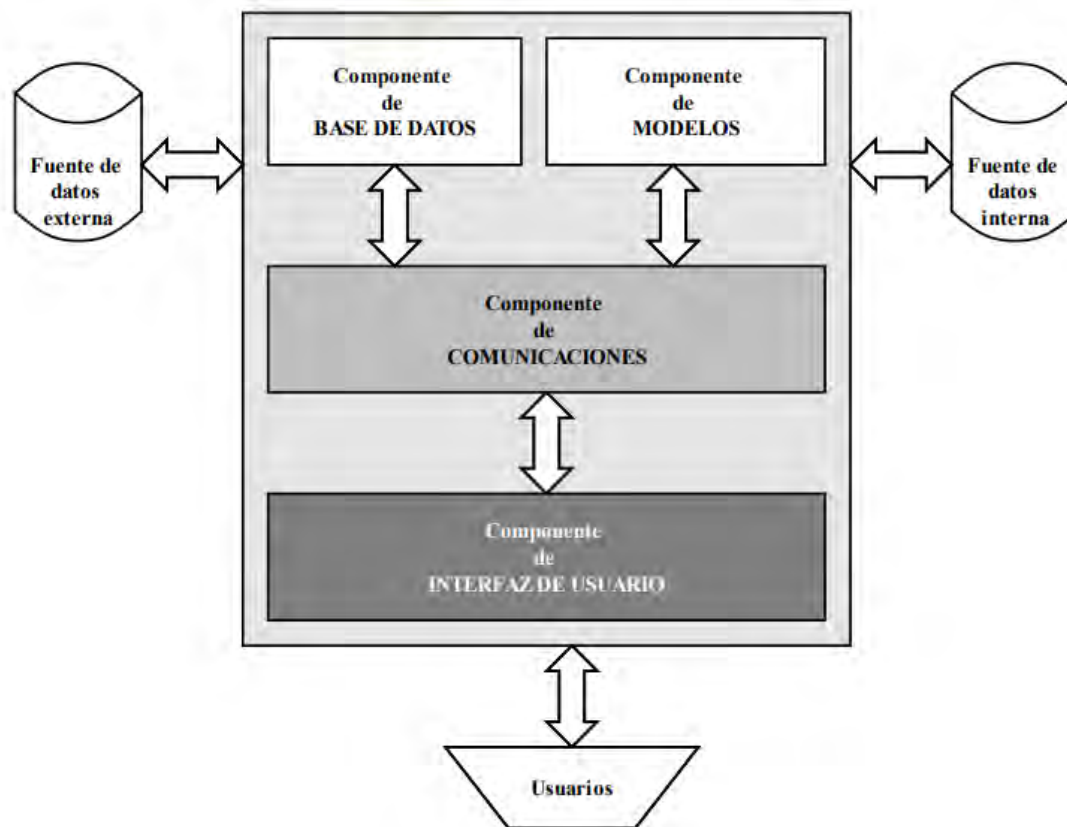


Figura 2.13: Arquitectura genérica de un DSS

- **Orientados a modelos:** utilizan modelos cuantitativos, matemáticos o estadísticos para facilitar el análisis de alternativas y la simulación de escenarios. Algunos ejemplos relevantes son los sistemas de optimización, como el *solver* de Frontline Systems (1987) [25], o herramientas de simulación como DYNAMO [26], desarrollado por Forrester en la década de 1960.
- **Orientados al conocimiento:** emplean técnicas para representar el conocimiento experto y proporcionar recomendaciones o diagnósticos. Entre los ejemplos más representativos se encuentran los sistemas expertos, como EXSYS [27] (1983).
- **Orientados a documentos:** están diseñados para gestionar y recuperar información no estructurada, como informes o textos. Esta categoría incluye motores de búsqueda y sistemas pioneros como PLATO Notes [28], considerado uno de los precursores de las plataformas colaborativas en la web.
- **Orientados a comunicaciones:** facilitan el trabajo colaborativo y la toma de decisiones en grupo mediante herramientas de comunicación. Ejemplos de esta categoría son los Sistemas de Soporte a Decisiones en Grupo (GDSS), como GroupSystems [29] y SAMM [30].

Ventajas y Desventajas

■ Ventajas:

- Mejoran la calidad y rapidez de las decisiones.
- Facilitan el acceso a información relevante.
- Permiten analizar múltiples escenarios.
- Fomentan la colaboración entre equipos.

■ Desventajas:

- Limitaciones en la representación de problemas complejos.
- Dependencia del diseño y conocimiento del usuario.
- Dificultad para incorporar creatividad e intuición humana.
- Algunos presentan una curva de aprendizaje pronunciada.

2.3. Optimización por Colonias de Hormigas (ACO)

Las metaheurísticas constituyen estrategias generales de optimización diseñadas para abordar problemas complejos, especialmente aquellos clasificados como NP-difíciles, en los que los métodos exactos resultan ineficaces debido a su elevado coste computacional.

La metaheurística ACO [31] se inspira en el comportamiento colectivo de las hormigas en la naturaleza, las cuales utilizan feromonas para marcar y reforzar las rutas más eficientes hacia las fuentes de alimento. En la Figura 2.14 se muestra una representación gráfica de este comportamiento, ilustrando el proceso anteriormente descrito.

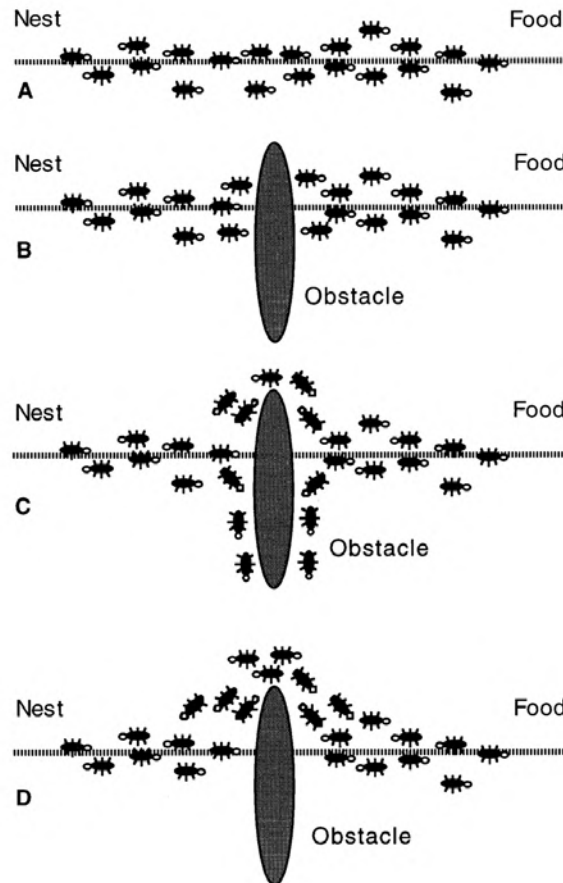


Figura 2.14: Representación gráfica del comportamiento de las hormigas [1]

2.3.1. Sistema de Hormigas (AS)

El algoritmo AS fue introducido por primera vez por Dorigo et al. [32] y ha demostrado ser eficaz para resolver problemas de optimización combinatoria, como el TSP, la planificación de rutas...

El algoritmo AS sigue los siguientes pasos (ver Figura 2.15):

1. **Inicialización:** se definen los parámetros iniciales y se asignan valores iniciales a las feromonas sobre los componentes del espacio de solución.
 - **Número de hormigas en la colonia:** un mayor número de hormigas permite una mejor exploración, pero incrementa el coste computacional.
 - **Valor inicial del rastro de feromona.**
 - **Tasa de evaporación de feromonas:** controla la pérdida de información histórica.

- α : peso que controla la influencia del rastro de feromona en la construcción de soluciones.
 - β : peso que determina la importancia de la información heurística a la hora de construir soluciones.
 - **Información heurística:** derivada de los datos del problema (por ejemplo, $1/d_{ij}$ en el Travelling Salesman Problem o TSP), puede calcularse antes de iniciar el algoritmo para acelerar el proceso de construcción de soluciones.
2. **Construcción de soluciones:** cada hormiga construye una solución de forma incremental, seleccionando en cada paso el siguiente componente de solución con base en una probabilidad que combina la intensidad de feromonas y la información heurística local. La probabilidad se calcula con la Ecuación 2.15.

$$p_{ij}^{(k)} = \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in \mathcal{N}_i^{(k)}} [\tau_{il}]^\alpha [\eta_{il}]^\beta} \quad (2.15)$$

Donde:

- $p_{ij}^{(k)}$: Probabilidad de que la hormiga k se desplace del nodo i al nodo j .
- τ_{ij} : Cantidad de feromona depositada en el arco (i, j) .
- η_{ij} : Información heurística del arco (i, j) , por ejemplo $\eta_{ij} = 1/d_{ij}$ en el TSP.
- α : Parámetro que pondera la influencia de la feromona en la elección.
- β : Parámetro que pondera la importancia de la heurística.
- $\mathcal{N}_i^{(k)}$: Conjunto de nodos no visitados por la hormiga k al estar en i .

3. **Búsqueda local:** se puede aplicar una heurística de mejora sobre las soluciones generadas por las hormigas con el objetivo de refinar su calidad. Esta etapa es opcional, pero suele mejorar significativamente la eficiencia del algoritmo, especialmente en problemas combinatorios [31].
4. **Actualización global de feromonas:** se actualizan los niveles de feromonas en función de la calidad de las soluciones encontradas por las hormigas, reforzando aquellas rutas que forman parte de las mejores soluciones y favoreciendo su exploración en futuras iteraciones. Además, se aplica un mecanismo de evaporación para evitar la convergencia prematura del algoritmo hacia óptimos locales. La actualización global se formaliza mediante la Ecuación 2.16.

$$\tau_{ij} = (1 - \rho) \cdot \tau_{ij} + \Delta\tau_{ij} \quad (2.16)$$

Donde:

τ_{ij} : Cantidad de feromona sobre el arco (i, j) .

ρ : Tasa de evaporación de feromonas, con $0 < \rho \leq 1$.

$\Delta\tau_{ij}$: Cantidad de feromona depositada sobre (i, j) por las mejores hormigas.

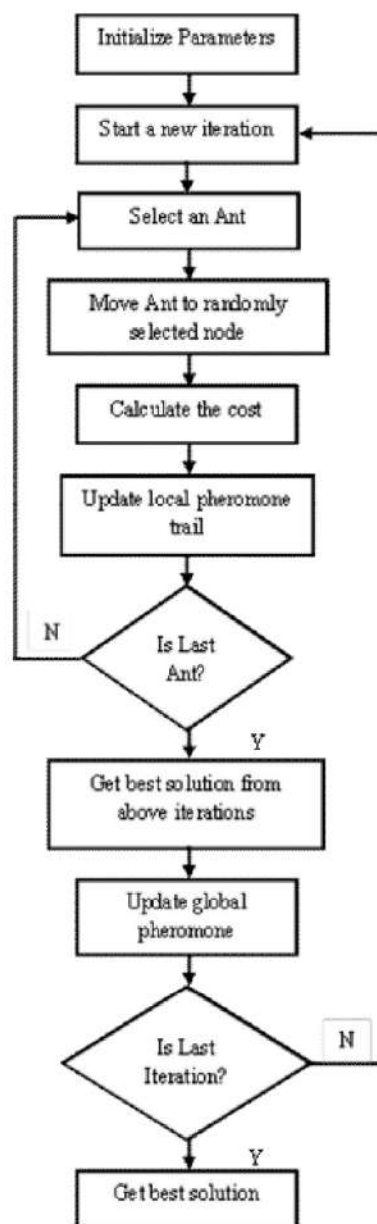


Figura 2.15: Algoritmo ACO [2]

2.3.2. Sistema de Colonias de Hormigas (ACS)

El ACS [1] es una variante del algoritmo ACO introducida por Marco Dorigo y Luca María Gambardella en 1997, diseñada para mejorar la eficiencia del algoritmo AS original. ACS incorpora mecanismos que equilibran mejor la exploración del espacio de búsqueda y la explotación del conocimiento adquirido, logrando una convergencia más rápida hacia soluciones de alta calidad.

A continuación, se describen los principales elementos y características que diferencian al ACS:

- Regla de transición pseudoaleatoria:** en lugar de seleccionar el siguiente nodo únicamente mediante una función de probabilidad como en ACO, ACS introduce una regla que combina decisión determinista y estocástica. Con una probabilidad q_0 , la hormiga selecciona el siguiente nodo de manera determinista, eligiendo aquel que maximiza la combinación entre feromona e información heurística. En caso contrario ($1 - q_0$), selecciona el nodo siguiente mediante la fórmula probabilística de la Ecuación 2.15. La expresión matemática de esta regla viene definida en la Ecuación 2.17.

$$j = \begin{cases} \arg \max_{l \in \mathcal{N}_i^{(k)}} (\tau_{il} \cdot \eta_{il}^\beta), & \text{si } q \leq q_0 \\ \text{Seleccionado según } p_{ij}^{(k)}, & \text{si } q > q_0 \end{cases} \quad (2.17)$$

Donde:

- j : Próximo nodo que selecciona la hormiga k desde el nodo i .
- q : Número aleatorio generado uniformemente en el intervalo $[0, 1]$.
- q_0 : Parámetro del algoritmo que controla el equilibrio entre explotación y exploración.
- τ_{il} : Cantidad de feromona presente en el arco (i, l) .
- η_{il} : Información heurística asociada al arco (i, l) .
- β : Parámetro que determina la influencia relativa de la heurística.
- $p_{ij}^{(k)}$: Prob. de seleccionar el nodo j desde i según la regla de transición probabilística.
- $\mathcal{N}_i^{(k)}$: Conjunto de nodos aún no visitados por la hormiga k al encontrarse en i .

- Actualización local de feromonas:** cada vez que una hormiga recorre una arista (i, j) , actualiza localmente su valor de feromona con el objetivo de hacerla

temporalmente menos atractiva para otras hormigas, fomentando así la diversificación. Podemos observar la expresión matemática para realizar la actualización local de feromonas en la Ecuación 2.18.

$$\tau_{ij} = (1 - \rho) \cdot \tau_{ij} + \rho \cdot \tau_0 \quad (2.18)$$

Donde:

τ_{ij} : Cantidad de feromona actual en el arco (i, j) .

ρ : Tasa de evaporación de feromonas, con $0 < \rho < 1$.

τ_0 : Valor inicial de feromona depositada en cada arco.

- **Actualización global de feromonas:** Al final de cada iteración, únicamente la mejor hormiga global (es decir, la que ha encontrado la mejor solución hasta el momento) actualiza las feromonas a lo largo de su recorrido. La actualización global la podemos ver en la Ecuación 2.19.

$$\tau_{ij} = (1 - \alpha) \cdot \tau_{ij} + \alpha \cdot \Delta\tau_{ij} \quad (2.19)$$

Donde:

τ_{ij} : Cantidad de feromona en el arco (i, j) .

α : Tasa de refuerzo aplicada al camino recorrido por la mejor hormiga.

$\Delta\tau_{ij}$: Cant. de feromona añadida según la calidad de la mejor sol. global, definida como:

$$\Delta\tau_{ij} = \begin{cases} \frac{1}{L_{gb}}, & \text{si } (i, j) \text{ pertenece al recorrido de la mejor hormiga global} \\ 0, & \text{en otro caso} \end{cases}$$

L_{gb} : Longitud total de la mejor solución encontrada hasta el momento.

2.4. Asistentes Virtuales

En esta sección se presentará una breve introducción a los asistentes virtuales, para concluir con un análisis de las tecnologías más avanzadas que se están utilizando en su implementación.

2.4.1. Introducción a los Asistentes Virtuales

Un asistente virtual [33] es un agente de software diseñado para asistir a los usuarios de sistemas computacionales mediante la automatización y ejecución de tareas, minimizando la necesidad de interacción directa entre el ser humano y la máquina. La comunicación entre el asistente virtual y el usuario debe desarrollarse de forma natural y fluida.

A lo largo del tiempo, los asistentes virtuales han experimentado una notable evolución, adaptándose a las crecientes demandas de los usuarios y aprovechando los continuos avances en el ámbito de la inteligencia artificial. En sus primeras versiones, estos sistemas se limitaban a la ejecución de tareas simples mediante comandos predefinidos. No obstante, con el desarrollo de nuevas tecnologías, han incorporado funcionalidades más sofisticadas, como el procesamiento del lenguaje natural (PLN), el aprendizaje automático y la interacción multimodal, lo que ha permitido establecer formas de comunicación cada vez más naturales e intuitivas.

En la actualidad, los asistentes virtuales no solo se encargan de tareas básicas [34] como establecer recordatorios o reproducir contenido multimedia, sino que también son capaces de gestionar agendas, controlar dispositivos inteligentes en el hogar [35], proporcionar información en tiempo real [36] y ofrecer soporte en múltiples ámbitos, desde la educación [37, 36] hasta la atención al cliente [38]. Esta evolución ha sido posible gracias al desarrollo de modelos de lenguaje de gran escala, como GPT-4o⁷ de OpenAI (ver Figura 2.16), que permiten una comprensión y generación de texto más precisa, coherente y contextualizada.



Figura 2.16: Logo de ChatGPT

⁷<https://openai.com/es-ES/index/hello-gpt-4o/>

2.4.2. Historia de los Asistentes Virtuales

Orígenes: ELIZA

El origen del concepto de asistente virtual se remonta a la década de 1960, con el desarrollo de ELIZA [39] por Joseph Weizenbaum en el Instituto Tecnológico de Massachusetts (MIT). Este sistema pionero simulaba una conversación con un psicoterapeuta mediante el uso de patrones de texto predefinidos, constituyendo un hito fundamental en la evolución de la interacción entre el ser humano y las computadoras (ver Figura 2.17).

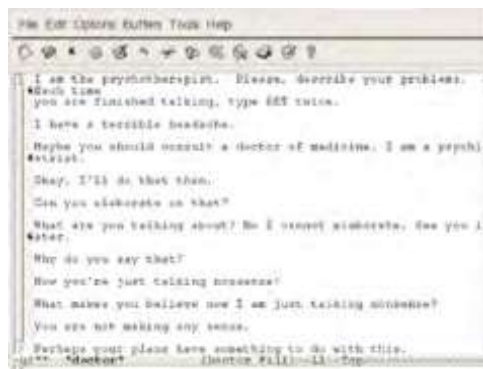


Figura 2.17: Eliza en Emacs

Integración a Entornos de Trabajo: Clippy

En 1997, Microsoft introdujo a Clippy [40], un asistente virtual integrado en Microsoft Office que ofrecía ayuda contextual a los usuarios. Aunque limitado en funcionalidad, representó un intento temprano de incorporar asistentes virtuales en aplicaciones de uso cotidiano (ver Figura 2.18).

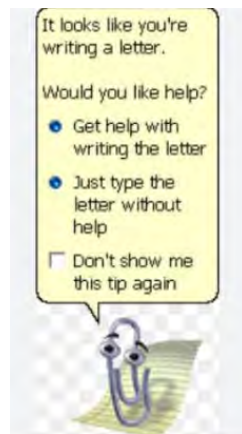


Figura 2.18: Asistente Virtual de Microsoft Clippy

Popularización con Dispositivos Móviles: Siri, Google Now, Alexa

El lanzamiento de Siri⁸ (ver Figura 2.19) por parte de Apple en 2011 representó un punto de inflexión en el desarrollo de asistentes virtuales, al ser el primero en integrarse de forma nativa en un teléfono inteligente. Esta innovación permitió a los usuarios interactuar con el dispositivo mediante comandos de voz para ejecutar diversas tareas, como el envío de mensajes, la configuración de recordatorios o la consulta de información, consolidando así el uso de la interacción por voz en el entorno móvil.

Posteriormente, Google introdujo Google Now en 2012 y, más adelante, Google Assistant⁹ en 2016, incorporando respuestas contextuales y una integración más profunda con el ecosistema de servicios de la compañía. Por su parte, Amazon lanzó Alexa¹⁰ en 2014, junto con el dispositivo Echo, lo que supuso una transformación en el ámbito de la automatización del hogar al posibilitar el control de dispositivos domésticos inteligentes a través de comandos de voz.

⁸<https://www.apple.com/es/siri/>

⁹<https://assistant.google.com/>

¹⁰<https://alexa.amazon.com/>



Figura 2.19: Logo de Siri

Actualidad: Integración de IA Generativa y RAG

En la actualidad, los asistentes virtuales han evolucionado mediante la incorporación de tecnologías avanzadas, como la IA generativa y la RAG, concepto que podemos explorar en la Sección 2.4.3. Esta última combina modelos de lenguaje generativo con sistemas de recuperación de información, lo que permite a los asistentes acceder a fuentes de datos externas para obtener información actualizada y específica. Como resultado, se mejora significativamente la precisión, relevancia y contextualización de las respuestas proporcionadas al usuario.

2.4.3. RAG

La RI [41] es el proceso orientado a localizar material de naturaleza no estructurada que responda a una necesidad de información dentro de grandes colecciones (ver Figura 2.20).

La tarea fundamental de la RI ha sido, tradicionalmente, identificar un conjunto ordenado de documentos relevantes para una consulta formulada por el usuario, a partir de un corpus de textos en lenguaje natural. En la actualidad, el objetivo se ha orientado hacia la obtención de la mejor respuesta posible que satisfaga de manera precisa la necesidad de información expresada en dicha consulta.

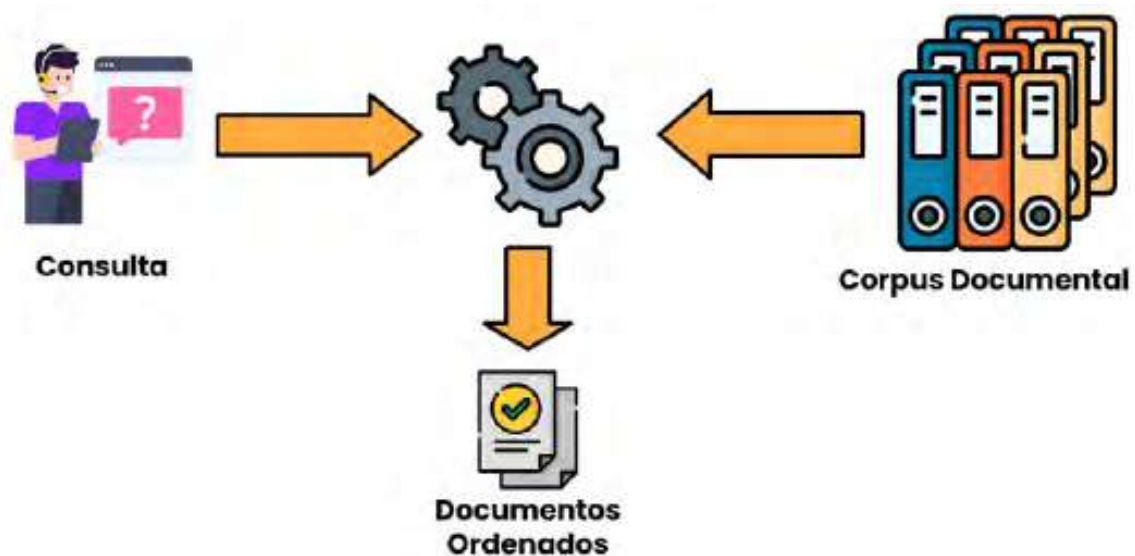


Figura 2.20: Esquema básico del contexto de un SRI

Funcionamiento de un sistema RAG

La arquitectura de la RAG integra modelos de lenguaje generativos con sistemas de RI, permitiendo a los asistentes virtuales no solo generar respuestas basadas en patrones aprendidos, sino también incorporar información actualizada y relevante proveniente de bases de datos externas. Esta combinación mejora significativamente la precisión, coherencia y contextualización de las respuestas generadas.

El funcionamiento de un sistema RAG se articula en las siguientes etapas [42] (ver Figura 2.21):

1. **Recopilación y segmentación de datos:** se recopilan datos relevantes a partir de diversas fuentes (documentos, bases de datos, artículos, entre otros) y se segmentan en unidades más pequeñas, denominadas *chunks*, con el fin de facilitar una recuperación más eficiente.
2. **Generación de incrustaciones (Embeddings):** cada fragmento de información se transforma en una representación numérica (vector) mediante modelos de aprendizaje automático. Estos vectores capturan el significado semántico del contenido, lo que permite realizar comparaciones eficientes.
3. **Almacenamiento en bases de datos vectoriales:** los vectores generados se almacenan en bases de datos especializadas (como FAISS¹¹, Pinecone¹² o Chro-

¹¹<https://github.com/facebookresearch/faiss>

¹²<https://www.pinecone.io/>

ma¹³), optimizadas para búsquedas rápidas y eficaces en espacios vectoriales.

4. **Recuperación de información relevante:** cuando un usuario formula una consulta, esta se convierte en un vector que se compara con los vectores almacenados, recuperando así los fragmentos de información más relevantes en función de su similitud semántica.
5. **Generación de respuestas contextualizadas:** los fragmentos recuperados se combinan con la consulta original para conformar un contexto enriquecido, que es utilizado por el modelo de lenguaje generativo para elaborar una respuesta precisa, coherente y fundamentada.
6. **Optimización y mejora continua:** el sistema es supervisado y ajustado de forma continua con el objetivo de optimizar la calidad de las respuestas y la relevancia de la información recuperada, incorporando mecanismos de retroalimentación proporcionados por los usuarios.

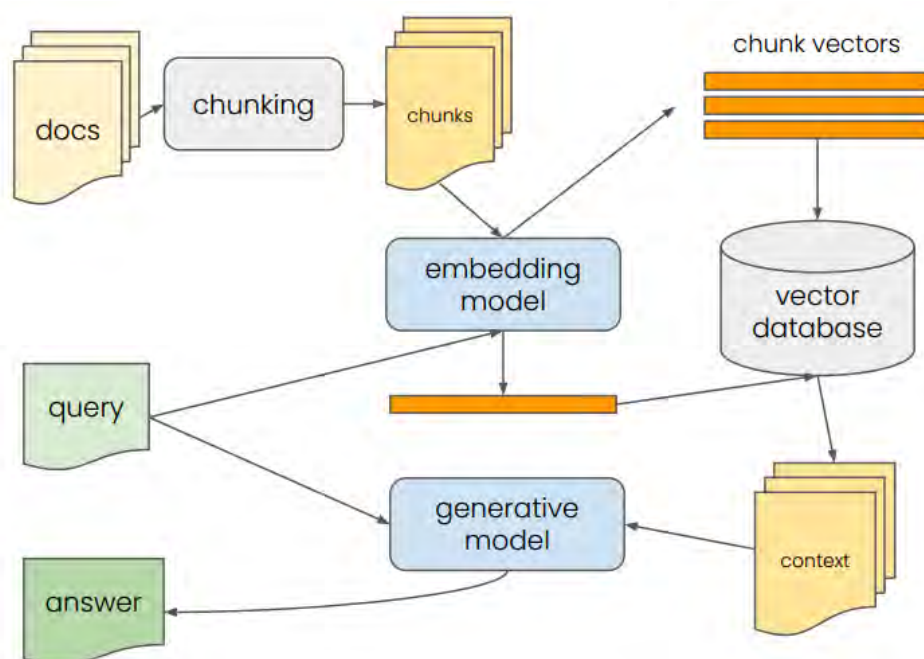


Figura 2.21: Funcionamiento de un RAG

2.4.4. Agentic RAG

El término **Agentic RAG** (Agentic Retrieval-Augmented Generation) representa un paradigma de vanguardia que combina la generación de texto mediante grandes mo-

¹³<https://www.trychroma.com/>

delos de lenguaje (LLMs) con una recuperación de información dinámica, gestionada por agentes inteligentes. A diferencia de los sistemas RAG tradicionales, que se basan en flujos de trabajo estáticos, los sistemas Agentic RAG introducen agentes autónomos capaces de tomar decisiones iterativas, planificar, utilizar herramientas y colaborar entre sí para enfrentarse a consultas complejas en entornos cambiantes.

El concepto se formalizó por primera vez en el artículo “AU-RAG: Agent-based Universal Retrieval Augmented Generation” [43], presentado en la conferencia SIGIR-AP 2024 por Jisoo Jang y Wen-Syan Li. En este trabajo se propone un marco universal que utiliza un agente que decide dinámicamente entre fuentes paramétricas y recuperadas, enriquecidas con metadatos, para optimizar la búsqueda de información y su integración en la generación de respuesta.

Más recientemente, el artículo titulado *Agentic Retrieval-Augmented Generation: A Survey on Agentic RAG* [44], publicado el 4 de febrero de 2025 por Aditi Singh, Abul Ehtesham, Saket Kumar y Tala Talaei Khoei, sistematiza este paradigma. En él se destaca que los agentes autónomos aplican patrones como reflexión, planificación, uso de herramientas y colaboración multiagente para:

- Ajustar de forma dinámica las estrategias de recuperación.
- Refinar iterativamente la comprensión del contexto.
- Adaptar el flujo de trabajo según la complejidad de las tareas.

Componentes de los sistemas Agentic RAG

De acuerdo con Singh et al. [44], los sistemas Agentic RAG se sustentan en una arquitectura compuesta por tres elementos esenciales: agentes autónomos, patrones de comportamiento y flujos de trabajo estructurados. Esta configuración proporciona al sistema la capacidad de tomar decisiones de forma autónoma, perfeccionar iterativamente sus respuestas y adaptarse dinámicamente a contextos complejos y cambiantes. A continuación, se describen sus componentes principales:

- **Agentes autónomos:** constituyen la unidad funcional básica de estos sistemas. Cada agente está diseñado para operar de forma independiente y dispone de los siguientes elementos:
 - **LLM:** que actúa como núcleo del razonamiento, la comprensión y la generación textual.

- **Memoria a corto y largo plazo:** que permite preservar el contexto durante interacciones sostenidas, así como almacenar conocimiento acumulado relevante para futuras tareas.
 - **Capacidades de planificación:** sustentadas en mecanismos de reflexión y autocrítica, que orientan el razonamiento iterativo, la descomposición de tareas complejas y la mejora progresiva de las respuestas generadas.
 - **Acceso a herramientas externas:** tales como motores de búsqueda, bases de datos, interfaces de programación de aplicaciones (APIs) o sistemas computacionales especializados, lo que amplía su capacidad operativa más allá de la generación textual basada exclusivamente en el modelo.
- **Patrones agentivos:** representan los principios metodológicos que guían el comportamiento de los agentes dentro del sistema. Estos patrones se estructuran en cuatro categorías fundamentales:
 1. **Reflexión:** permite a los agentes realizar procesos de autoevaluación y mejora iterativa, identificando deficiencias, errores o incoherencias en sus respuestas.
 2. **Planificación:** facilita la descomposición de tareas complejas en subtareas más simples, permitiendo establecer una secuencia lógica y eficiente para su resolución.
 3. **Uso de herramientas:** habilita el acceso a fuentes de información externas o recursos computacionales adicionales, ampliando la capacidad del agente para obtener datos actualizados o realizar cálculos específicos.
 4. **Colaboración multiagente:** posibilita la cooperación entre varios agentes especializados, que interactúan de forma orquestada para abordar consultas complejas o multidominio.
 - **Flujos de trabajo agentivos:** constituyen la estructura operativa que organiza las interacciones entre los agentes y las tareas que deben ejecutar. Estos flujos permiten gestionar la complejidad de los procesos y optimizar el rendimiento del sistema. Entre los principales esquemas se encuentran:
 - **Encadenamiento de prompts:** estructura las tareas en una secuencia lógica de pasos interdependientes, facilitando la resolución progresiva de problemas complejos.
 - **Enrutamiento:** asigna las consultas entrantes a los procesos o agentes más adecuados en función de su naturaleza, optimizando así el tratamiento de cada solicitud.

- **Paralelización:** permite la ejecución simultánea de tareas independientes, reduciendo los tiempos de respuesta y aumentando la eficiencia computacional.
- **Modelo orquestador-trabajador:** establece una jerarquía funcional en la que un agente orquestador se encarga de dividir el trabajo en subtareas y delegarlas en agentes trabajadores especializados.
- **Evaluador-optimizador:** implementa un mecanismo iterativo mediante el cual un agente evaluador revisa los resultados generados y, si es necesario, activa procesos de optimización para mejorar la calidad final de las respuestas.

Comportamiento general de los sistemas Agentic RAG

Todos los sistemas Agentic RAG, independientemente de su arquitectura específica, comparten un flujo general estructurado en fases sucesivas. Este comportamiento se fundamenta en la capacidad de los agentes para tomar decisiones dinámicas, ajustar estrategias de recuperación y refinar iterativamente los resultados obtenidos. El esquema general se desarrolla a través de las siguientes etapas (ver Figura 2.22):

1. **Recepción y análisis de la consulta:** el proceso se inicia con la recepción de la consulta por parte de un agente coordinador, que puede ser un agente único, un agente jerárquico o un orquestador en sistemas multiagente.
2. **Evaluación contextual y estratégica:** el agente responsable analiza la naturaleza y complejidad de la consulta, determinando si requiere recuperación simple, búsqueda multi-hop o integración de distintas fuentes (estructuradas o no estructuradas).
3. **Recuperación de información:** el sistema ejecuta la búsqueda a través de uno o varios agentes especializados. Estos agentes pueden acceder a bases de datos, sistemas de recuperación semántica, motores de búsqueda web o grafos de conocimiento, según las necesidades específicas del caso.
4. **Validación y refinamiento iterativo:** la información recuperada es evaluada por agentes correctores o módulos críticos que determinan su relevancia y calidad. Si es necesario, se activan mecanismos de refinamiento, que pueden incluir reescritura de consultas, búsqueda en fuentes alternativas o incorporación de nuevo contexto.

5. **Generación y síntesis de la respuesta:** finalmente, un LLM, integrado en el flujo, sintetiza los resultados validados con el conocimiento interno del modelo, produciendo una respuesta coherente, precisa y alineada con la intención de la consulta inicial.

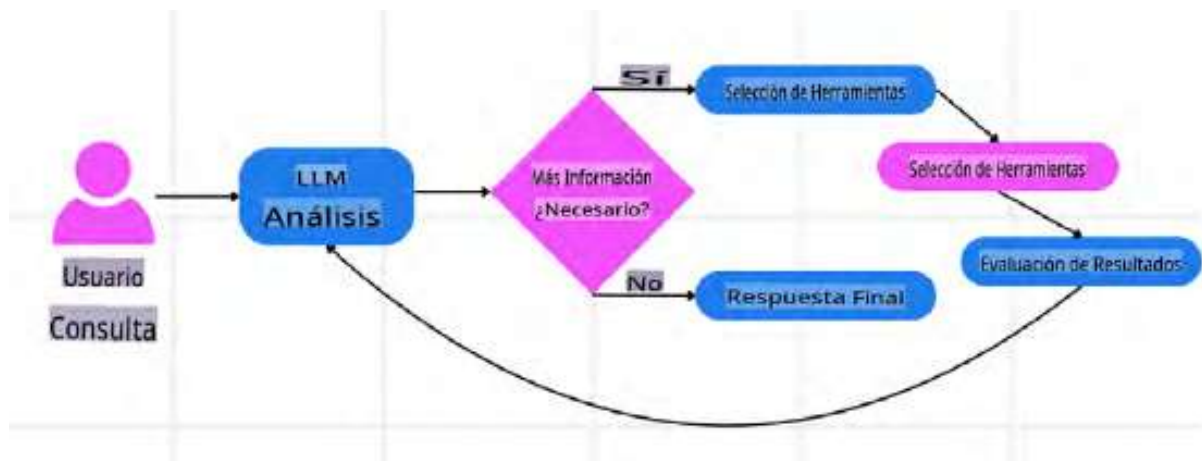


Figura 2.22: Bucle principal de un Agentic RAG

Clasificación de los sistemas Agentic RAG

La clasificación de los sistemas Agentic RAG se fundamenta en su arquitectura y en la forma en que organizan y distribuyen los agentes a lo largo del flujo de trabajo. Singh et al. [44] proponen una taxonomía que distingue distintos tipos de sistemas en función de su complejidad, grado de autonomía y capacidad de adaptación:

- **Single-Agent Agentic RAG (Router):** un único agente se encarga de gestionar la consulta, decidir a qué fuentes de información acudir (bases de datos, APIs, web, entre otras), recuperar los documentos relevantes y sintetizar una respuesta mediante un LLM.
 - **Ventajas:** presenta una arquitectura sencilla, con bajo coste computacional, adecuada para sistemas con un número reducido de fuentes de información o tareas claramente definidas.
- **Multi-Agent Agentic RAG:** múltiples agentes especializados operan de forma paralela. Cada uno de ellos se encarga de una fuente o tarea específica (por ejemplo, uno consulta bases de datos SQL, otro realiza búsquedas web y otro accede a APIs especializadas). Un agente coordinador se encarga de orquestar el flujo general.

- **Ventajas:** proporciona escalabilidad y flexibilidad, siendo especialmente adecuado para consultas complejas que requieren acceder a diversas fuentes heterogéneas.
- **Hierarchical Agentic RAG:** organiza los agentes en una estructura jerárquica. Los agentes de nivel superior actúan como supervisores, distribuyendo tareas entre agentes subordinados, responsables de búsquedas o subtareas específicas. Esta arquitectura facilita la toma de decisiones estratégicas sobre la priorización de fuentes.
 - **Ventajas:** permite gestionar consultas de alta complejidad que requieren múltiples niveles de decisión y coordinación.
- **Corrective Agentic RAG:** incorpora agentes correctivos encargados de evaluar la calidad de los documentos recuperados. Si detectan información irrelevante o insuficiente, refinan la consulta o exploran nuevas fuentes.
 - **Ventajas:** mejora significativamente la precisión del sistema y reduce el riesgo de generar respuestas erróneas o incompletas.
- **Adaptive Agentic RAG:** emplea clasificadores que determinan dinámicamente el tipo de recuperación necesaria según la naturaleza de la consulta. Puede optar por una recuperación simple, una búsqueda compleja en varios pasos o, si es posible, resolver la consulta directamente con el conocimiento interno del LLM.
 - **Ventajas:** optimiza el uso de recursos computacionales, priorizando rapidez o precisión en función de la dificultad de la consulta.
- **Graph-Based Agentic RAG:** combina fuentes estructuradas en forma de grafos (por ejemplo, relaciones entre conceptos o entidades) con documentos no estructurados. Utiliza agentes especializados para recorrer estos grafos y extraer relaciones complejas.
 - **Ventajas:** permite realizar razonamientos complejos sobre las relaciones entre conceptos, siendo especialmente útil en ámbitos como el diagnóstico médico o el análisis jurídico.

Capítulo 3

INVESTIGACIÓN DE USUARIO

En este capítulo se realizará una aproximación progresiva al usuario objetivo, comenzando con un análisis general y concluyendo con el desarrollo de un marco conceptual destinado a modelar al usuario en función de sus objetivos, metas e inquietudes que trascienden el producto.

3.1. Contenidos Populares

Para aproximar las necesidades e intereses de los usuarios, resulta pertinente analizar las consultas realizadas en el principal motor de búsqueda a nivel global. Con este objetivo, se empleará la herramienta Google Trends¹ para examinar las tendencias de búsqueda registradas a lo largo del último año en todo el mundo. La selección de términos se ha realizado en función de su relevancia en relación con el videojuego objeto de análisis y su ecosistema:

- **FC 25:** denominación oficial del videojuego.
- **FC Ultimate Team:** modalidad de juego principal sobre la que se centra esta investigación.
- **Futbin²:** plataforma web de referencia para la comunidad de jugadores, que proporciona información detallada y herramientas de apoyo relacionadas con el videojuego.

¹<https://trends.google.com/trends/>

²<https://www.futbin.com/>

- **SBC (Squad Building Challenges)**: submodalidad altamente popular dentro de FC Ultimate Team, que consiste en la creación de plantillas bajo condiciones específicas.

Principales Contenidos de Búsqueda Relacionados con FC 25

Los términos más buscados relacionados con los anteriores hacen referencia, en su mayoría, a herramientas y plataformas diseñadas para optimizar la experiencia de juego en FC Ultimate Team. Entre ellos destacan:

- **Paletools**³: herramienta destinada a optimizar la gestión de plantillas y SBC.
- **Sofifa FC 25**⁴: plataforma que ofrece estadísticas detalladas de jugadores, útiles tanto para el modo Carrera como para Ultimate Team.
- **Web oficial**⁵: aplicación web oficial de EA Sports que permite gestionar Ultimate Team desde el navegador, facilitando operaciones como compra-venta de jugadores o gestión táctica.
- **Futbin**⁶: plataforma ampliamente utilizada que proporciona estadísticas de jugadores, seguimiento de precios, constructores de plantillas y soluciones para SBC.
- **Squad Builder**: interfaces de creación de plantillas.
- **Easy SBC**⁷: herramienta diseñada para simplificar la resolución de SBC, minimizando costes y tiempos.
- **Auto SBC**⁸: sistema automatizado para completar SBC, reduciendo al mínimo la intervención manual y optimizando el tiempo de ejecución.

Este análisis evidencia que, dentro de la comunidad de FC 25 Ultimate Team, existe una demanda significativa por herramientas web que faciliten la toma de decisiones, permitan una construcción eficiente de plantillas y automaticen tareas repetitivas, tales como los SBC.

³<https://pale.tools/>

⁴<https://sofifa.com/>

⁵<https://www.ea.com/ea-sports-fc/ultimate-team/web-app/>

⁶<https://www.futbin.com/>

⁷<https://www.easysbc.io/>

⁸<https://www.autosbc.app/>

3.2. Aproximación al usuario

El juego cuenta con clasificación PEGI⁹ 3, lo que indica que es adecuado para todas las edades. En consecuencia, el público objetivo puede clasificarse en los siguientes perfiles:

- **Principiantes:** niños y adolescentes que se inician en los videojuegos de fútbol y buscan una experiencia accesible y orientada al aprendizaje.
- **Nivel intermedio:** adolescentes y jóvenes adultos interesados en perfeccionar sus habilidades y estrategias dentro del juego.
- **Usuarios avanzados:** personas con experiencia en videojuegos de fútbol que desean optimizar su rendimiento y competir en niveles superiores.

3.3. Herramientas utilizadas por los usuarios

En esta sección se presentan las herramientas más utilizadas tanto para la obtención de conocimiento aplicable al FC 25 como para la automatización de los procesos más tediosos presentes en el videojuego. Posteriormente, se exponen las debilidades comunes que presentan estas herramientas.

3.3.1. Descripción de las herramientas

A continuación se enumeran las principales herramientas empleadas por los usuarios tanto para ampliar su conocimiento sobre FC Ultimate Team como para automatizar procesos relacionados con su uso:

- **Futbin¹⁰:** proporciona una base de datos completa con estadísticas detalladas de jugadores, seguimiento en tiempo real del mercado, creador de plantillas y soluciones para los SBC.
- **Fut.gg¹¹:** ofrece prácticamente las mismas funcionalidades que *futbin.com*.
- **EasySBC.io¹²:** está especializada en la resolución optimizada de SBC, mostran-

⁹<https://pegi.info/es>

¹⁰<https://www.futbin.com/>

¹¹<https://www.fut.gg/>

¹²<https://www.easysbc.io/>

do plantillas que cumplen con los requisitos al menor coste posible. Dispone de una interfaz más sencilla que el resto, aunque esto compromete en cierta medida la usabilidad.

- **Paletools**¹³: centrada igualmente en la resolución de SBC, permite importar directamente los requisitos de los desafíos y proporciona una o varias soluciones adaptadas.

3.3.2. Limitaciones detectadas en las herramientas existentes

A pesar de la utilidad que ofrecen estas herramientas, presentan diversas carencias que afectan a la experiencia del usuario:

- **Insuficiencia en la toma de decisiones complejas**: carecen de sistemas avanzados de apoyo a la decisión. Las herramientas disponibles únicamente permiten aplicar filtros simples y no contemplan la introducción de múltiples preferencias sobre atributos numéricos o nominales de los jugadores.
- **Falta de inteligencia explicativa**: las soluciones propuestas no incluyen explicaciones que justifiquen las elecciones realizadas ni permiten explorar por qué una alternativa es preferible a otra. Esta carencia dificulta que el usuario comprenda los criterios subyacentes, limitando tanto su aprendizaje como el control sobre el proceso.
- **Baja personalización táctica**: estas herramientas se centran fundamentalmente en criterios básicos como valoración, precio y química, pero no consideran aspectos tácticos avanzados, como los roles, los estilos de juego o las preferencias individuales respecto a planteamientos ofensivos o defensivos. Esto puede provocar que las plantillas sugeridas no se adapten completamente al estilo de juego del usuario.
- **Interfaces poco intuitivas**: la mayoría de las herramientas analizadas presentan interfaces sobrecargadas o poco claras. Esta situación puede generar rechazo en usuarios principiantes y fatiga cognitiva en usuarios habituales.

¹³<https://pale.tools/>

3.4. Modelado del Usuario

En esta sección se presenta el proceso de modelado del usuario, cuyo propósito es identificar y delimitar, en la medida de lo posible, el conjunto de usuarios finales del producto. Para ello, se definen las características relevantes de los usuarios, tales como sus metas, objetivos y las condiciones en las que interactúan con el producto.

3.4.1. Introducción al Modelado del Usuario

Para el modelado del usuario y la elaboración de escenarios, se empleará el método de las personas propuesto por Alan Cooper [45] en 1999. Las personas son representaciones ficticias que representan los distintos tipos de usuarios o audiencias a los que va dirigido un producto o servicio. Estas se centran en los objetivos y necesidades de los usuarios, diferenciándose claramente de la segmentación basada en criterios demográficos propia de la investigación de mercado. De hecho, dos personas pueden pertenecer a grupos demográficos distintos pero compartir comportamientos y objetivos subyacentes similares.

Asimismo, las personas difieren de las descripciones tradicionales de roles o puestos profesionales, ya que dos usuarios con el mismo rol pueden perseguir objetivos distintos o realizar tareas de manera diferente.

El análisis mediante personas se enfoca en la identificación de los siguientes aspectos clave:

- **Metas personales:** aspiraciones del usuario que generalmente trascienden el uso del producto en sí.
- **Experiencia de usuario:** percepciones y sensaciones que experimenta al interactuar con el producto.
- **Objetivos finales:** resultados concretos que el usuario espera obtener mediante el uso del producto.

Los escenarios son relatos breves que describen el uso de un producto en contextos cotidianos, sin detallar interfaces o aspectos técnicos específicos. Generalmente, los escenarios incluyen información como:

- **Contexto de uso:** lugar y dispositivo desde donde se utiliza el producto (por ejemplo, oficina, dispositivo móvil en la calle, equipo compartido).
- **Frecuencia y duración del uso:** desde tareas breves y esporádicas hasta procesos complejos y continuos.
- **Actividades simultáneas:** si el usuario se concentra exclusivamente en la tarea o realiza múltiples actividades con interrupciones frecuentes.
- **Uso combinado de otros productos o software:** interacción entre diferentes herramientas, transferencia de información, etc.
- **Principales actividades y tareas a realizar.**
- **Resultados esperados tras completar la tarea.**

A diferencia de los requisitos formales, que describen las funcionalidades que debe cumplir el sistema, los escenarios permiten comprender el contexto de uso, las implicaciones de las decisiones de diseño y facilitan la priorización de requisitos en el proyecto, alineando las expectativas de las partes interesadas.

3.4.2. Arquetipo 1: El Estudiante

- **Nombre:** Luis.
- **Edad:** 15 años.
- **Ocupación:** Estudiante de secundaria
- **Descripción:** Luis (ver Figura 3.1) es un jugador principiante de FC 25 y estudiante de secundaria que juega de forma ocasional. Disfruta el videojuego porque le permite compartir momentos y divertirse con sus compañeros de clase. Sin embargo, enfrenta dificultades académicas que limitan su tiempo disponible tanto para jugar como para profundizar en estrategias competitivas. Además, se siente aislado en las conversaciones sobre Ultimate Team, debido a su falta de conocimiento en los conceptos clave del juego. Su aspiración es superar en alguna ocasión a sus amigos, quienes cuentan con más experiencia y tiempo para prepararse.
- **Escenario:** Un viernes, tras la jornada escolar, Luis se reúne con sus amigos para jugar FIFA. Consciente de su falta de preparación, accede rápidamente a FC Assistant para consultar estrategias y, debido a su limitado tiempo, solicita

al asistente que le confeccione automáticamente un equipo competitivo. Gracias a esta ayuda, Luis logra vencer a sus amigos por primera vez, quienes quedan sorprendidos por su desempeño y nuevas habilidades en el juego.



Figura 3.1: Arquetipo 1: El estudiante

3.4.3. Arquetipo 2: La Madre

- **Nombre:** Ana.
- **Edad:** 38 años.
- **Ocupación:** Enfermera.
- **Descripción:** Ana (ver Figura 3.2) es una madre que percibe los videojuegos como una herramienta potencial de aprendizaje para su hijo de cinco años. Su hijo muestra un gran interés por el fútbol, los cromos y el seguimiento de los partidos, aunque no está motivado a jugar videojuegos de fútbol. Por ello, Ana se dedica a buscar guías y recursos que le permitan adquirir conocimientos básicos sobre estos juegos para poder enseñarle a su hijo de manera adecuada.

- **Escenario:** Ana y su hijo se sientan juntos frente al ordenador para consultar al asistente diversas dudas relacionadas con el juego. Gracias a las respuestas claras, estructuradas y accesibles que ofrece la herramienta, ambos aprenden de forma conjunta mientras comparten un tiempo de calidad en familia. Incluso es posible que terminen jugando juntos. Ana experimenta una profunda satisfacción al contribuir al desarrollo cognitivo de su hijo mientras fortalecen su vínculo afectivo a través de una actividad compartida.



Figura 3.2: Arquetipo 2: La madre

3.4.4. Arquetipo 3: El Jugador Competitivo

- **Nombre:** Javier.
- **Edad:** 22 años.
- **Ocupación:** Estudiante universitario.
- **Descripción:** Javier (ver Figura 3.3) juega a nivel competitivo en FC 25. En cada partido debe adaptarse al estilo de juego de su oponente, por lo que necesita confeccionar plantillas personalizadas que contrarresten las tácticas rivales y le permitan superarlas. Su objetivo no es adquirir conocimiento sobre el juego, sino

disponer de una herramienta que le facilite seleccionar y comparar jugadores de forma precisa, atendiendo a sus propios criterios estratégicos.

- **Escenario:** Javier tiene programado un partido para mañana frente a un oponente que utiliza habitualmente una formación 4-4-2. Con el fin de contrarrestarlo, decide configurar una plantilla que utiliza la formación 3-5-2, ajustada a sus preferencias tácticas. Para ello, accede a FC Assistant, donde puede diseñar su equipo de manera ágil y eficiente en función de sus prioridades. Configurando cada una de ellas detenidamente para cada posición de la formación. Gracias a esta preparación, logra imponerse a su rival, disponiendo de los jugadores más adecuados para su planteamiento.



Figura 3.3: Arquetipo 3: El Jugador Competitivo

3.4.5. Arquetipo 4: El Entrenador

- **Nombre:** Roberto.
- **Edad:** 40 años.
- **Ocupación:** Entrenador de fútbol juvenil.

- **Descripción:** Roberto (ver Figura 3.4) es entrenador de un equipo de fútbol formado por adolescentes. Su principal objetivo es enseñarles conceptos tácticos para que mejoren su rendimiento en el campo, pero se enfrenta a la falta de atención e interés por parte de sus jugadores, propios de su edad. A pesar de sus esfuerzos, no logra encontrar una metodología que capte su atención y facilite el aprendizaje.
- **Escenario:** Con el fin de motivar a sus jugadores, Roberto organiza competiciones de FIFA entre ellos e introduce el uso de la herramienta FC Assistant para ayudarles a construir sus equipos de forma estratégica. Gracias a la combinación del componente competitivo y el apoyo de la herramienta, los jugadores desarrollan un conocimiento profundo de los conceptos tácticos. Esta experiencia se traduce en una mejora tangible en su rendimiento deportivo, logrando posteriormente ganar un torneo de relevancia.



Figura 3.4: Arquetipo 4: El Entrenador

Capítulo 4

DESARROLLO DEL PROYECTO

La sección de desarrollo incluye el ciclo de vida completo del proyecto propuesto, abarcando desde el análisis inicial del problema y el diseño de la solución, hasta las particularidades de la implementación y las herramientas empleadas para en esta etapa.

4.1. Descripción del Proyecto

El presente proyecto tiene como finalidad el diseño y desarrollo de un sitio web denominado FC Assistant, orientado a asistir a los usuarios en la confección de plantillas para el modo Ultimate Team del videojuego FC 25.

Este proyecto se enmarca dentro del ámbito de los sistemas de información y, más específicamente, se clasifica como un DSS (término que se introduce en la Sección [2.2.4](#)), donde la creación de plantillas se aborda como un problema de toma de decisiones desde una perspectiva innovadora y creativa.

Entre los aspectos técnicos más relevantes del proyecto destacan, por una parte, la escalabilidad del sistema, garantizada mediante una arquitectura distribuida que permite la incorporación modular de nuevas funcionalidades o componentes sin afectar la disponibilidad ni requerir rediseños globales. Por otra parte, resalta la integración de una diversidad de tecnologías y disciplinas del ámbito informático, que han posibilitado el desarrollo de una herramienta accesible, usable e innovadora.

Asimismo, FC Assistant incorpora funcionalidades exclusivas basadas en IA, diseñadas para mejorar la accesibilidad del juego al reducir su complejidad. Estas capa-



Figura 4.1: Logo de FC Assistant

ciudades están orientadas a beneficiar a un amplio espectro de usuarios, incluyendo personas aquellos sin conocimientos previos sobre fútbol, ampliando así el alcance hacia nuevos nichos de mercado. El logo del sitio web se muestra en la Figura 4.1.

4.2. Análisis del Problema

En esta sección se lleva a cabo el análisis del problema. En primer lugar, se identifica y se describe su naturaleza. A continuación, se definen los elementos que intervienen en el mismo y, finalmente, se delimita el espacio de soluciones y se formalizan sus componentes con el fin de facilitar su almacenamiento y tratamiento posterior.

4.2.1. Identificación y Diagnóstico del Problema

El DSS desarrollado en este proyecto aborda la creación de una plantilla de fútbol en base a un presupuesto predefinido, como un problema de toma de decisión. Este problema, de naturaleza combinatoria y alta complejidad, puede descomponerse en subproblemas más manejables mediante la estrategia algorítmica conocida como *divide y vencerás*. Por lo tanto, desde esta perspectiva, la construcción de una plantilla se interpreta como una agregación de múltiples decisiones independientes, una por cada posición dentro de la formación seleccionada.

Cada decisión está condicionada por la táctica individual implícita en la posición del jugador, ya que esta define, al menos en parte, las tareas que debe desempeñar durante el desarrollo del juego. Por ello, la elección de los jugadores más adecuados para cada posición requiere una evaluación diferenciada de sus atributos, en función de las exigencias específicas de su posición en el campo. Esto justifica la necesidad de un sistema de apoyo a la toma de decisiones que permita a los usuarios seleccionar jugadores según cualquier característica relevante, adaptando así la plantilla a su estrategia de juego. En la Figura 4.2, se puede ver un esquema general del problema

de toma de decisión a resolver por el sistema.

Por ejemplo, en una posición como la de extremo, donde la principal responsabilidad es desbordar por las bandas con velocidad y profundidad, se priorizan atributos como la aceleración, la agilidad y el regate. Sin embargo, un centrocampista orientado a la gestión del juego requiere atributos como el pase corto, la visión y el control del balón.

Teniendo en cuenta la naturaleza del problema, anteriormente definida, podemos decir que estamos ante un problema de toma de decisión semiestructurado donde:

- La intuición y el juicio humano desempeñan un papel fundamental, complementándose con herramientas analíticas.
- No existe un procedimiento único ni claramente definido para su resolución, dado que, aunque las posibles acciones están establecidas, no se determina ni el orden ni la frecuencia con la que deben ejecutarse para alcanzar una decisión óptima.

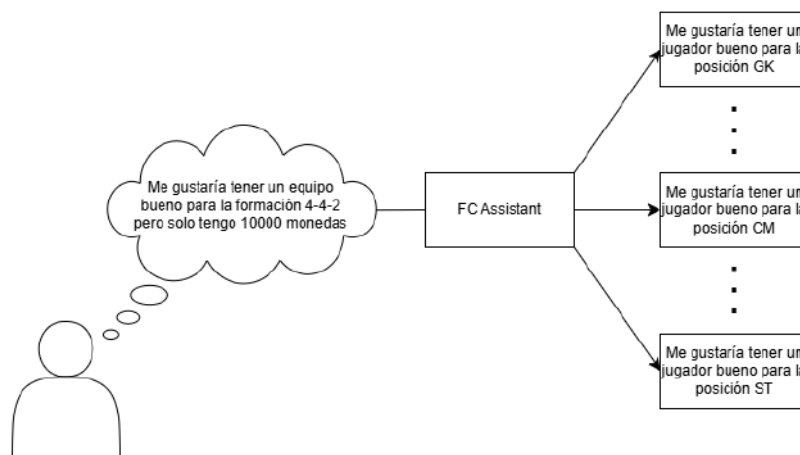


Figura 4.2: Estructura del problema de toma de decisión

4.2.2. Definición de los elementos del Problema

Siguiendo el procedimiento habitual empleado en la resolución de problemas de toma de decisión, tal como se expone en la Sección 2.2.1, el primer paso consiste en identificar todos los elementos relevantes que intervienen en el problema, los cuales son los siguientes:

- **Formación:** define la táctica individual de los jugadores en función de la posición que ocupan dentro de la misma. Además, influye en la táctica colectiva del equipo, ya que algunas formaciones presentan un perfil más ofensivo o defensivo. La definición técnica puede consultarse en la Sección [2.1.1](#).
- **Presupuesto:** establece el valor máximo permitido para la suma de los precios de los jugadores que componen la formación.
- **Prioridades por posición:** representan las preferencias asignadas por el usuario a cada uno de los criterios considerados en cada posición de la plantilla. En la interfaz web se denominan *importancias* para facilitar la comprensión intuitiva del concepto por parte de los usuarios.
- **Requisitos por posición:** corresponden a los filtros o condiciones que debe cumplir el jugador seleccionado para ocupar una posición específica dentro de la formación.
- **Química:** medida que cuantifica la sinergia entre los jugadores, calculada de acuerdo con las reglas formalizadas en la sección de antecedentes de este trabajo (concretamente en la Sección [2.1.2](#)). Esta métrica, acotada en el intervalo $[0, 33]$, contribuye a potenciar los atributos individuales de los jugadores.

4.2.3. Definición de las Alternativas

Basándonos en el diagnóstico del problema podemos identificar dos niveles de selección de alternativas:

- **Nivel Superior:** hace referencia al problema principal, la creación de la plantilla y se compone por todas las combinaciones de las alternativas o jugadores del nivel inferior aportando un elemento de cada uno de los problemas de toma de decisión para cada una de las posiciones.
- **Nivel Inferior:** se refiere al problema de seleccionar un jugador para una posición y las alternativas que lo forman son los jugadores que puedan jugar en la posición.

Los dos niveles tienen un conjunto finito de alternativas, por lo que podemos afirmar que el problema de toma de decisión es discreto. Aunque debemos tener en cuenta las siguientes realidades:

- **Los problemas de toma de decisión del nivel inferior poseen un número pequeño de alternativas manejable por un modelo de toma de decisión básico:** suponiendo el peor de los casos en el que la totalidad de jugadores presentes en el juego se contemplen en el proceso de toma de decisión tendríamos 20000 alternativas. Esta cifra se ha estimado a partir del número actual de jugadores (15000), al que se ha añadido un margen de crecimiento de 500 jugadores para contemplar posibles ampliaciones futuras.
- **El número de alternativas del problema de toma de decisión del nivel superior nos obliga a emplear métodos de optimización heurísticos:** a pesar de ser discreto, al tener tantas alternativas nos vemos obligados a emplear un método de optimización heurístico para la correcta exploración y explotación de un espacio tan grande de soluciones posibles inabarcable utilizando un modelo estándar. Considerando, en un caso favorable, que cada jugador puede desempeñar cualquier posición (lo que reduce el tamaño del espacio en comparación con un enfoque en el que las posiciones fueran independientes que es el peor de los casos), y suponiendo un conjunto reducido de 3000 jugadores como candidatos por posición (un número bajo, dado que suponemos 20000 jugadores en total), el tamaño total del espacio de soluciones para 11 posiciones lo podemos observar en la Ecuación 4.1.

$$\binom{3000}{11} = \frac{3000!}{11! \cdot (3000 - 11)!} = \frac{3000!}{11! \cdot 2989!} = 4,35719 \times 10^{30} \quad (4.1)$$

Incluso suponiendo que nuestro ordenador fuese capaz de evaluar 10^{12} combinaciones por segundo, necesitaría aproximadamente $1,38 \times 10^{11}$ años para explorar exhaustivamente todo el espacio de soluciones. Si comparamos esta cifra con la edad estimada del universo, $1,37 \times 10^{10}$ años, resulta evidente que el tiempo requerido sería desproporcionadamente elevado.

4.2.4. Formalización de los elementos del problema para su procesamiento

A continuación, se define una representación formal de cada uno de los elementos que componen el problema de toma de decisiones, con el objetivo de facilitar su almacenamiento en estructuras de datos y su integración en los modelos de decisión.

- **Formación:** una formación de fútbol puede representarse mediante un vector cuyos elementos son abreviaturas que identifican posiciones específicas. Estas

posiciones pueden disponerse siguiendo un criterio espacial, como de izquierda a derecha y de defensa a ataque.

- **Prioridades y requisitos:** el conjunto de prioridades y requisitos puede representarse mediante un vector de listas, de forma que a cada posición del vector de formación le corresponde una lista asociada con sus criterios específicos.
- **Jugadores:** cada jugador se representa formalmente mediante un vector numérico, donde cada componente corresponde a un atributo particular del jugador. Esta representación permite considerar a los jugadores como alternativas dentro del modelo de toma de decisiones. Se distinguen los siguientes casos:
 - **Atributos nominales:** se emplea codificación *one-hot*, que consiste en asignar el valor 1 a la posición del vector correspondiente a una característica cuando el jugador la posee, y 0 en caso contrario. La problemática asociada al uso de esta codificación se detalla posteriormente en la Sección 4.7.1.
 - **Atributos ordinales:** se asignan valores numéricos que reflejan el orden inherente de las categorías.
 - **Atributos numéricos.**

4.3. Especificación de Requisitos

En esta sección se detallan los requisitos del sistema, abarcando tanto las funcionalidades que deberá ofrecer el sistema objetivo como las características cualitativas que definirán su comportamiento y rendimiento.

4.3.1. Requisitos Funcionales

Los requisitos funcionales del sistema describen las acciones y capacidades que la aplicación debe ofrecer para cumplir con sus objetivos. A continuación, se detallan los principales requisitos identificados:

- **Gestión de plantillas:**
 - Permitir la creación de plantillas de forma manual o automática, a partir de las prioridades y filtros establecidos por el usuario, orientadas al modo Ultimate Team.

- Ofrecer la posibilidad de seleccionar entre distintas formaciones tácticas predefinidas (4-3-3, 4-4-2, entre otras).
 - Permitir el completado manual o el autocompletado de plantillas incompletas, es decir, aquellas a las que les falte uno o varios jugadores por definir.
 - Facilitar la configuración de un presupuesto máximo para la creación de plantillas.
 - Permitir al usuario especificar si las prioridades asignadas a los distintos criterios se interpretan como criterios de coste o de beneficio.
 - Ofrecer la opción de excluir el precio de determinados jugadores del cálculo del coste total de la plantilla.
- **Visualización de la información:**
- Mostrar las cartas de los jugadores respetando su diseño visual característico.
 - Generar resúmenes visuales, como gráficos o tablas comparativas, con las estadísticas de los jugadores.
- **Asistencia inteligente:**
- Incorporar un chatbot que atienda consultas relacionadas con el juego.
 - Proporcionar análisis automatizados de las plantillas creadas por los usuarios.
 - Sugerir estrategias de juego adaptadas a la plantilla seleccionada.
 - Permitir la creación automática de una plantilla a partir de un *prompt*¹ proporcionado por el usuario.
- **Información de mercado:**
- Mostrar los precios actualizados de los jugadores en el mercado.

4.3.2. Requisitos No Funcionales

Los requisitos no funcionales establecen las características cualitativas que debe presentar el sistema, describiendo cómo debe operar en lugar de qué debe hacer. Estos requisitos abordan aspectos como el rendimiento, la seguridad, la usabilidad, la disponibilidad y la escalabilidad, entre otros. Su objetivo es asegurar que el sistema

¹Un prompt es el texto que se le proporciona a un LLM para que genere una respuesta o realice una tarea específica, es decir, es la forma en que el usuario comunica con el LLM.

cumpla con los estándares de calidad establecidos y ofrezca una experiencia satisfactoria al usuario final.

- **Interfaz intuitiva y visualmente atractiva.** El sistema debe contar con una interfaz accesible, clara y visualmente cuidada, adecuada al perfil del usuario objetivo.
- **Lenguaje accesible y adaptado al usuario.** El lenguaje utilizado en la interfaz debe evitar tecnicismos propios del ámbito de la toma de decisiones, sustituyéndolos por términos comprensibles y familiares para el usuario final, facilitando así la interacción con el sistema.
- **Escalabilidad.** El sistema debe estar preparado para escalar horizontalmente, de manera que sea capaz de gestionar eficientemente grandes volúmenes de datos, como más de 18.000 jugadores, precios históricos, consultas concurrentes al asistente virtual y múltiples solicitudes de ejecución de algoritmos de optimización al mismo tiempo.
- **Modularidad.** La arquitectura debe estar diseñada de forma modular, permitiendo la incorporación de nuevos servicios o funcionalidades sin requerir modificaciones sustanciales en el resto del sistema.
- **Actualización dinámica de datos.** El sistema debe disponer de mecanismos automáticos de actualización periódica de datos, como información de jugadores y precios de mercado, mediante técnicas de *scraping* o servicios externos, garantizando que este proceso no interrumpa la experiencia del usuario.
- **Interacción personalizada.** La interfaz debe ofrecer opciones de personalización en función del nivel de experiencia del usuario (novato, intermedio o experto), adaptando el contenido y las sugerencias a sus necesidades particulares.
- **Internacionalización.** El sistema debe estar preparado para facilitar su adaptación a diferentes idiomas, aunque la primera versión se implemente en español.

4.4. Análisis del Sistema

En esta sección se analiza el sistema partiendo de una visión general, examinando la interacción entre los distintos elementos que lo forman, estudiando los casos de uso, describiendo el diseño de cada uno de sus componentes y, finalmente, presentando aspectos relevantes sobre su implementación.

4.4.1. Arquitectura del Sistema

El sistema propuesto se organiza en módulos o subsistemas desarrollados de forma independiente, siguiendo una estructura modular que facilita su mantenimiento y evolución. La mayoría de estos módulos se corresponden con los componentes tradicionales de un DSS. No obstante, la arquitectura ha sido ampliada para incorporar nuevas funcionalidades, como la integración de un asistente virtual y su despliegue como sistema de información accesible a través de la web.

Los módulos que integran el sistema son los siguientes:

- **Módulo de Recuperación y Almacenamiento de Datos (MRAD):** encargado de las funciones propias del subsistema de datos en un DSS. Este módulo se compone de los siguientes elementos:
 - Conjunto de scrapers.
 - Base de datos distribuida.
- **Módulo de Gestión de Peticiones (MGP):** actúa como interfaz de usuario y componente de comunicaciones del sistema, gestionando las solicitudes de los usuarios, dirigiendo dichas peticiones a los servidores correspondientes y mostrando los resultados a través de la interfaz. Está formado por:
 - Servidor web.
 - Servidor back-end.
 - Servidor Redis.
- **Módulo de Realización de Tareas (MRT):** desempeña el papel del subsistema de modelos en la arquitectura clásica de un DSS. Está compuesto por:
 - Servidores workers.
 - Base de datos local reducida.
- **Módulo del Asistente (MA):** satisface la necesidad de incorporar conocimiento experto durante el proceso de toma de decisiones, facilitando la interacción con el usuario. Está integrado por:
 - Servidor RAG.
 - Servidor de almacenamiento documental.
 - Colección de documentos.

- Modelo de lenguaje natural.

Podemos visualizar la interacción del sistema de manera general, en la Figura 4.3.

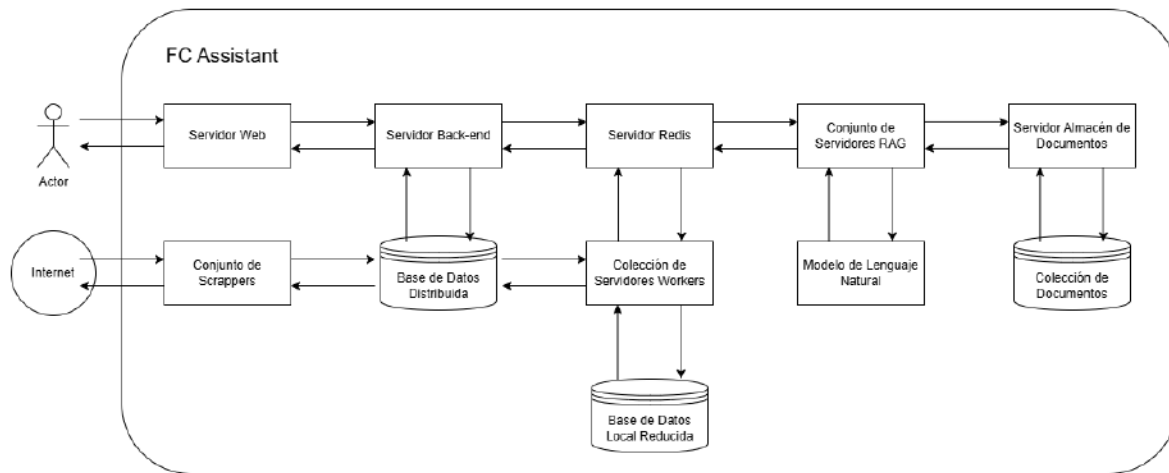


Figura 4.3: Diagrama de interacción del sistema

4.4.2. Casos de Uso

Un caso de uso describe una secuencia de acciones llevadas a cabo por un actor (usuario o sistema externo) con el objetivo de lograr una meta específica mediante la interacción con el sistema. A continuación, se definen los principales casos de uso contemplados en el sistema, todos ellos representados de forma general en la Figura 4.4.

Caso de Uso 1: Crear una plantilla

- **Actor principal:** usuario.
- **Objetivo:** obtener una plantilla completa que respete el presupuesto máximo definido, teniendo en cuenta las prioridades y filtros especificados por el usuario para cada posición.
- **Precondiciones:** el usuario ha seleccionado una formación, ha establecido un presupuesto máximo y ha configurado las posiciones correspondientes de la formación.
- **Postcondiciones:** se genera y muestra una plantilla optimizada que cumple con los criterios definidos por el usuario.

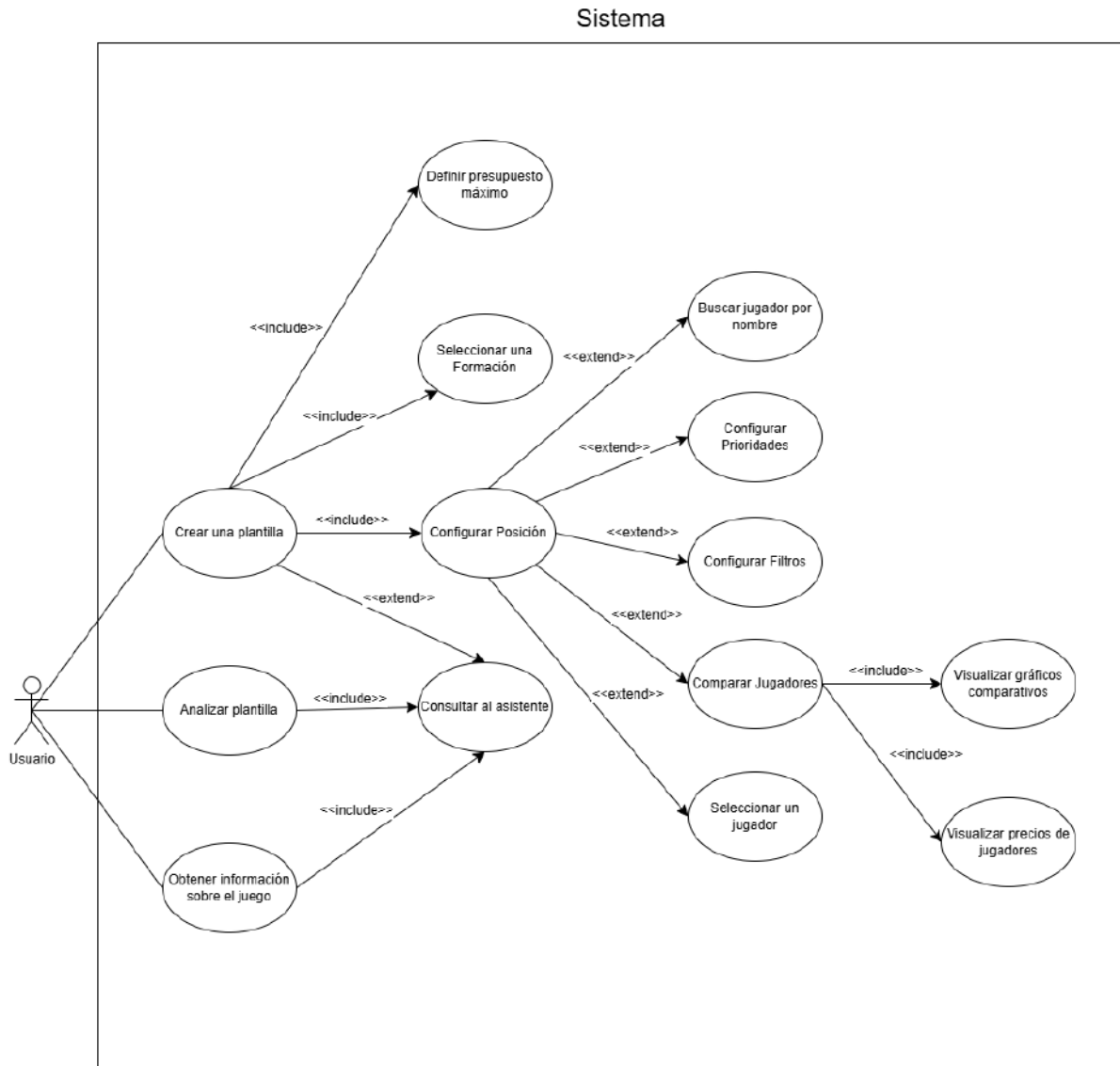


Figura 4.4: Diagrama de caso de uso general

■ Flujo principal de eventos:

1. El usuario accede a la interfaz de creación de plantillas.
2. El usuario selecciona la formación táctica deseada.
3. El usuario introduce el presupuesto máximo permitido para la plantilla.
4. El usuario configura cada una de las posiciones de la formación.
5. El sistema genera y presenta la plantilla correspondiente.

■ Flujos alternativos:

- Si el usuario configura únicamente las prioridades y filtros, sin seleccionar jugadores concretos, podrá solicitar la creación automática de la plantilla. El

sistema ejecutará el algoritmo de optimización correspondiente y mostrará la plantilla resultante.

- Alternativamente, el usuario podrá solicitar la creación automática de una plantilla mediante el asistente virtual, indicando sus preferencias a través de un *prompt*. El sistema procesará la solicitud y generará una plantilla acorde a los requisitos especificados.

Podemos visualizar el diagrama de este caso de uso en la Figura 4.5.

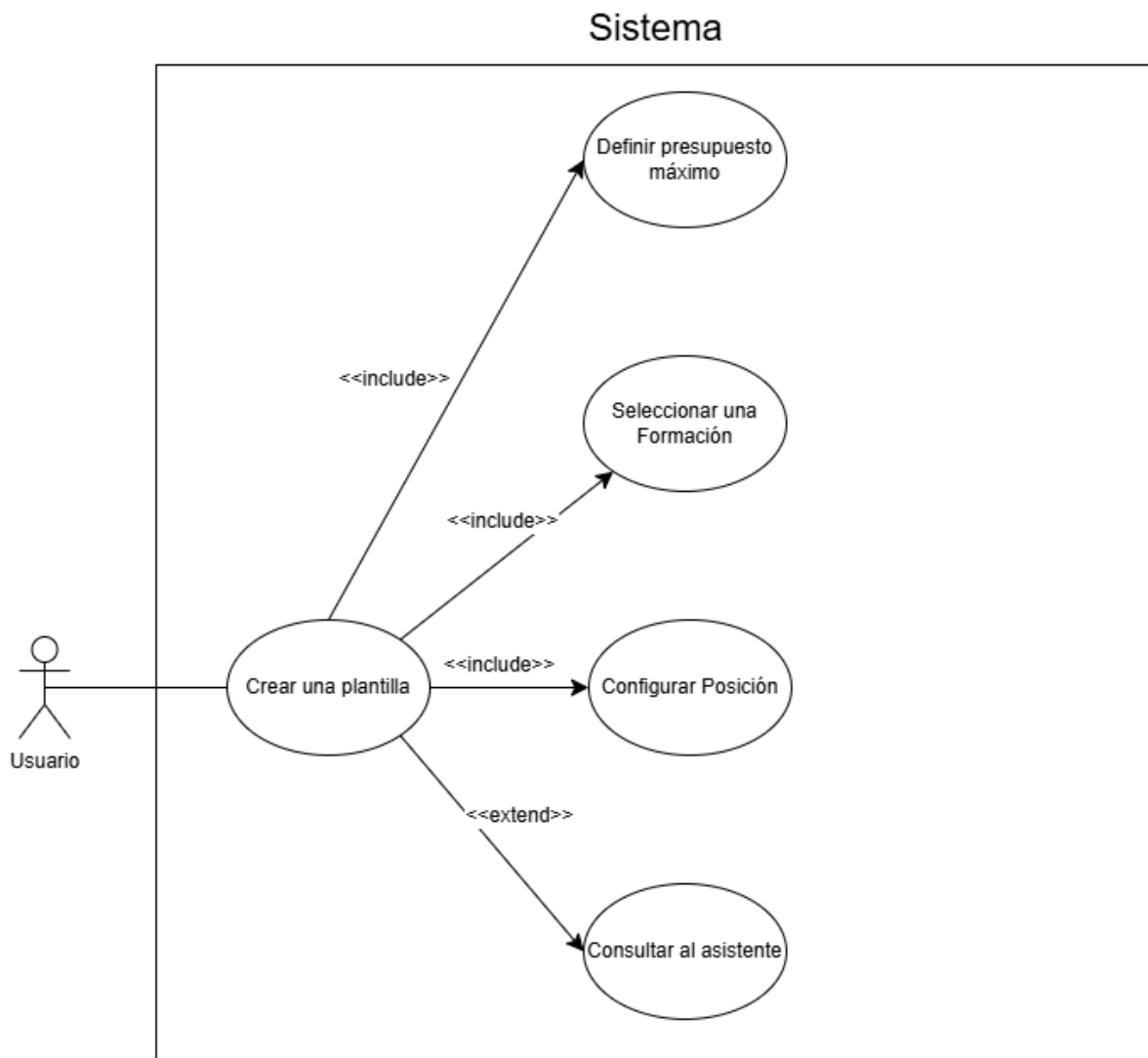


Figura 4.5: Diagrama de caso de uso: Crear un equipo

Caso de Uso 2: Configurar Posición

- **Actor principal:** usuario.

- **Objetivo:** definir el jugador que ocupará una posición determinada dentro de una formación táctica, ya sea seleccionando un jugador específico o configurando filtros y prioridades que permitan determinarlo automáticamente.
- **Precondiciones:**
 - El usuario ha seleccionado una formación táctica.
 - Se ha establecido un presupuesto máximo para la plantilla.
 - La ventana modal correspondiente a la posición que se desea configurar está abierta.
- **Postcondiciones:**
 - El sistema muestra el jugador seleccionado para esa posición o las prioridades configuradas por el usuario.
- **Flujo principal de eventos:**
 1. El usuario accede a la página de creación de plantillas.
 2. Selecciona la formación táctica deseada.
 3. Establece el presupuesto máximo permitido.
 4. Abre la ventana modal correspondiente a la posición que desea configurar.
 5. Realiza las acciones necesarias hasta definir el jugador que ocupará dicha posición.
 6. El sistema actualiza la plantilla mostrando el jugador seleccionado o las prioridades configuradas.
- **Flujo alternativo de eventos:**

Para determinar el jugador que ocupará una posición, el usuario puede realizar cualquiera de las siguientes acciones, en el orden y la cantidad de veces que considere necesario:

 - **Buscar un jugador por nombre:** introducir el nombre de un jugador concreto que tenga en mente para ocupar la posición.
 - **Configurar prioridades:** definir los criterios de preferencia que debe cumplir el jugador deseado.
 - **Aplicar filtros:** restringir el conjunto de jugadores posibles mediante condiciones específicas (por ejemplo, liga, nacionalidad o precio).
 - **Comparar jugadores:** visualizar una comparativa gráfica entre dos jugadores para facilitar la toma de decisiones.

- **Seleccionar jugador:** confirmar la elección pulsando sobre el jugador deseado.

Podemos visualizar el diagrama de este caso de uso en la Figura 4.6.

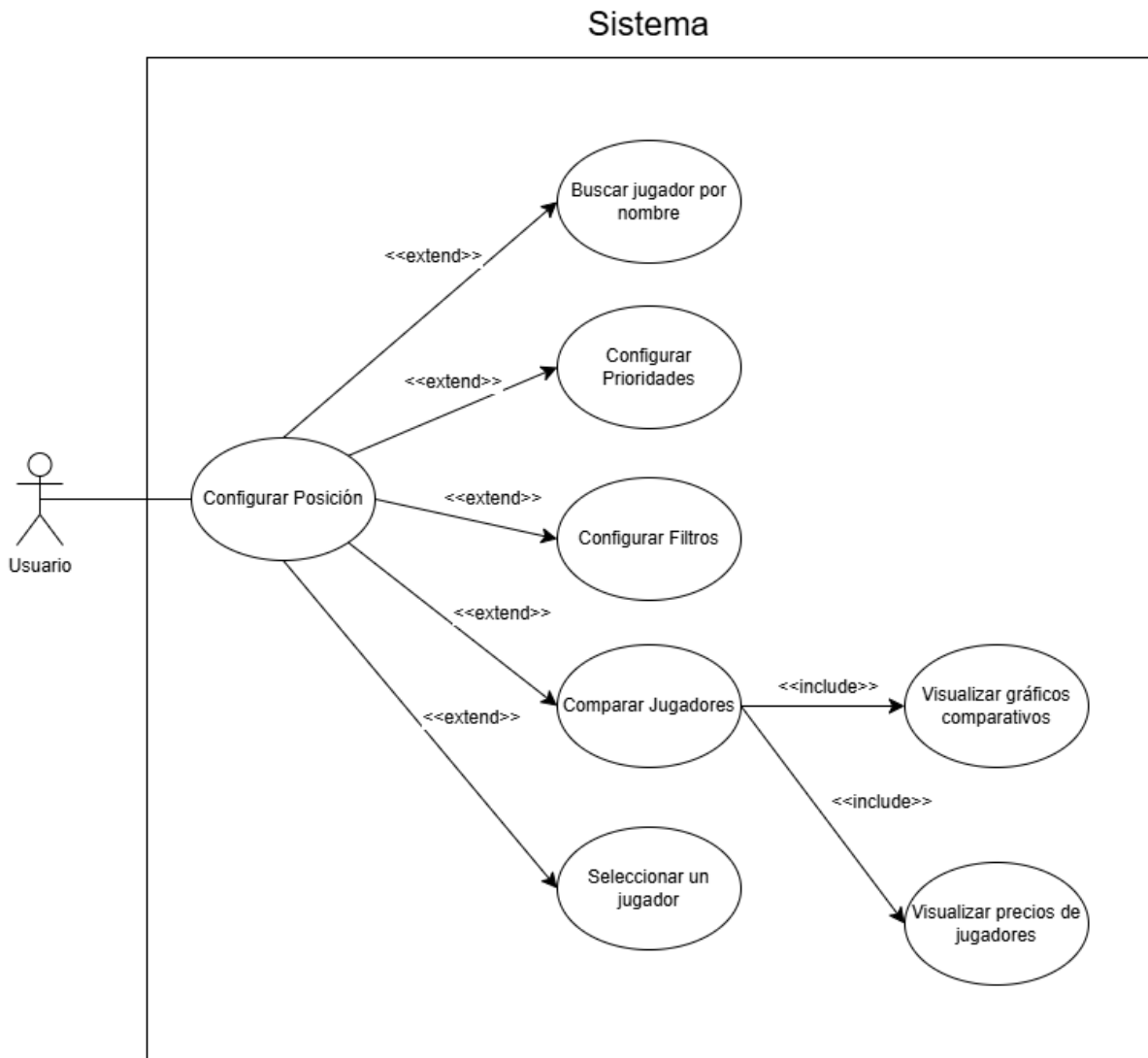


Figura 4.6: Diagrama de caso de uso: Configurar una posición

Caso de Uso 3: Consultar al Asistente

- **Actor principal:** usuario.
- **Objetivo:** formular consultas relacionadas con el videojuego, solicitar un análisis de una plantilla existente con recomendaciones personalizadas o generar automáticamente una plantilla proporcionando las prioridades mediante un *prompt*.
- **Precondiciones:** la ventana del asistente virtual debe estar visible y operativa.
- **Postcondiciones:** el sistema proporciona una respuesta adecuada a la consulta realizada por el usuario.
- **Flujo principal de eventos:**
 1. El usuario accede a la interfaz del asistente virtual.
 2. El usuario decide el tipo de consulta que desea realizar.
 3. El usuario introduce su consulta en el campo de entrada habilitado.
 4. El usuario envía la consulta mediante el botón correspondiente.
 5. El sistema procesa la consulta y muestra la respuesta generada.
- **Flujos alternativos:**
 - Durante la interacción con el asistente, el usuario puede formular distintas consultas, tales como:
 - Solicitar un análisis detallado de su plantilla actual, incluyendo fortalezas y debilidades.
 - Obtener información general sobre el funcionamiento del videojuego o sobre aspectos concretos del modo Ultimate Team.
 - Recibir recomendaciones tácticas personalizadas basadas en la plantilla proporcionada.
 - Solicitar la creación automática de una plantilla, indicando preferencias o prioridades mediante un *prompt* escrito.

Podemos visualizar el diagrama de este caso de uso en la Figura 4.7.

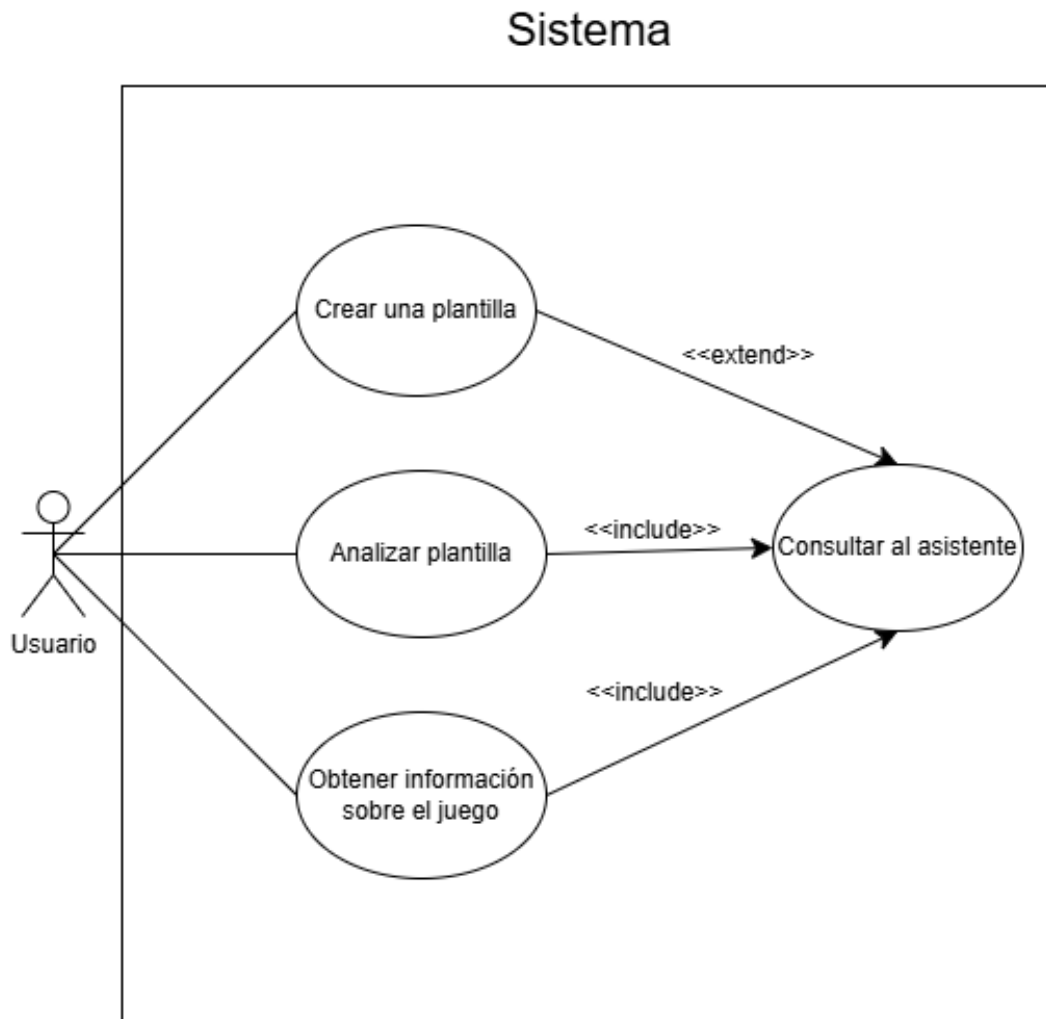


Figura 4.7: Diagrama de caso de uso: Consultar Asistente

4.5. Diseño del MRAD

En esta sección, se va a detallar todas las decisiones de diseño tomadas a la hora de definir el MRAD, tanto sus componentes, como las particularidades y configuraciones de los mismos.

4.5.1. Conjunto de Scrapers

El sistema dispone de un conjunto de dispositivos dedicados a la ejecución de técnicas de *web scraping*, que consisten en la consulta automatizada de uno o varios sitios web con el fin de extraer información relevante. En el contexto de este proyecto,

dichos scrapers se encargan de acceder periódicamente a páginas web relacionadas con Ultimate Team para recopilar datos actualizados, que posteriormente son almacenados en la base de datos del sistema.

Para garantizar una alta disponibilidad y minimizar el tiempo de obtención de la información sobre los jugadores, lo ideal sería emplear múltiples dispositivos trabajando de manera distribuida. No obstante, en el prototipo actual, el algoritmo concurrente está diseñado para ejecutarse en un único dispositivo.

Para alcanzar la escalabilidad deseada, sería necesario adaptar el algoritmo para que funcione mediante paso de mensajes entre varias máquinas, facilitando así una distribución eficiente de la carga de trabajo. Esta comunicación entre dispositivos podría implementarse, por ejemplo, utilizando directamente la base de datos como medio de intercambio de mensajes.

Para la obtención automatizada de información relativa al modo Ultimate Team, se ha seleccionado la plataforma FUT.GG², cuya página de jugadores se muestra en la Figura 4.8. Esta elección se fundamenta en que dicho sitio proporciona todos los datos necesarios sobre los jugadores, cuenta con una estructura de páginas coherente y homogénea, y facilita la superación eficiente de las medidas de seguridad implementadas para evitar el scraping.

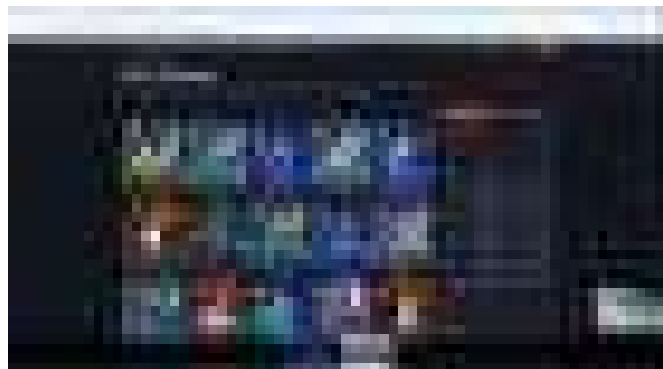


Figura 4.8: Sitio web FUT.GG

El sitio web implementa mecanismos de seguridad basados en la monitorización de la cantidad y frecuencia de solicitudes realizadas desde una misma dirección IP pública. Cuando se detecta un patrón de accesos excesivamente constante o acelerado, la plataforma puede catalogar dicha IP como sospechosa de actividad automatizada, activando mecanismos de verificación como captchas. Esta situación se produjo durante las primeras pruebas de scraping, momento en el cual las páginas comenzaron a solicitar la resolución de captchas para confirmar que el acceso estaba siendo realizado por un usuario legítimo.

²<https://www.fut.gg/>

Para mitigar esta limitación, se empleó la herramienta Urban VPN³ (ver Figura 4.9), un servicio gratuito que permite establecer conexiones privadas virtuales desde diferentes ubicaciones geográficas. Si bien al ser una herramienta gratuita no garantiza una privacidad absoluta frente a la propia plataforma de la VPN, pero resulta adecuada en este caso, ya que permite actuar como usuarios anónimos frente a FUT.GG y evitar bloqueos relacionados con la dirección IP original.



Figura 4.9: Logo de Urban VPN

El principal desafío en las tareas de web scraping es la obtención de contenidos dinámicos, ya que este tipo de información no se encuentra directamente en el código fuente de la página, sino que se genera a través del renderizado realizado por el navegador mediante tecnologías como JavaScript.

En este caso particular, datos relevantes como los precios históricos de los jugadores, sus roles, estilos de juego o posiciones principales se cargaban dinámicamente mediante JavaScript, lo que imposibilitaba su obtención mediante solicitudes HTTP simples. Ante esta situación, se tomaron dos decisiones estratégicas para abordar el problema:

- **Emplear la biblioteca Playwright⁴ para Python.** Esta herramienta permite controlar un navegador real de forma automatizada, posibilitando así la carga y extracción de contenidos dinámicos. No obstante, su uso conlleva una mayor carga computacional y una velocidad inferior respecto a los métodos tradicionales basados en solicitudes HTTP (GET).
- **Dividir el proceso de extracción de información en etapas o procesos independientes.** Con el objetivo de optimizar el rendimiento del proceso, se priori-

³<https://www.urban-vpn.com/>

⁴<https://playwright.dev/python/>

zó la obtención de aquellos datos accesibles directamente mediante solicitudes GET, reservando el proceso que usa renderizado dinámico exclusivamente para aquellos elementos que lo requerían.

Además de los procesos destinados a la obtención de contenido dinámico o actualizado periódicamente, ha sido necesario diseñar algoritmos específicos para la extracción de contenido estático, es decir, información permanente que no está sujeta a cambios o ampliaciones en el tiempo.

Entre los elementos considerados como contenido estático se incluyen, por ejemplo, las imágenes de las banderas de los diferentes países, los escudos de los clubes, los emblemas de las ligas y los iconos correspondientes a los distintos estilos de juego.

El sistema de scraping ha sido diseñado para ejecutarse de forma concurrente en una única máquina, estructurando su funcionamiento mediante cuatro tipos diferenciados de hilos, cada uno encargado de una tarea específica dentro del proceso de extracción de datos:

- **Hilo de escaneo de jugadores.** Responsable de recorrer las páginas que exponen a los diferentes jugadores, aplicando un filtro por valoración global en orden descendente. El proceso comienza con los jugadores de mayor valoración (por ejemplo, 90) y continúa de forma secuencial hasta alcanzar las valoraciones más bajas. Para cada jugador identificado se registran los siguientes datos:
 - ID del jugador: generado mediante la concatenación normalizada de los atributos nombre, tipo de carta y valoración. Por ejemplo: *lionelmessiultimatesuccession92*.
 - URL de la página del jugador.
 - URL de la imagen del jugador.
 - Valoración global del jugador.
- **Hilo de descarga de imágenes de jugadores.** Encargado de obtener las imágenes a partir de las URLs proporcionadas por el hilo de escaneo. Aunque la operación se reduce a una solicitud GET, se implementa como un hilo independiente debido a la importancia de estos elementos visuales en la presentación del contenido. Las imágenes de los jugadores representan un recurso fundamental para la experiencia del usuario y, por extensión, para la fidelización del mismo.

- **Hilo de extracción de información estática de jugadores.** Utilizando las URLs obtenidas por el hilo de escaneo, este hilo realiza solicitudes GET a las páginas individuales de los jugadores con el fin de extraer toda aquella información que sea accesible directamente sin necesidad de renderizado dinámico. Entre los datos recopilados se incluyen:
 - Nombre del jugador.
 - Atributos numéricos.
 - Pierna dominante o pierna débil.
 - Otros metadatos relevantes accesibles por solicitud GET básica.

- **Hilo de extracción de información dinámica de jugadores.** Dedicado a la obtención de aquellos datos que requieren renderizado mediante JavaScript. Para esta tarea se emplea la biblioteca *Playwright*, que permite interactuar con un navegador controlado por código. Este hilo extrae información como:
 - Evolución de los precios del jugador a lo largo del tiempo.
 - Posición principal en el terreno de juego.
 - Estilos de juego asociados.
 - Otros contenidos dinámicos relevantes.

Tanto en el *hilo de extracción de información estática de jugadores* como en el *hilo de extracción de información dinámica de jugadores* se aplica un criterio de priorización orientado a optimizar la relevancia del contenido obtenido. En ambos casos, se concede preferencia a los jugadores con una mayor valoración global, partiendo de la hipótesis razonable de que los jugadores mejor valorados generan un mayor interés por parte de los usuarios y, por tanto, constituyen un contenido prioritario para mantener actualizado.

Adicionalmente, en el caso del *hilo de extracción de información dinámica de jugadores*, cuando dos jugadores presentan la misma valoración global, se prioriza la extracción de aquel cuya información lleve más tiempo sin ser actualizada, garantizando así una mayor coherencia temporal en los datos almacenados.

Por otro lado, es importante destacar que el *hilo de extracción de información estática de jugadores* únicamente realiza el scraping de aquellos jugadores que aún no han sido incorporados a la base de datos. En cambio, el *hilo de extracción de información dinámica de jugadores* requiere necesariamente que el jugador ya se encuentre registrado previamente, ya que la información dinámica se asocia a registros existentes en la colección.

4.5.2. Base de Datos Distribuida

La arquitectura del sistema está diseñada para incorporar una base de datos distribuida, dado el volumen considerable de información que se prevé gestionar. Si bien se contempla la implementación de dicha infraestructura en un entorno real, en esta instancia se ha optado por simular su despliegue mediante el uso de contenedores Docker⁵.

Los datos almacenados se pueden clasificar en las siguientes categorías:

- **Datos de los usuarios:** por un lado, se almacenará la información estrictamente necesaria para la autenticación de los usuarios, como credenciales de inicio de sesión. Además, se prevé la posibilidad de registrar información relativa a la interacción de los usuarios con el sistema, con el fin de llevar a cabo análisis de comportamiento o analítica web en caso de ser necesario. El volumen de estos datos dependerá del crecimiento progresivo del sitio web y de su grado de popularidad. Si bien inicialmente no resulta imprescindible recurrir a una base de datos distribuida para esta finalidad, pero se considera adecuado contemplar desde el diseño la posibilidad de una futura escalabilidad horizontal del sistema, especialmente si el proyecto alcanza una expansión internacional o un aumento considerable de usuarios.

Cabe señalar que, en el presente trabajo, la gestión y almacenamiento de información relacionada con los usuarios y sus interacciones ha sido excluida, dado que se ha priorizado el desarrollo de contenidos y funcionalidades esenciales según lo establecido en la Sección 1.2.

- **Contenido del sitio web:**
 - **Información sobre los jugadores:** se estima que el número total de jugadores ronda los 18.000. Esta cifra, por sí sola, no justifica la necesidad de recurrir a una base de datos distribuida, ya que puede ser gestionada eficientemente mediante arquitecturas convencionales.
 - **Histórico de precios de los jugadores:** este apartado representa el principal desafío en términos de almacenamiento. Considerando aproximadamente 18.000 jugadores, con una frecuencia estimada de 24 registros diarios por jugador a lo largo de todo un año, el volumen de datos esperado lo podemos consultar en la Ecuación 4.2.

⁵<https://www.docker.com/>

$$N_{\text{precios}} = 365 \times 24 \times 18,000 \times k = 157,680,000 \times k \quad (4.2)$$

donde k representa un factor de ajuste en función del número real de precios que finalmente se decida almacenar por jugador. Este volumen de registros hace imprescindible el uso de una base de datos distribuida para garantizar una adecuada capacidad de almacenamiento y eficiencia en el acceso a la información.

Además, conservar un historial completo de precios resulta fundamental para habilitar funcionalidades avanzadas en el futuro, tales como el análisis de series temporales, estudios de comportamiento de mercado, o el entrenamiento de modelos predictivos destinados a generar recomendaciones personalizadas para los usuarios.

El sistema de gestión de bases de datos seleccionado para este proyecto es MongoDB⁶ (ver Figura 4.10), fundamentado en diversas razones técnicas y estratégicas. En primer lugar, se trata de una base de datos NoSQL orientada a documentos, lo que proporciona una notable flexibilidad en el almacenamiento y gestión de datos con estructuras heterogéneas o en constante evolución. Esta característica resulta especialmente adecuada para un sistema que debe adaptarse a cambios frecuentes en el tipo o formato de los datos.



Figura 4.10: Logo de MongoDB

Asimismo, MongoDB ofrece soporte nativo para la escalabilidad horizontal mediante el mecanismo de sharding, funcionalidad clave para garantizar el correcto funcionamiento del sistema ante el previsible aumento del volumen de datos, particularmente en lo relativo al almacenamiento del histórico de precios de los jugadores. Esta capacidad permite distribuir los datos de manera eficiente entre múltiples nodos, asegurando tiempos de respuesta adecuados incluso en entornos con grandes cantidades de información. La arquitectura de esta base de datos se divide en los siguientes componentes (ver Figura 4.11):

- **Servidores de configuración:** tres servidores de configuración forman un conjunto de réplicas que mantienen tanto la metainformación del clúster, como el

⁶<https://www.mongodb.com/>

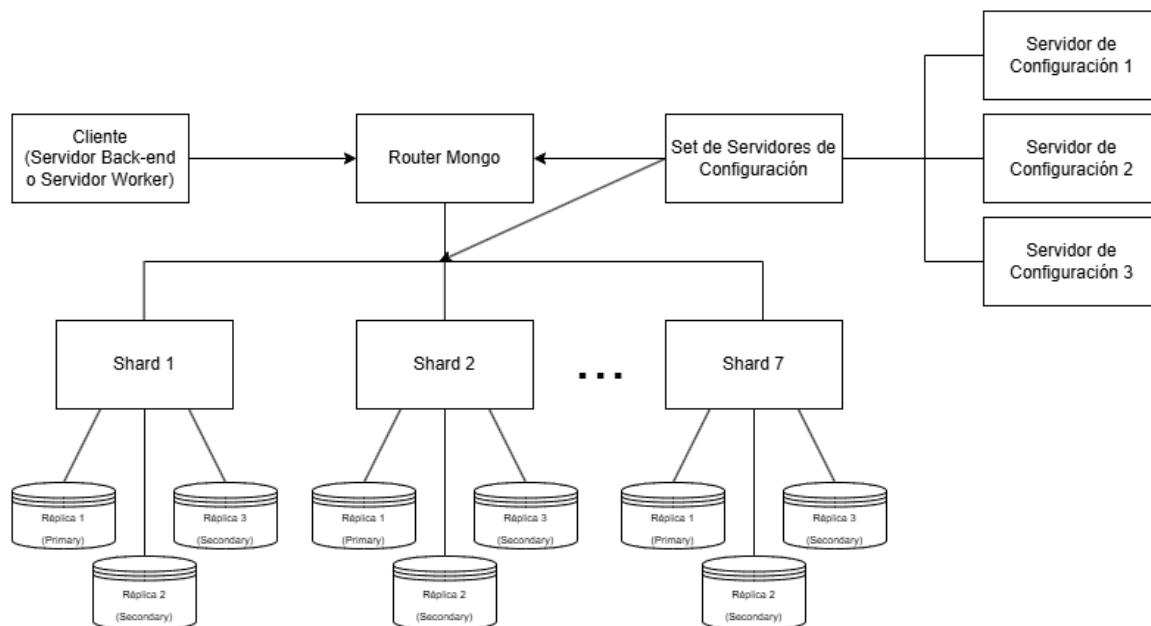


Figura 4.11: Arquitectura de la Base de Datos

estado de los shards, la distribución de los datos y las configuraciones internas necesarias para el enrutamiento. Es importante destacar que MongoDB no admite configuraciones con dos servidores de configuración, ya que no se puede alcanzar el quórum necesario para operaciones críticas.

- **Shards:** se definen siete conjuntos de réplicas que actúan como shards. Cada shard está compuesto por tres nodos que garantizan redundancia y tolerancia a fallos. Los shards almacenan los datos de forma distribuida, permitiendo el particionado horizontal (sharding).
- **Router (mongos):** un router central actúa como punto de entrada al sistema. Es el componente responsable de recibir las peticiones del cliente y dirigir las al shard correspondiente, basándose en la información proporcionada por los servidores de configuración.

El diseño de la base de datos contempla tres colecciones⁷ principales, encargadas de almacenar la información necesaria para que el sistema lleve a cabo las funcionalidades previstas. Las colecciones definidas son las siguientes (ver Figura 4.12):

- **Colección de Escaneo de Jugadores.** Esta colección almacena los jugadores que han sido detectados por el *hilo de escaneo de jugadores*. Sus principales atributos son:

⁷Las colecciones relacionadas con los usuarios y con el registro de su interacción con el sistema han sido excluidas en esta fase, dado que no forman parte del alcance del presente trabajo.

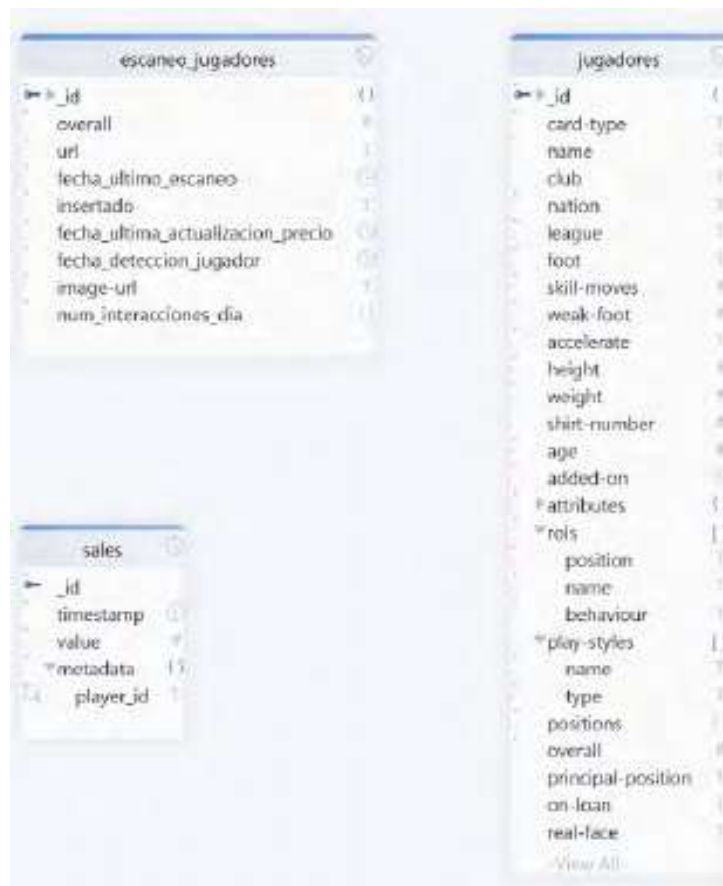


Figura 4.12: Esquemas de las colecciones de la base de datos

- **Valoración global o media (overall).** Permite establecer un criterio de priorización a la hora de programar la extracción de información detallada del jugador.
 - **Fecha de última actualización de precios.** Almacena la fecha de la última actualización de los precios asociados al jugador, facilitando la planificación de futuras extracciones de información dinámica.
 - **Insertado.** Indica si el jugador ya ha sido incorporado a la colección de jugadores por el *hilo de extracción de información estática de jugadores*.
- **Colección de Jugadores.** Contiene la información completa de cada jugador. Entre sus atributos, destaca:
 - **Atributos (attributes).** Representa los distintos atributos numéricos del jugador.
 - **Colección de Precios.** Almacena el histórico de precios asociados a los jugadores en momentos temporales concretos. Los atributos más relevantes de esta colección son:

- **Id (`_id`)**. Identificador único generado automáticamente por MongoDB utilizando UUID.
- **Id del jugador (`metadata.player_id`)**. Referencia al identificador del jugador al que corresponde el precio registrado en el documento.

Un índice⁸ es una estructura de datos diseñada para optimizar el rendimiento de las consultas en una base de datos. Su función principal consiste en facilitar un acceso más rápido a la información, evitando búsquedas secuenciales completas sobre el conjunto de documentos o registros.

En MongoDB, los índices se implementan mediante la tecnología de almacenamiento WiredTiger, que utiliza una estructura basada en árboles B con ciertas optimizaciones particulares. Aunque comúnmente se describe como un árbol B o B+, WiredTiger no implementa un árbol B+ clásico en sentido estricto [46].

En MongoDB, el atributo `_id` funciona como índice predeterminado. Sin embargo, con el fin de optimizar el rendimiento de ciertas consultas, resulta recomendable definir índices adicionales sobre aquellos atributos que se utilicen con frecuencia como criterios de búsqueda. En consecuencia, se han adoptado las siguientes decisiones respecto a cada colección:

- En la colección de precios se ha definido el atributo `player_id` como índice, dado que constituye la principal referencia en la mayoría de las consultas, si no en su totalidad.
- En la colección de jugadores, no existe un atributo claramente predominante que justifique la creación de un índice adicional. Considerando que el atributo `_id` ya incorpora el nombre del jugador, este resulta suficiente para las búsquedas basadas en dicho criterio, manteniéndose únicamente el índice predeterminado.
- Finalmente, en la colección de escaneo de jugadores, cuyo único consumidor es el componente de scraping, se ha decidido indexar los atributos `overall` (valoración media o global), `fecha de última actualización de precio` y `insertado`, con el propósito de facilitar un acceso eficiente a los registros relevantes durante el proceso de extracción de datos.

⁸<https://www.mongodb.com/docs/manual/indexes/>

4.6. Diseño del MGP

En esta sección se detallará el diseño del MGP. Se comenzará describiendo el servidor web, abordando sus características técnicas y funcionamiento lógico. A continuación, se presentará el diseño del back-end, incluyendo la definición y estructura de los distintos endpoints expuestos. Finalmente, se explicará el diseño e implementación del servidor intermediario encargado de establecer la conexión entre este módulo y el MRT.

4.6.1. Servidor Web

El servidor web es el componente responsable de proporcionar al usuario la interfaz y los contenidos necesarios para la interacción con el sistema. En aquellos casos en los que sea preciso, este módulo se encarga de solicitar dichos contenidos al servidor *Back-end*.

El framework seleccionado para el desarrollo del servidor web es Angular⁹, una plataforma de desarrollo de aplicaciones web de código abierto mantenida por Google.

Angular ha sido elegido por su idoneidad para la creación de aplicaciones de una sola página (SPA), permitiendo una experiencia de usuario más fluida al evitar recargas completas de la interfaz durante la navegación. Angular emplea *TypeScript*¹⁰ como lenguaje principal, una extensión tipada de JavaScript que mejora la robustez del código al permitir la detección temprana de errores durante la compilación, facilitando así el desarrollo de aplicaciones escalables y mantenibles. Además, integra un mecanismo de enlace de datos bidireccional, que asegura la sincronización automática entre el estado de los componentes y su representación visual, simplificando la gestión de la interfaz de usuario y reduciendo la necesidad de manipulaciones manuales del DOM.

Las principales entidades que conforman un proyecto desarrollado con Angular son las siguientes (ver Figura 4.13):

- **Módulos.** Constituyen la unidad organizativa fundamental de una aplicación Angular. Cada módulo agrupa componentes, directivas, servicios y otros módulos relacionados que forman una unidad funcional coherente. El módulo principal de cualquier aplicación es `AppModule`, definido en el archivo `app.module.ts`.

⁹<https://angular.dev/>

¹⁰<https://www.typescriptlang.org/>

- **Componentes.** Representan bloques funcionales de la interfaz de usuario. Cada componente se compone, habitualmente, de los siguientes elementos:
 - Una clase escrita en TypeScript, que implementa la lógica del componente.
 - Una plantilla en HTML, que define su estructura visual.
 - Una hoja de estilos en CSS o SCSS, que determina su apariencia.
- **Plantillas.** Definen la estructura visual del componente mediante HTML enriquecido con directivas y expresiones específicas del propio framework Angular.
- **Servicios.** Encapsulan la lógica de negocio o funcionalidades reutilizables (por ejemplo, realización de peticiones HTTP o manipulación de datos). Estos servicios se inyectan en componentes u otros servicios a través del sistema de inyección de dependencias que proporciona Angular.
- **Rutas.** Permiten gestionar la navegación entre diferentes vistas de la aplicación. Se configuran utilizando el módulo `RouterModule` y definen qué componente debe renderizarse para cada URL asociada.
- **Pipes.** Proporcionan mecanismos para transformar o formatear datos antes de ser presentados en la interfaz. Son útiles, por ejemplo, para el formateo de fechas, monedas o para implementar filtros personalizados.

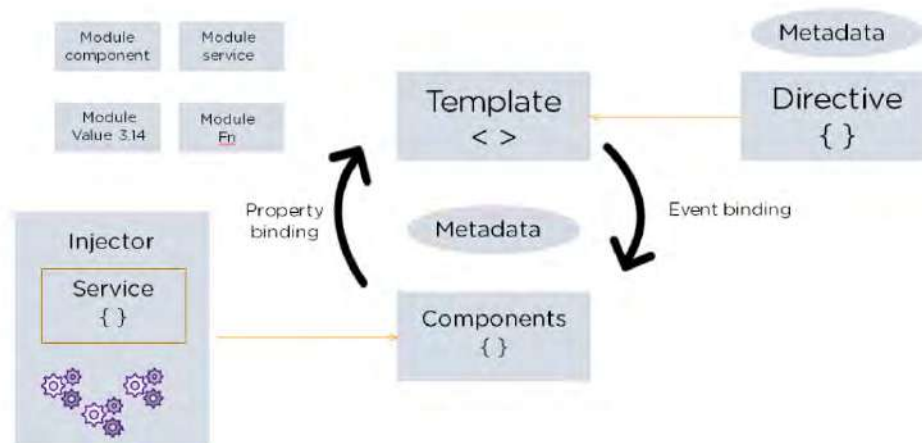


Figura 4.13: Arquitectura de Angular

Desde un punto de vista estrictamente lógico, las entidades principales que conforman el servidor web son las siguientes (ver Figura 4.14):

- **Componente de creación de plantillas.** Este componente define la interfaz destinada a la creación de plantillas y almacena toda la información relevante

relacionada con el proceso, así como con la plantilla que el usuario está configurando (jugadores seleccionados, preferencias establecidas, formación escogida, entre otros). Además, proporciona el contexto necesario al asistente mediante el servicio de comunicación.

- **Componente de ventana modal para selección de jugadores por posición.** Este componente se encarga de la representación gráfica de la ventana modal utilizada para configurar una posición específica dentro de la formación. Establece comunicación con el componente de creación de plantillas mediante herencia, permitiendo modificar únicamente los datos asociados a la posición en cuestión dentro de la estructura de datos que representa la plantilla, por ejemplo, actualizando un único elemento en un vector donde cada posición corresponde a un puesto en la formación.

- **Componente del asistente.** Este facilita la interacción del usuario con el asistente. Este componente almacena el historial de conversación y, a través del servicio de comunicación, solicita y recibe el contexto virtual del usuario, que incluye:
 - La formación actual.

 - Preferencias asociadas o jugador seleccionado por posición.

- **Servicio de comunicación.** servicio encargado de gestionar de forma centralizada las comunicaciones, tanto con elementos externos como entre los distintos componentes del servidor web, concretamente cuando dichas entidades no mantienen relaciones de herencia directa.

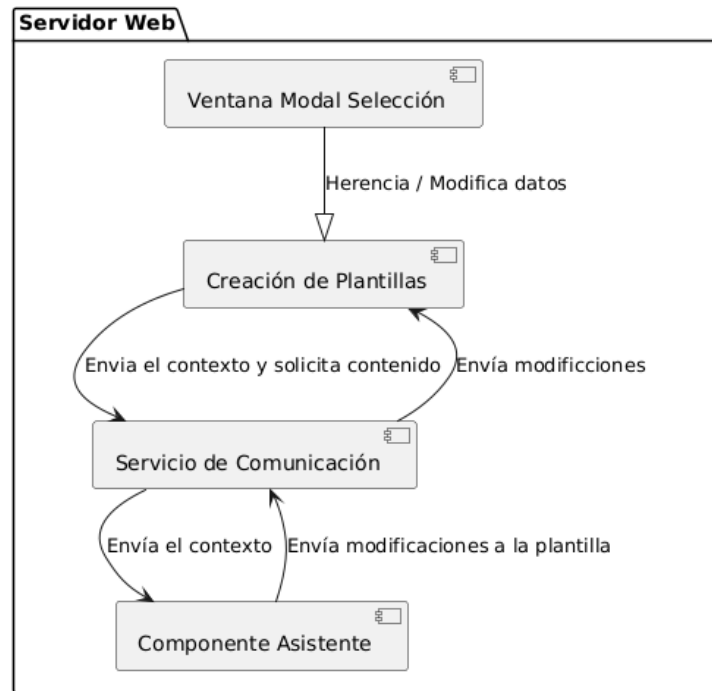


Figura 4.14: Diagrama del Funcionamiento Lógico del Servidor Web

El sistema está diseñado para que el servidor web gestione el almacenamiento de los siguientes recursos:

- Imágenes de todos los jugadores presentes en el juego.
- Imágenes de todos los equipos presentes en el juego.
- Imágenes de todas las ligas presentes en el juego.
- Imágenes de todas las nacionalidades presentes en el juego.
- Imágenes de todos los estilos de juego presentes en el juego.
- Imágenes estáticas utilizadas en el sitio web.

En el prototipo en desarrollo, todas las imágenes se almacenan localmente en el servidor web. Esta solución resulta adecuada y eficiente en esta etapa, dado que el volumen de recursos es limitado y las condiciones de uso se encuentran bajo control.

Sin embargo, esta arquitectura presenta limitaciones en términos de escalabilidad, por lo que no es recomendable para un entorno de producción, donde se prevé un incremento en la demanda y la concurrencia de usuarios.

Para un despliegue a gran escala, se aconseja externalizar el almacenamiento y la distribución de recursos estáticos mediante servicios especializados, tales como sistemas de almacenamiento de objetos como Amazon S3¹¹, Google Cloud Storage¹², o CDN como Bunny CDN¹³, Fastly¹⁴. Esta estrategia proporciona múltiples beneficios:

- Reducción de la carga en el servidor principal.
- Mejora en los tiempos de respuesta y en la experiencia del usuario final, gracias a la distribución geográfica y almacenamiento en caché de los contenidos.
- Mayor flexibilidad para escalar el sistema conforme crezca el volumen de recursos y usuarios.

Contemplando el fin de cumplir el requisito de facilitar la internacionalización del sitio web y asegurar su mantenimiento a lo largo del tiempo, se han adoptado las siguientes medidas:

- **La estructura de los menús desplegados extensos se almacenará en archivos JSON fácilmente modificables.** Esto permite que la incorporación de nuevos elementos, como podría ser la inclusión de un nuevo juego, se realice con un nivel mínimo de complejidad, limitándose únicamente a la edición del archivo JSON correspondiente.
- **Las etiquetas utilizadas en la interfaz se gestionarán mediante un diccionario multilingüe.** Este diccionario posibilita el almacenamiento de etiquetas en distintos idiomas y facilita, asimismo, la modificación sencilla de la terminología empleada en el sitio web.

4.6.2. Servidor Back-end

El servidor back-end será el componente encargado de centralizar las comunicaciones entre todos los módulos del sistema y el servidor web, proporcionando respuestas que incluyan los contenidos solicitados por el usuario a través de la interfaz. En esencia, sus funciones principales serán:

¹¹<https://aws.amazon.com/es/pm/serv-s3/>

¹²<https://cloud.google.com/products/storage>

¹³<https://bunny.net/lp-cdn/>

¹⁴<https://www.fastly.com/es/services/managed-cdn>

- Suministrar contenido al *Servidor Web*.
- Establecer la conexión entre los distintos servidores y el *Servidor Web*.
- Gestionar la comunicación entre la *Base de Datos Distribuida* y el *Servidor Web*.
- Administrar la autenticación y autorización de los usuarios.

NestJS¹⁵ (ver Figura 4.15) es un framework progresivo para el desarrollo de aplicaciones back-end con Node.js. Está construido sobre el motor de ejecución de JavaScript de Node.js y utiliza TypeScript como lenguaje principal, lo que permite contar con tipado estático y una arquitectura orientada a objetos que facilita el desarrollo escalable y mantenible.

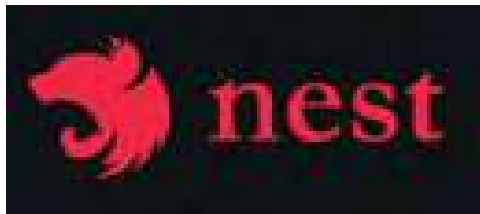


Figura 4.15: Logo de NestJs

Entre sus principales ventajas se encuentran la compatibilidad con TypeScript¹⁶, la cual incrementa la productividad y reduce la probabilidad de errores gracias a un sistema de tipado estático y fuerte. Además, cuenta con una arquitectura basada en controladores y servicios que facilita la organización modular del código y promueve buenas prácticas de diseño de software. Dispone de un sistema de inyección de dependencias que permite el desacoplamiento eficiente de componentes, favoreciendo la mantenibilidad y la testabilidad del sistema. También ofrece integración nativa con bibliotecas modernas como TypeORM¹⁷, Mongoose¹⁸, GraphQL¹⁹ y web sockets²⁰, entre otras, así como un soporte integral para pruebas que incluye herramientas integradas para la ejecución sencilla y efectiva de pruebas unitarias y de integración.

NestJS estructura su arquitectura mediante un conjunto de entidades claramente definidas, cada una con un propósito específico dentro de la aplicación. A continuación, se describen las más relevantes:

¹⁵<https://nestjs.com/>

¹⁶<https://www.typescriptlang.org/>

¹⁷<https://typeorm.io/>

¹⁸<https://mongoosejs.com/>

¹⁹<https://graphql.org/>

²⁰<https://docs.nestjs.com/websockets/gateways>

- **Módulos.** Constituyen la unidad organizativa fundamental en NestJS. Cada módulo agrupa componentes relacionados, tales como controladores, servicios y proveedores, facilitando así la modularidad y escalabilidad del sistema.
- **Controladores.** Son responsables de gestionar las solicitudes entrantes (HTTP, WebSocket, entre otras) y de proporcionar las respuestas correspondientes al cliente.
- **Servicios.** Contienen la lógica de negocio de la aplicación y actúan como capa intermedia entre los controladores y la capa de acceso a datos.
- **Proveedores.** Representan cualquier clase que pueda ser inyectada mediante el sistema de inyección de dependencias de NestJS. Incluyen servicios, repositorios, adaptadores y otros componentes reutilizables.
- **Middlewares.** Son funciones que se ejecutan antes de que una solicitud alcance el controlador, empleadas para tareas como autenticación, registro de eventos (logging) o transformación de datos.
- **Interceptores.** Permiten intervenir en el flujo de ejecución antes o después de la invocación de un método del controlador, siendo útiles para transformar respuestas, ampliar funcionalidades o gestionar errores de forma centralizada.
- **Pipes.** Encargados de transformar y validar los datos de entrada antes de que estos sean procesados por el controlador.
- **Guards.** Implementan la lógica de autorización, determinando si una solicitud tiene permisos para acceder a un recurso específico.

El servidor back-end está compuesto por los siguientes controladores principales (ver Figura 4.16):

- **Controlador de Jugadores.** Encargado de gestionar todas las solicitudes relacionadas con consultas a la base de datos de jugadores.
- **Controlador de Creación de Plantillas.** Responsable de manejar las solicitudes dirigidas al módulo de ejecución de tareas, incluyendo tanto la creación automática de plantillas como la ejecución de modelos de toma de decisión para la selección de jugadores en posiciones específicas dentro de la plantilla.
- **Controlador RAG.** Administra las consultas dirigidas al módulo del asistente basado en Retrieval-Augmented Generation (RAG).



Figura 4.16: API que proporciona el Back-end

4.6.3. Servidor Redis

El servidor Redis actúa como intermediario entre el servidor back-end y los servidores workers y servidores RAG. Para ello, mantiene dos colas de trabajo diferenciadas: una destinada a las tareas gestionadas por los servidores workers y otra para las consultas dirigidas a los servidores RAG.

Redis (Remote Dictionary Server) es un sistema de almacenamiento en memoria de código abierto basado en un modelo clave-valor. Se distingue por su elevado rendimiento y gran versatilidad, presentando las siguientes características principales:

- **Almacenamiento en memoria.** Redis gestiona todos los datos directamente en la memoria RAM, lo que garantiza tiempos de lectura y escritura extremadamente bajos.
- **Estructuras de datos avanzadas.** Ofrece soporte para una amplia variedad de estructuras, tales como listas, hashes, conjuntos y conjuntos ordenados, entre otras.
- **Persistencia opcional.** Aunque trabaja principalmente en memoria, permite la persistencia de datos en disco mediante snapshots o mediante el registro de operaciones (Append-Only File, AOF).
- **Escalabilidad.** Soporta replicación, particionado mediante clústeres y mecanismos de alta disponibilidad para entornos críticos.

■ **Casos de uso habituales:**

- **Caché de aplicaciones.** Almacenamiento temporal de datos frecuentemente accedidos para reducir la carga sobre las bases de datos principales y mejorar el rendimiento general.
- **Gestión de sesiones.** Almacenamiento temporal de sesiones de usuario en aplicaciones web.
- **Colas de mensajes.** Ideal para sistemas de procesamiento asíncrono, tal como se emplea en este proyecto.
- **Procesamiento en tiempo real y datos geoespaciales.** Facilita operaciones que requieren acceso rápido y eficiente a los datos.

Biblioteca seleccionada para la implementación de colas

Para la gestión e implementación de las colas en el servidor Redis, se ha optado por utilizar la biblioteca BullMQ²¹, debido a su robustez, eficiencia y compatibilidad con entornos Node.js.

4.7. Diseño del MRT

Este módulo se encarga de procesar tareas computacionalmente costosas de forma asíncrona, permitiendo que:

- El servidor web y el servidor back-end queden liberados de la ejecución de operaciones complejas, tales como la ejecución de modelos de toma de decisión o algoritmos de optimización.
- Al contar con sistemas especializados y dedicados exclusivamente a la ejecución de estas tareas, se minimiza el tiempo de respuesta del sistema, aspecto fundamental en entornos web y especialmente relevante para la experiencia del usuario objetivo.

²¹<https://docs.bullmq.io/>

4.7.1. Servidores Worker

Los servidores worker son los componentes responsables de monitorear de forma continua la incorporación de nuevas tareas en las colas gestionadas por el *servidor Redis*, para proceder a su procesamiento. Su función principal es ejecutar las tareas encoladas y devolver las respuestas correspondientes. Entre las responsabilidades fundamentales de estos servidores destacan:

- La generación automática de plantillas basadas en prioridades predefinidas, lo cual implica la ejecución de un algoritmo de optimización multiobjetivo.
- La resolución de problemas de toma de decisión individualizados para cada posición dentro de la formación, mediante la ejecución de modelos específicos de toma de decisión.

La implementación de los servidores worker se realizará utilizando el framework NestJS, al igual que en el servidor back-end. Para la ejecución de los algoritmos asociados, se empleará la biblioteca `child_process`, que permite la creación y gestión de procesos hijos. A dichos procesos se les suministrarán, en el momento de su inicialización, los parámetros correspondientes, tales como prioridades, filtros y presupuesto disponible.

El modelo de toma de decisión propuesto es **TOPSIS** (modelo que se explica detenidamente en la Sección [2.2.3](#)), con un método de asignación directa de pesos por parte del usuario. El procedimiento general se estructura en las siguientes etapas:

1. En primer lugar, el usuario asigna una puntuación numérica a cada uno de los criterios que considere relevantes, dentro de un rango comprendido entre 0 y 100.
2. Posteriormente, estas puntuaciones se normalizan de forma que su suma sea igual a uno.
3. Se aplica una normalización min-max a los vectores de características de todos los jugadores (alternativas).
4. A continuación, se evalúan las alternativas mediante el algoritmo TOPSIS.
5. Finalmente, se genera y presenta al usuario un ranking con los resultados obtenidos.

Este modelo ha sido seleccionado con el objetivo de ofrecer un sistema accesible y comprensible para usuarios jóvenes o aficionados al fútbol, quienes no necesariamente poseen conocimientos técnicos sobre matemáticas o técnicas de toma de decisiones.

Para ilustrar las dificultades que pueden surgir al aplicar el método TOPSIS a variables categóricas codificadas mediante one-hot, se presenta el siguiente ejemplo. La matriz de decisión incluye los siguientes criterios:

- C_1 : Altura (criterio de beneficio, variable continua)
- C_2 : Nacionalidad española (criterio de beneficio, variable categórica)

Las alternativas consideradas corresponden a dos jugadores:

Jugador	Altura (C_1)	Español (C_2)
Emiliano	189	0
Kepa	186	1

El proceso de resolución se organiza en los siguientes pasos:

- **Vector de prioridades.** Los pesos asignados a cada criterio, en función de su importancia relativa, se muestran en la Ecuación 4.3:

$$\mathbf{w} = (0,9, 0,1) \quad (4.3)$$

- **Determinación de las soluciones ideal y anti-ideal.** Se consideran soluciones ideales absolutas para ambos criterios, como se indica en la Ecuación 4.4:

$$A^+ = (1, 1), \quad A^- = (0, 0) \quad (4.4)$$

- **Cálculo de los coeficientes de similitud.** Los coeficientes de similitud obtenidos para cada alternativa, utilizando dos métodos de normalización, se recogen en la Tabla 4.7.1:

A partir de múltiples experimentaciones, se ha comprobado empíricamente que la combinación de normalización min-max con soluciones ideales absolutas proporciona

Jugador	Normalización Min-Max	Normalización Distributiva
Emiliano	0.47	0.3005
Kepa	0.44	0.3068

Tabla. 4.1: Coeficientes de similitud por normalización

rankings coherentes e intuitivos, incluso al trabajar con criterios nominales codificados mediante one-hot.

La posible pérdida de información asociada al uso de normalización min-max es irrelevante en el contexto de este sistema, ya que no se trata de un entorno crítico. De hecho, una ligera imprecisión puede ser incluso beneficiosa, al fomentar una mayor interacción por parte del usuario, quien tenderá a consultar con más frecuencia distintas configuraciones en busca de alternativas óptimas.

Para abordar el problema de la creación automática de plantillas, se ha seleccionado un algoritmo ACOS [1] (este se explica de manera extensa en la Sección 2.3.2), el cual ha sido adaptado y personalizado con el objetivo de incrementar la capacidad de exploración y explotación del espacio de soluciones mediante la implementación de una búsqueda local para refinar las soluciones generadas.

Como en toda metaheurística, es necesario definir previamente los elementos clave que estructuran el problema:

- **Espacio de soluciones.** Cada solución se representa mediante un vector de 11 elementos, donde cada posición corresponde a un jugador específico, conformando así una plantilla completa.
- **Función objetivo.** La evaluación de cada solución se realiza calculando la media aritmética de los siguientes componentes:
 - La química total de la plantilla.
 - La media de los valores obtenidos por cada jugador al aplicar el método TOPSIS individualmente a su posición dentro de la formación.
- **Restricciones.** El proceso de optimización debe cumplir las siguientes condiciones:
 - Un límite máximo de precio para la plantilla.
 - La aplicación de filtros específicos sobre los jugadores disponibles para cada posición.

4.7.2. Base de Datos Local Reducida

Cada servidor worker mantiene una base de datos local de tamaño reducido en formato JSON. El propósito de esta base de datos no es resolver consultas específicas, sino proporcionar de forma íntegra la información actualizada y necesaria sobre todos los jugadores del juego que puedan ocupar las posiciones implicadas en el problema de optimización.

El mantenimiento de estas bases de datos locales reducidas ofrece las siguientes ventajas:

- **Reducción del tiempo de respuesta.** El acceso a los datos es inmediato al estar disponibles de forma local en cada servidor.
- **Descongestión de la base de datos distribuida.** La actualización de los datos locales puede programarse de forma periódica, por ejemplo, una vez al día o con la frecuencia que se estime oportuna, preferiblemente en franjas horarias de baja demanda, minimizando así el impacto sobre el sistema principal.

La base de datos local reducida se organiza en varios archivos JSON que conservan una estructura similar a la utilizada en la base de datos documental principal, proporcionando los mismos datos pero con ciertas modificaciones en su formato para optimizar su uso en el proceso de optimización:

- **Incorporación del precio.** El valor almacenado para el precio corresponde a la mediana de los precios del jugador registrados durante los siete días previos. En caso de que el jugador no haya sido adquirido en dicho periodo, se recurre a la mediana calculada a partir del histórico completo de precios. La elección de la mediana responde a su carácter robusto frente a valores extremos, lo que permite obtener una estimación más representativa y fiable del precio real del jugador.
- **Atributos nominales como claves del objeto.** Los atributos nominales, como el rol o el estilo de juego, se representan directamente como claves del objeto correspondiente al jugador. Esta estructura permite consultas eficientes, ya que basta con verificar la existencia de una clave específica en el diccionario para conocer si un jugador posee un determinado atributo.
- **Un archivo JSON por posición.** Dado que cada problema de toma de decisiones se resuelve de forma independiente para una posición concreta, se ha

optado por organizar los jugadores en archivos separados por posición. Esta segmentación evita búsquedas innecesarias y reduce el espacio de búsqueda para cada ejecución del algoritmo de toma de decisión.

El proceso de construcción de la base de datos local reducida puede programarse para su ejecución durante los periodos de menor tráfico del sistema, con el fin de evitar la sobrecarga de la base de datos principal. La construcción de la base de datos local se realiza de forma concurrente y sigue las siguientes etapas para cada una de las posiciones:

1. En primer lugar, el hilo principal obtiene el conjunto completo de jugadores asociados a una posición concreta.
2. A continuación, dicho conjunto se divide en varios subconjuntos, asignando uno a cada hilo disponible para su procesamiento concurrente.
3. Cada hilo consulta el histórico de precios correspondiente a los jugadores de su subconjunto, aplicando las reglas definidas previamente para el cálculo de la mediana.
4. Finalmente, las características de los jugadores de cada posición, junto con sus respectivos precios calculados, se almacenan en un archivo JSON.

4.8. Diseño del MA

El MA tiene como objetivo principal proporcionar al usuario el conocimiento experto necesario durante el proceso de toma de decisiones. Este módulo está diseñado para complementar las capacidades del sistema, ofreciendo asistencia especializada en aquellos casos en los que el usuario requiera apoyo adicional. Su estructura se compone de los siguientes elementos:

4.8.1. Servidores RAG

Los servidores RAG son los encargados de atender las consultas publicadas en la cola gestionada por el servidor Redis, donde se envían las solicitudes destinadas al asistente o chatbot. Su función principal es generar respuestas fundamentadas en:

- Los documentos del corpus, que deben ser recuperados desde el servidor de almacén de documentos.
- Las consultas formuladas por los LLMs encargados del procesamiento semántico.

Si bien en una implementación completa los servidores RAG deberían consultar de forma explícita los documentos relevantes al Servidor de Almacén de Documentos, en esta versión prototipo se ha optado por integrar dichas funcionalidades directamente en los propios servidores RAG. Esta decisión responde al propósito de simplificar la arquitectura durante las fases de desarrollo y prueba.

A diferencia de un sistema RAG convencional, los servidores RAG implementan un comportamiento alineado con los principios de los sistemas RAG agénticos (véase la Sección 2.4.4), con el fin de abordar tareas más complejas que requieren capacidades avanzadas de razonamiento y generación. Entre estas tareas se incluyen:

- **Creación automática de plantillas a partir de un prompt.** Esta funcionalidad permite la generación completa de plantillas mediante la definición automática de prioridades. En versiones anteriores, el algoritmo basado en colonias de hormigas generaba plantillas de forma automática, pero requería que el usuario configurase manualmente las prioridades para cada posición. La nueva implementación incorpora un mecanismo que automatiza también la asignación de dichas prioridades, mejorando la experiencia del usuario. Esta definición automática es realizada por los Servidores RAG a partir de:
 - Las prioridades explícitas definidas por el usuario en el prompt.
 - El conocimiento experto contenido en el corpus documental.
 - Escala nominal definida en el *manual* del asistente, que establece que el asistente debe describir la importancia de los atributos utilizando las siguientes categorías:
 - Muy importante.
 - Alta importancia.
 - Importancia media.
 - Baja importancia.
 - Sin importancia.
- **Evaluación de plantillas en función de la información del corpus.** Además de generar plantillas automáticamente en función de las prioridades indicadas por el

usuario mediante interacción semántica, el sistema es capaz de analizar dichas plantillas y proporcionar recomendaciones tácticas basadas en el conocimiento disponible en el corpus. Este proceso dota al sistema de una capacidad explicativa adicional, ofreciendo al usuario sugerencias personalizadas, estrategias de juego y justificaciones fundamentadas sobre las decisiones propuestas.

El Agentic RAG puede clasificarse dentro del paradigma de arquitectura de agente único, este se describe en la Sección 2.4.4. Aunque físicamente se dispone de un único agente hardware, este puede comportarse como múltiples agentes lógicos, que varían según el contexto y cada uno se ejecuta en base a un conjunto específico de instrucciones.

El diseño y la arquitectura del sistema se desarrollaron considerando la problemática de que los modelos de lenguaje no han sido entrenados en un dominio específico, en este caso, el Ultimate Team. Por lo tanto, resulta imprescindible guiar al agente durante su proceso de planificación. Para afrontar esta dificultad, se implementan dos tipos de planificación:

- **Planificación estática.** Este tipo de planificación depende de la funcionalidad solicitada y permanece constante. Define los pasos generales que deben seguirse para resolver una tarea y está incorporada directamente en el código. Aunque los pasos globales no pueden modificarse, el sistema adapta cada uno de ellos a las solicitudes específicas del usuario, es decir, el conocimiento recuperado en cada paso se ajusta según las preferencias o inquietudes planteadas en la consulta principal.
- **Planificación dinámica.** Esta planificación corresponde a las distintas consultas que el agente realiza de forma autónoma para resolver una query concreta. Se orienta por un modelo de razonamiento diseñado para compensar el desconocimiento del dominio por parte de los modelos de lenguaje.

La arquitectura está constituida por los siguientes agentes y componentes (ver Figura 4.17):

- **Manual.** Es una estructura de datos estática que proporciona información sobre el procedimiento a seguir ante distintas situaciones o funcionalidades solicitadas. Para cada funcionalidad se almacenan los siguientes elementos:
 - **Nombre.** Designa la funcionalidad. Se contemplan tres funcionalidades principales:

- Evaluar plantilla.
 - Comparar jugadores.
 - Crear un equipo.
 - **Tipo.** Indica si la funcionalidad corresponde a una consulta o a una tarea.
 - **Preguntas de ejemplo.** Ejemplos de preguntas que facilitan la identificación de la funcionalidad solicitada.
 - **Modelo de razonamiento.** Prompt que especifica el modo de pensar, razonar y actuar en relación con dicha funcionalidad.
 - **Instrucciones.** Pasos o aspectos que deben resolverse previamente a la consulta del usuario, con el fin de proporcionar un contexto adecuado para la consulta principal. Estas instrucciones no son prompts, sino que son otras instrucciones del manual que se deben explorar antes de ejecutar la consulta principal. Por ejemplo, antes de completar un equipo deberemos evaluar la plantilla para que la selección de los jugadores restantes sea acorde a la táctica que emplea el usuario.
- **Agente orquestador.** Responsable de recibir la consulta y determinar la intención del usuario. Esta identificación se realiza tomando como referencia el manual.

Una vez determinada la acción a ejecutar, asigna tareas a los diferentes agentes. Asimismo, si se trata de una tarea específica, sigue una secuencia de comportamiento predefinida (planificación estática).

- **Agente planificador.** Este agente recibe la siguiente información de entrada:
- **Consulta formulada por el usuario.**
 - **Modelo de razonamiento asociado a la instrucción:** prompt diseñado específicamente para una instrucción determinada, que establece el procedimiento a seguir ante dicha instrucción. Este prompt varía en función de la instrucción identificada, pero se mantiene invariable para cada instrucción concreta. Su finalidad es reducir la imprecisión del modelo, dado que este no ha sido entrenado específicamente para el propósito particular al que se destina.
 - **Contexto inicial recuperado:** conjunto de datos preliminares obtenidos mediante una consulta directa al sistema de recuperación de información, cuya finalidad es paliar la posible falta de conocimiento específico sobre el dominio tratado.

A partir de esta información, el agente planificador genera las consultas necesarias para atender adecuadamente la solicitud del usuario, gestionando de este modo la planificación dinámica del proceso.

- **Agente recuperador.** Encargado de recuperar documentos relevantes para todas las consultas y de resumirlos en función de:
 - El modelo de razonamiento.
 - La consulta del usuario.

- **Agente validador.** Responsable de verificar que la respuesta generada por el sistema sea completa y satisfaga adecuadamente la consulta del usuario.

Esta propuesta de arquitectura contribuye a reducir la variabilidad de los resultados, mejorando así la precisión y la consistencia de las respuestas proporcionadas por el asistente. No obstante, esta aproximación limita en cierta medida la creatividad del agente. Esta solución puede resultar especialmente adecuada para aquellas empresas que requieran un servicio de asistencia con las siguientes características:

- Que se base en la ejecución de un conjunto concreto de tareas recurrentes, solicitadas mayoritariamente por los usuarios, y cuya resolución cuente con procedimientos bien definidos.

- Que se trate de pequeñas empresas con recursos limitados, que no puedan asumir los costes asociados al uso de modelos de lenguaje de mayor capacidad o sofisticación, ni a su reentrenamiento para dominios altamente especializados.

- Que las tareas a realizar exijan un alto grado de precisión.

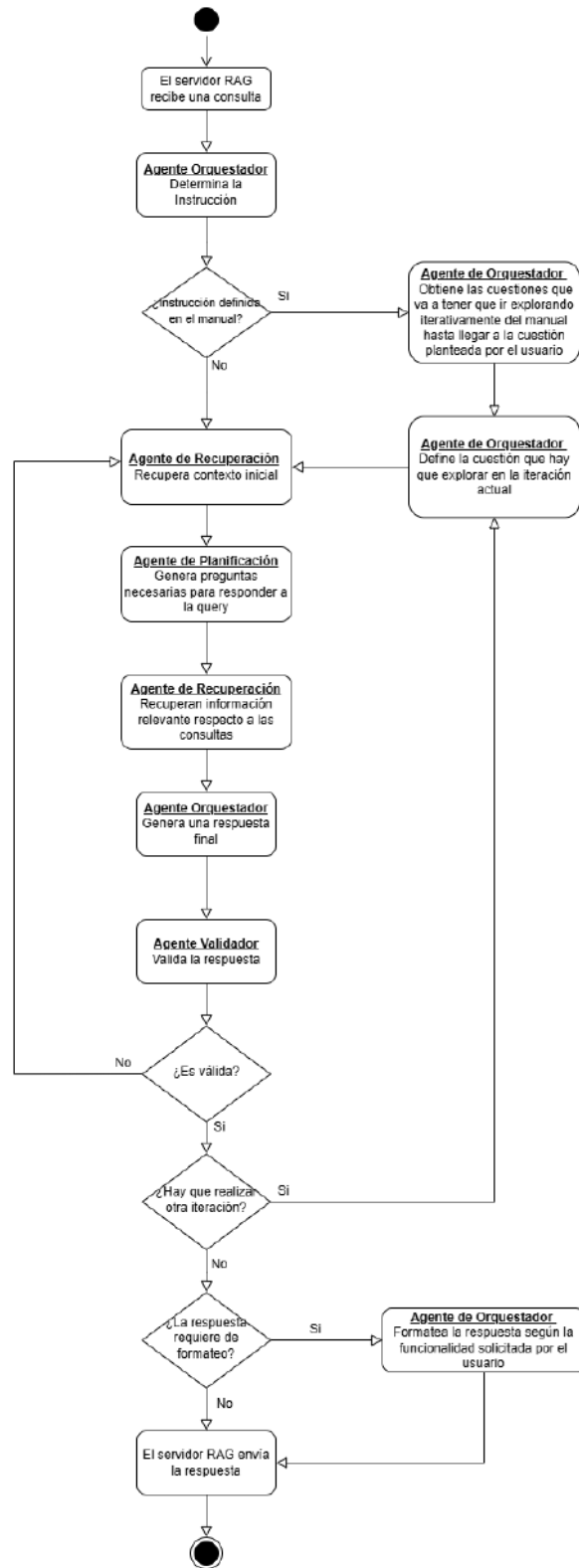


Figura 4.17: Funcionamiento de los Servidores RAGs

4.8.2. Servidor de Almacenamiento de Documentos

Este componente consiste en un base de datos vectorial, en la que se almacena un embedding por cada documento del corpus. La características teóricas de este componente se pueden consultar en la Sección [2.4.3](#).

Para su integración en el sistema, existen diversas tecnologías especializadas en el almacenamiento y la búsqueda eficiente de vectores de alta dimensión. Entre las soluciones más destacadas se encuentran:

- **FAISS (Facebook AI Similarity Search)**. Biblioteca desarrollada por Meta, altamente optimizada para la búsqueda de similitud en grandes volúmenes de vectores. Soporta indexación en CPU y GPU, y resulta especialmente adecuada para escenarios que requieren búsquedas en tiempo real sobre conjuntos de datos de gran tamaño.
- **Chroma**. Alternativa ligera orientada a entornos de desarrollo local o prototipado. Ofrece una integración sencilla con aplicaciones desarrolladas en Python y bibliotecas como LangChain.

En el prototipo desarrollado, la base de datos vectorial ha sido reemplazada por una estructura simplificada que consiste en una lista de embeddings generados mediante un modelo *transformer* ejecutado localmente. Estos embeddings se almacenan en memoria como una lista, lo que permite realizar búsquedas de similitud de forma básica empleando la distancia coseno como métrica de referencia.

Aunque esta solución no es escalable ni está optimizada para entornos con grandes volúmenes de información, resulta adecuada en el contexto de un entorno de pruebas o de validación preliminar del sistema.

Modelo de Transformers utilizado para generar los embeddings

Para la generación de embeddings en este prototipo se ha utilizado el modelo `intfloat/multilingual-e5-large` [47], desarrollado por Microsoft. Este modelo se basa en la arquitectura *XLM-RoBERTa-large* y ha sido entrenado específicamente para generar representaciones vectoriales de alta calidad, optimizadas para tareas de recuperación de información y búsqueda semántica en entornos multilingües.

Principales características del modelo:

- **Arquitectura.** 24 capas y aproximadamente 560 millones de parámetros.
- **Dimensión de los embeddings.** 1024 dimensiones.
- **Cobertura lingüística.** Más de 100 idiomas, incluyendo español, inglés, árabe y chino, entre otros.
- **Límite de tokens por entrada.** hasta 512 tokens.

La integración del modelo se ha realizado mediante el uso de las bibliotecas Transformers de Hugging Face²² y PyTorch.

4.8.3. Modelo de Lenguaje Natural

Este componente está constituido por un conjunto de servidores que ejecutan modelos de lenguaje natural, a los cuales se les puede enviar consultas.

OpenRouter²³ es una plataforma que proporciona una API unificada para el acceso a más de 300 modelos de lenguaje de grandes proveedores, entre los que se incluyen OpenAI, Anthropic, Google, Meta y Mistral, entre otros.

Las principales características que ofrece esta API son:

- **Acceso unificado.** Permite interactuar con diversos modelos de lenguaje mediante una única API, simplificando los procesos de integración y reduciendo la complejidad del desarrollo.
- **Compatibilidad con SDKs.** Es compatible con el SDK de OpenAI y admite múltiples lenguajes de programación, como Python y TypeScript, facilitando así su integración en distintos entornos de desarrollo.
- **Soporte multimodal.** Permite el envío de imágenes y archivos PDF a través de la API, habilitando el uso de modelos capaces de procesar tanto entradas textuales como visuales.
- **Disponibilidad de modelos gratuitos.** Ofrece acceso a una selección de modelos de uso gratuito, lo que resulta especialmente útil en contextos de experimentación o desarrollo preliminar.

²²<https://huggingface.co/>

²³<https://openrouter.ai/>

4.8.4. Colección de Documentos

Esta colección está compuesta por documentos no estructurados que han sido recopilados de diversas fuentes relevantes, tales como:

- Transcripciones de vídeos de creadores de contenido (youtubers) y expertos (jugadores profesionales) que abordan temas de interés relacionados con el juego.
- Guías públicas disponibles en línea que ofrecen información sobre el juego.
- Noticias y artículos obtenidos de sitios web especializados en la temática, como por ejemplo FUT.GG.

Proceso de Elaboración de la Colección de Documentos

El proceso seguido para la elaboración de la colección de documentos se compone de las siguientes etapas:

1. En primer lugar, se obtiene el contenido de cada archivo que se desea incorporar a la colección y se convierte a un formato estándar adecuado para el almacenamiento de archivos de texto. Para ello, se emplean los siguientes procedimientos:
 - Si el archivo se encuentra en formato de audio, se utiliza el modelo *Whisper*²⁴ (versión base) para transcribir su contenido a texto.
 - Si el archivo es un documento en formato PDF, se emplea la biblioteca *PyMuPDF* para extraer el texto completo de cada página.
2. A continuación, el texto se divide en fragmentos o *chunks* de tamaño fijo, permitiendo un solapamiento parcial entre ellos con el objetivo de preservar el contexto semántico.
3. En aquellos casos en los que los *chunks* hayan sido generados a partir de transcripciones de audio, cada fragmento es posteriormente procesado mediante un modelo de lenguaje con el fin de generar un resumen detallado, estructurado y claro. El *prompt* empleado en este paso ha sido diseñado específicamente para mejorar la calidad de las transcripciones y extraer información relevante en el contexto del videojuego FC 25. El *prompt* utilizado es el siguiente:

²⁴<https://openai.com/es-ES/index/whisper/>

Por favor, genera un resumen detallado y extenso en español del siguiente texto, centrado en información relevante para un jugador de FC 25. Comienza el resumen introduciendo la temática central de forma clara y directa. Incluye siempre los nombres específicos de las formaciones y de los jugadores mencionados. Destaca atributos, estadísticas, estrategias, tácticas, roles y estilos de juego. Corrige cualquier error ortográfico o de expresión que pueda haber debido a una mala transcripción del audio. Procura que el resumen sea completo, bien estructurado y fácil de entender.

4.9. Diseño del Sitio Web

En esta sección se detallan todas las decisiones de diseño adoptadas, junto con su justificación. Se abordan aspectos que van desde la elección del nombre de dominio y la arquitectura de la información del sitio, hasta el diseño visual y la interacción en cada una de las páginas.

4.9.1. Nombre de Dominio

La elección de un nombre de dominio adecuado constituye un elemento fundamental en el diseño del sitio web. Para su selección, se han tenido en cuenta los siguientes criterios:

- **Brevidad y facilidad de memorización.** *FC Assistant*, compuesto por tan solo dos palabras y diez caracteres, es un nombre conciso, fácil de recordar y de escribir.
- **Sonoridad.** Se trata de un nombre sencillo de pronunciar y que transmite una imagen profesional.
- **Proyección internacional.** Las siglas *FC* hacen referencia a la nueva serie de videojuegos de fútbol de EA Sports, mientras que *Assistant* es un término en inglés de uso global, lo que favorece su comprensión y aceptación a nivel internacional. Además, la similitud fonética y gráfica entre *Assistant* y *asistente* facilita su identificación por parte de usuarios hispanohablantes con conocimientos básicos de inglés.
- **Coherencia con el contenido del sitio.** El nombre refleja de manera clara el propósito del proyecto: ser un asistente para el videojuego *FC 25*. Aunque es-

tá orientado principalmente a la modalidad *Ultimate Team*, permite también la creación de equipos para el resto de modalidades del juego.

4.9.2. Paleta de Colores

Dado que no dispongo de una formación especializada en diseño gráfico, he optado por inspirarme en otros sitios web ajenos a la temática de este proyecto, pero cuyo diseño considero visualmente atractivo y profesional.

La referencia seleccionada ha sido el sitio web de 3 Sided Cube²⁵ (ver Figura 4.18), que destaca por su diseño moderno y cuidado, así como por el uso de colores saturados y vibrantes. Estimo que la paleta cromática empleada en dicho sitio puede resultar más eficaz para captar la atención del público objetivo, especialmente si se compara con propuestas cromáticas más conservadoras como las que presentan plataformas especializadas en FIFA, como FUT.gg o Futbin.com.



Figura 4.18: Sitio web 3 Sided Cube

4.9.3. Fuentes Tipográficas

La selección de fuentes tipográficas responde a la intención de transmitir una estética lúdica, juvenil y contemporánea. El propósito es que el usuario no perciba el

²⁵<https://3sidedcube.com/>

sitio web como el portal de una entidad formal o institucional, sino como un entorno atractivo, dinámico y alineado con el universo que le apasiona.

Además, las tipografías seleccionadas guardan coherencia con aquellas utilizadas habitualmente en los vídeos y contenidos de los creadores más influyentes para nuestro público objetivo, lo que refuerza una conexión visual y emocional inmediata entre el sitio web y los referentes audiovisuales ya presentes en su día a día.

Para la barra de navegación principal se ha empleado la tipografía mostrada en la Figura 4.19.

WHEREAS DISREGARD AND CONTEMPT FOR HUMAN RIGHTS HAVE RESULTED

Figura 4.19: Fuente tipográfica: Bangers

En cuanto al resto del texto, se ha utilizado la fuente mostrada en la Figura 4.20.

Whereas disregard and contempt for human rights have resulted

Figura 4.20: Fuente tipográfica: Mouse Memoirs

4.9.4. Etiquetado

Uno de los requisitos no funcionales establecidos para el proyecto es que el etiquetado debe ser claro, sencillo e intuitivo para el usuario objetivo. Con el fin de garantizar una comprensión adecuada por parte de los usuarios, especialmente aquellos sin conocimientos previos en técnicas de toma de decisiones, se han adoptado las siguientes decisiones terminológicas:

- **Sustitución del término 'Prioridades' por 'Importancias'**. Esta modificación permite que el propósito de la sección dedicada al establecimiento de prioridades para cada jugador sea más comprensible de forma intuitiva.
- **Sustitución del término 'Criterios' por 'Características'**. Se ha optado por este cambio para emplear un término más familiar y accesible para el público general.
- **Sustitución del término 'Requisitos' por 'Filtros'**. Dado que los usuarios están más habituados al concepto de filtros en entornos digitales, se ha considerado más adecuado utilizar este término, ya que ambos representan la misma funcionalidad dentro del contexto de la aplicación.

4.9.5. Página Principal

La página principal ha sido diseñada con el objetivo fundamental de comunicar de forma clara y efectiva el propósito y las capacidades del sitio web. En la Figura 4.21, se puede el wireframe de la misma.

La estructura de la página se organiza en torno a cuatro bloques principales:

- **Naturaleza del sitio.** Presentación concisa que describe la finalidad y el enfoque del proyecto.
- **Funcionalidades.** Descripción de las principales herramientas y acciones que el usuario puede llevar a cabo dentro de la plataforma.
- **Acceso y usabilidad.** Indicaciones breves y claras que orientan al usuario sobre el uso de las funcionalidades más relevantes.
- **Validación externa.** Inclusión de testimonios y valoraciones de figuras reconocidas en el ámbito, como creadores de contenido y jugadores profesionales de FC Ultimate Team, con el fin de reforzar la credibilidad del sitio.

Con el propósito de alinearse con las tendencias actuales en el consumo de contenido digital, se contempla la incorporación de material audiovisual breve y de carácter explicativo. Esta decisión responde a la preferencia generalizada de los usuarios por formatos visuales, que facilitan una comprensión más rápida e intuitiva.

La página se completa con un pie de página en el que se incluye toda la información relativa a derechos de autor y demás aspectos legales pertinentes.

4.9.6. Página de Creación de Plantillas

Esta sección proporciona al usuario la interfaz destinada a la creación de plantillas (ver Figura 4.22). Teniendo en cuenta el perfil del usuario objetivo y el análisis comparativo con otras plataformas del sector, se han tomado las siguientes decisiones de diseño:

La eliminación del nombre de la página constituye una decisión arriesgada, adoptada en función de los siguientes criterios:

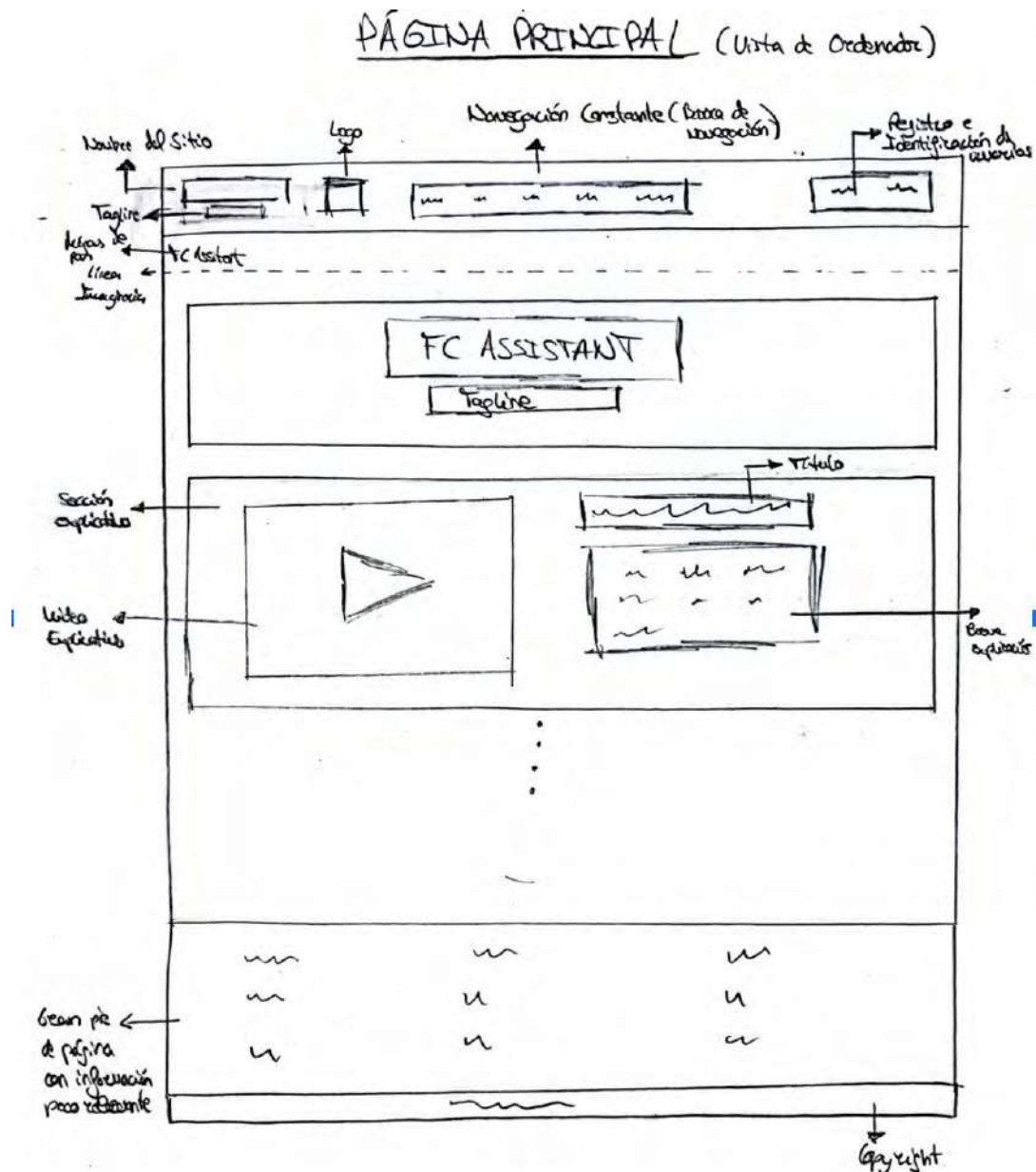


Figura 4.21: Wireframe de la página principal

- La totalidad de las plataformas competidoras que ofrecen funcionalidades de creación de plantillas prescinden del nombre de página, lo que sugiere una convención consolidada en este tipo de interfaces.
- La eliminación del encabezado permite optimizar el espacio disponible, favoreciendo la incorporación de opciones y elementos de interés funcional para el usuario.
- Esta decisión rompe con la estructura visual típica de las secciones informativas del sitio, con el propósito de generar una percepción diferenciada. El objetivo es que el usuario identifique esta página como un entorno eminentemente interactivo, en el que se fomenta la exploración y el descubrimiento de funcionalidades.

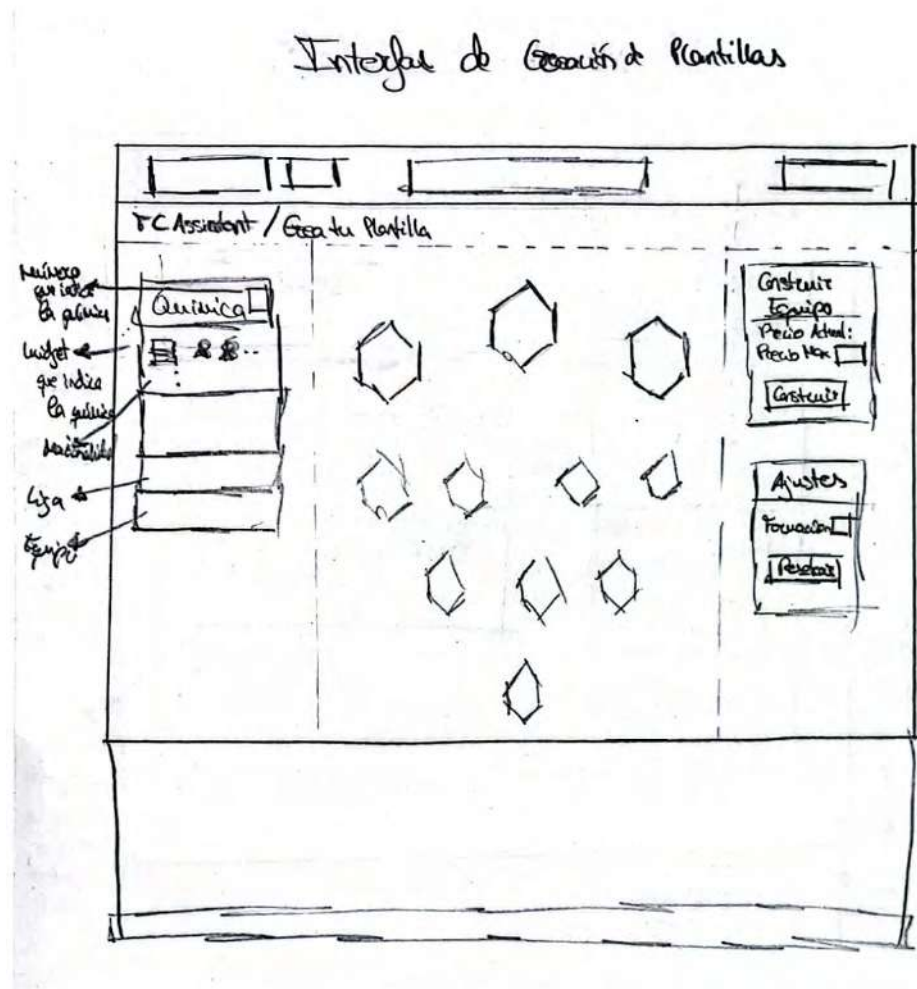


Figura 4.22: Wireframe de la página de creación de plantillas

Con el fin de mejorar la experiencia de usuario y optimizar la interacción con la interfaz, la página se ha estructurado en tres secciones diferenciadas:

- **Sección izquierda.** Destinada a la visualización de gráficos que se actualizan dinámicamente en función de la interacción del usuario. Esta sección puede mostrar:
 - **Gráfico de química del equipo.** Se muestra cuando no se ha seleccionado ninguna posición concreta dentro de la formación.
 - **Gráficos informativos sobre los jugadores.** Se presentan cuando el usuario selecciona una posición específica de la formación. Dependiendo del contexto, pueden visualizarse los siguientes gráficos:
 - **Gráfico de representación de importancias.** Gráfico de sectores que representa las prioridades establecidas para esa posición. Se muestra

únicamente si la posición seleccionada aún no está ocupada por ningún jugador.

- **Gráfico de estadísticas del jugador.** Gráfico tipo radar (o gráfico de araña) que resume las principales estadísticas del jugador asignado a la posición seleccionada.
- **Sección central.** Alberga la representación gráfica de la formación, diseñada conforme al modelo mental del usuario. Esta representación busca facilitar la comprensión del esquema táctico y aportar una experiencia visual, dinámica y lúdica, que supera las limitaciones de una interfaz puramente textual o convencional.
- **Sección derecha.** Contiene los diferentes ajustes configurables que afectan al equipo de forma global. Esta decisión responde a una convención extendida en diseño de interfaces, según la cual los elementos más estáticos se ubican a la izquierda, mientras que los elementos más dinámicos y configurables se sitúan a la derecha, reforzando así la lógica visual y funcional del entorno.

A partir del análisis comparativo con las plataformas de creación de plantillas de la competencia, se ha identificado que muchas de sus interfaces presentan una disposición abigarrada de elementos y no consideran adecuadamente la sobrecarga cognitiva y memorística que pueden experimentar usuarios potenciales, especialmente aquellos jóvenes o con poca experiencia en este tipo de entornos. Este perfil de usuario, por lo general, no está dispuesto a invertir tiempo en interfaces poco intuitivas y saturadas de opciones.

El diseño propuesto pretende abordar esta problemática mediante una organización más eficiente y progresiva de la información, mostrando únicamente aquellos elementos que resultan necesarios en cada contexto. Con este objetivo, se prescinde de los menús extensos ubicados en la parte derecha de la interfaz, sustituyéndolos por una ventana modal que se despliega únicamente cuando el usuario selecciona una posición en la formación. Esta solución ofrece varias ventajas:

- **Descongestionar la sección derecha,** reservándola exclusivamente para los ajustes globales de la formación, evitando así la saturación visual.
- **Incorporar de forma estructurada y visualmente atractiva toda la información relevante para la toma de decisiones sobre una posición concreta,** favoreciendo una experiencia más fluida y orientada a la acción.

4.9.7. Ventana Modal de Comparación de Jugadores

La ventana modal de comparación de jugadores se concibe como un cuadro de mando diseñado para facilitar al usuario la resolución de un problema de toma de decisiones individual, permitiéndole seleccionar el jugador más adecuado para una posición concreta en función de sus propias prioridades (ver Figura 4.23).

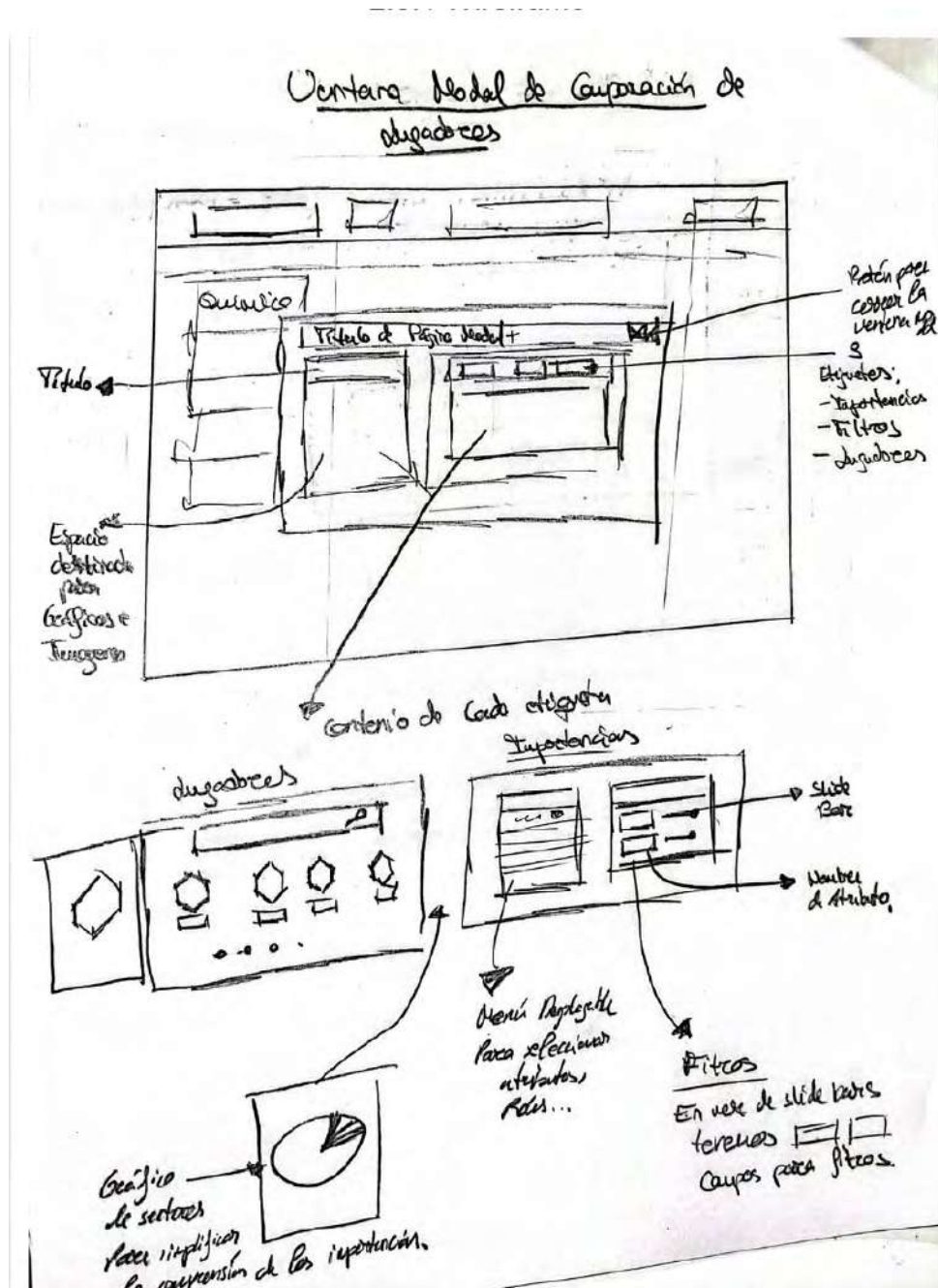


Figura 4.23: Wireframe de la ventana modal de comparación de jugadores

La integración de un proceso de toma de decisiones en una interfaz orientada a usuarios no expertos constituye un desafío no trivial. Para afrontarlo, se ha desarrolla-

do un diseño creativo, coherente y centrado en la experiencia de usuario, adoptando las siguientes decisiones:

Encabezado Informativo

En la parte superior de la ventana se incorpora un encabezado que contiene el título de la ventana y la posición específica que se está configurando. Esta elección resulta esencial para reducir la carga cognitiva del usuario y facilitar su orientación a lo largo del proceso de selección.

Etiquetas Principales

Bajo el encabezado se disponen tres etiquetas que estructuran el contenido de la ventana de forma clara e intuitiva:

- **Jugadores** (seleccionada por defecto).
- **Importancias.**
- **Filtros.**

Sección *Jugadores*

Al acceder a la ventana, el usuario visualiza por defecto los jugadores disponibles para la posición seleccionada, ordenados por precio de mayor a menor, siguiendo criterios de accesibilidad económica en función del presupuesto definido.

Dentro de esta sección se ofrecen los siguientes modos de interacción:

- Al hacer clic sobre un jugador, este se selecciona y su tarjeta se muestra ampliada en el panel izquierdo.
- Si el usuario sitúa el puntero sobre otro jugador mientras uno ya está seleccionado, se genera una comparativa gráfica entre ambos, lo que permite visualizar de forma concisa y efectiva las diferencias entre sus atributos. Este recurso facilita una toma de decisiones más precisa y fundamentada.

Sección *Importancias*

La sección importancias despliega un menú desplegable que permite al usuario seleccionar los aspectos que desea priorizar durante el proceso de comparación:

- **Atributos.** Atributos numéricos escalados en el rango $[0, 100]$.
- **Roles.**
- **Estilos de juego.**
- **Detalles del jugador.** Como el precio, la química, la altura o atributos numéricos no escalados, entre otros.

Cada uno de los criterios seleccionados se incorpora a una lista situada en el panel derecho, junto con un control deslizante (*slider*) que permite ajustar su importancia relativa respecto al resto.

Durante el proceso de evaluación con usuarios, se identificó que este mecanismo no resultaba lo suficientemente intuitivo, ya que no se ajustaba plenamente al modelo mental de los participantes. En concreto, tendían a interpretar que “cuanto más a la derecha, mayor importancia”, cuando en realidad la relevancia viene determinada por la separación entre los distintos deslizadores, es decir, por su relación relativa entre sí y no por su posición absoluta. Esta discrepancia generaba confusión y dificultaba la correcta interpretación del sistema.

Para solventar este problema, se ha incorporado en el panel izquierdo un diagrama de sectores explicativo, que se actualiza dinámicamente para representar gráficamente la proporción asignada a cada aspecto, proporcionando así una referencia visual clara y comprensible.

Sección *Filtros*

La sección filtros mantiene una estructura de interacción coherente con la del apartado anterior:

- Selección de filtros mediante un menú desplegable.
- Configuración específica de cada filtro en el panel derecho.

4.10. Implementación

En esta sección se exponen los aspectos más relevantes relacionados con la implementación del proyecto. En primer lugar, se detallan las herramientas empleadas durante el desarrollo. A continuación, se presentan tres bloques diferenciados que abordan: el modelo de toma de decisiones, el algoritmo de optimización y una función personalizada para el cálculo de la química, diseñada específicamente para este trabajo. Esta última constituye una propuesta original, sin precedentes identificados en los recursos disponibles en línea.

4.10.1. Herramientas utilizadas

Dada la complejidad del sistema y la variedad de componentes que lo conforman, la elección de cada herramienta se justifica en la sección correspondiente al diseño de cada componente específico. No obstante, en esta sección se proporciona una visión general de las tecnologías empleadas, recopiladas en la Tabla 4.2, donde se especifican las herramientas utilizadas en cada uno de los componentes principales del sistema.

Componente	Herramientas
Conjunto de Scrapers	Python, Playwright
Base de Datos Distribuida	MongoDB
Servidor Web	Angular
Servidor Back-end	NestJS, BullMQ
Servidor Redis	Redis
Servidor de Workers	NestJS
Algoritmo de Toma de Decisión	Java
Algoritmos de Optimización	Java
Servidores RAG	Python

Tabla. 4.2: Tecnologías utilizadas en cada componente del sistema

4.10.2. TOPSIS

El modelo de toma de decisión que se ejecuta en los servidores *workers* para abordar el problema de selección de jugadores para una posición específica dentro de la plantilla es TOPSIS (véase la Sección 2.2.3).

La implementación desarrollada para resolver este problema en particular se describe en el Algoritmo 1.

Algoritmo 1: TOPSIS

Result: Ranking de jugadores para una posición
inicialización;
normalización min-max;
definición de soluciones ideal y anti-ideal;
calculo del índice de similaridad para cada alternativa;
elaboracion del ranking en base al índice de distancia relativa;

■ **Proceso de Inicialización:** este proceso consiste en recibir como parámetros los siguientes elementos:

- Las prioridades definidas por el usuario respecto a los distintos criterios considerados.
- La posición específica para la cual se realiza la selección.
- Los filtros especificados por el usuario, que delimitan el espacio de soluciones disponibles.
- El precio máximo permitido para el jugador, en función de las restricciones presupuestarias.
- Los jugadores presentes en la formación, necesarios para el cálculo de la química.

Con esta información, se procede a obtener los datos correspondientes desde una base de datos local reducida para dicha posición, lo que implica la lectura de un archivo JSON y, posteriormente, la vectorización de los jugadores. Este proceso consiste en incorporar el valor de cada atributo del jugador —correspondiente al criterio presente en vector de prioridades proporcionado por el usuario— en el vector representativo del jugador, manteniendo el mismo orden que el vector de prioridades. Es importante señalar que los atributos nominales y ordinales se procesan de manera diferenciada:

- **Atributos nominales:** se asigna un valor de 1 si el jugador posee dicha característica y 0 en caso contrario.
- **Atributos ordinales:** cada categoría nominal se codifica mediante un número entero diferente, que preserva el orden relativo inherente a las categorías nominales correspondientes.

De manera simultánea a la vectorización, se aplican los filtros definidos por el usuario para delimitar el conjunto de alternativas realmente disponibles en el problema.

- **Normalización min-max:** las razones por la que se implementa este método de normalización las podemos ver en la Sección [4.7.1](#).
- **Definición de las soluciones ideal y anti-ideal:** en esta etapa, se asumen puntos ideales y anti-ideales absolutos. Gracias a la normalización min-max, el valor máximo de cada atributo es siempre 1, y el mínimo, 0.
- **Calculamos los coeficientes de cercanía relativa para cada alternativa.**
- **Elaboración del ranking:** una vez calculados los coeficientes de cercanía relativa para cada alternativa, se procede a la construcción del ranking que ordena las n opciones disponibles. Para ello, se recorre el vector que almacena dichos coeficientes y se inserta cada jugador en una cola de prioridad ordenada de acuerdo con el valor obtenido mediante la métrica de TOPSIS. Finalmente, se extraen de esta estructura los n jugadores requeridos, conformando el ranking definitivo.

4.10.3. ACOS

El método de optimización seleccionado para la creación automática de plantillas es un **ACOS optimizado mediante Búsqueda Local** cuyo diseño se detalla en la Sección [4.7.1](#). Este algoritmo ha sido implementado de forma concurrente y su funcionamiento puede describirse a través de los pasos que vienen descritos en el Algoritmo [2](#).

- **Proceso de Inicialización:** el proceso de inicialización primeramente consiste en recibir como parámetros:
 - Un vector en el que se puede incluir en cada posición:
 - Una lista prioridades y filtros correspondientes a la posición de la formación a la que hace referencia la posición del vector.
 - El id de un jugador que ya está seleccionado en la plantilla que hace referencia
 - El precio máximo de que puede alcanzar la plantilla.
 - Los jugadores cuyo valor no contemplamos a la hora de calcular el precio total de la plantilla.

Algoritmo 2: ACOS

Result: Lista de 11 jugadores óptimos según criterios de puntuación y química
Inicialización;

for cada iteración **do**

for cada hormiga en paralelo **do**

 Construir solución;

 Actualización local de feromonas;

 Aplicar búsqueda local a la solución generada;

 Evaluar solución;

end

 Actualizar mejor solución global encontrada;

 Evaporar feromonas en toda la matriz;

 Actualizar feromonas basado en calidad de soluciones encontradas;

 Refuerzo elitista: actualizar feromonas con la mejor solución global;

end

;

return mejor solución encontrada;

- Los diferentes filtros que haya especificado el usuario delimitando el espacio de soluciones disponibles.

Una vez obtenidos estos valores, se aplica el método TOPSIS individualmente para cada posición, generando así un ranking para cada una. Los elementos de estos rankings conformarán el espacio de soluciones, definido como el conjunto de todas las combinaciones posibles formadas seleccionando un elemento de cada ranking TOPSIS.

Posteriormente se inicializan los parámetros propios del sistema de colonias de hormigas, estos parámetros se detallan de manera extensa en la Sección 2.3.2.

Por último, se inicializa la matriz de feromonas.

- **Construir Solución:** para construir soluciones primero se calculan unas probabilidades y posteriormente seleccionamos un jugador para cada posición.

- **Cálculo de probabilidades.** La selección del siguiente jugador por parte de una hormiga se rige por una combinación de estrategias de explotación y exploración, según la regla definida en la Ecuación 4.5. Si se opta por la exploración ($q > q_0$), la selección se realiza de forma probabilística conforme a la Ecuación 4.6, donde la probabilidad de seleccionar un jugador depende de la cantidad de feromona y de una heurística asociada.

$$j = \begin{cases} \arg \max_{l \in \mathcal{N}_i^{(k)}} (\tau_{il} \cdot \eta_{il}^\beta), & \text{si } q \leq q_0 \\ \text{Seleccionado según } p_{ij}^{(k)}, & \text{si } q > q_0 \end{cases} \quad (4.5)$$

- j : Próximo nodo (jugador) que selecciona la hormiga k desde el nodo i .
 q : Número aleatorio generado uniformemente en el intervalo $[0, 1]$.
 q_0 : Parámetro del algoritmo que controla el equilibrio entre explotación y exploración.
 τ_{il} : Cantidad de feromona presente en el arco o alternativa (i, l) .
 η_{il} : Información heurística asociada al arco (i, l) , puntuación TOPSIS.
 β : Parámetro que determina la influencia relativa de la heurística.
 $p_{ij}^{(k)}$: Probabilidad de seleccionar el nodo j desde i según una regla estocástica.
 $\mathcal{N}_i^{(k)}$: Conjunto de nodos aún no visitados por la hormiga k al estar en i .

$$p_{ij}^{(k)} = \begin{cases} \frac{[\tau_{ij}]^\alpha \cdot [\eta_{ij}]^\beta}{\sum_{l \in \mathcal{N}_i^{(k)}} [\tau_{il}]^\alpha \cdot [\eta_{il}]^\beta}, & \text{si } j \in \mathcal{N}_i^{(k)} \\ 0, & \text{en otro caso} \end{cases} \quad (4.6)$$

- j : Próximo nodo (jugador) que selecciona la hormiga k desde el nodo i .
 q : Número aleatorio generado uniformemente en el intervalo $[0, 1]$.
 q_0 : Parámetro que controla el equilibrio entre explotación y exploración.
 τ_{ij} : Cantidad de feromona presente en el arco (i,j) .
 η_{ij} : Información heurística asociada al arco (i,j) .
 α, β : Parámetros que modulan la influencia de la feromona y la heurística.
 $p_{ij}^{(k)}$: Probabilidad de transición del nodo i al nodo j para la hormiga k .
 $\mathcal{N}_i^{(k)}$: Conjunto de nodos aún no visitados por la hormiga k al estar en i .

- Búsqueda Local:** se parte de una solución inicial y se explora sistemáticamente su vecindario para encontrar mejoras. En cada iteración, se generan soluciones vecinas intercambiando un jugador de la solución actual por otro disponible en el mismo subconjunto. Cada solución vecina se evalúa y se compara con la mejor encontrada hasta el momento. Si la nueva solución es válida y su puntuación supera la actual, se actualiza como la mejor solución temporal. El proceso continúa hasta examinar todos los posibles vecinos, devolviendo la mejor solución encontrada en el entorno local (ver Algoritmo 3).

Algoritmo 3: Algoritmo de Búsqueda Local

Result: Mejor solución vecina encontrada en la búsqueda local
 Inicializar `mejorVecino` como copia de la solución actual;
 Calcular puntuación `mejorPuntuacion` de `mejorVecino`;
for cada posición i en la solución **do**
 for cada jugador j en el subconjunto i **do**
 if el jugador j es diferente al jugador actual en posición i **then**
 Crear vecino `vecinoActual` copiando la solución actual;
 Reemplazar jugador en posición i con jugador j ;
 if la solución `vecinoActual` es válida **then**
 Calcular puntuación `puntuacionActual` de `vecinoActual`;
 if `puntuacionActual` > `mejorPuntuacion` **then**
 Actualizar `mejorVecino` CON `vecinoActual`;
 Actualizar `mejorPuntuacion` CON `puntuacionActual`;
 end
 end
 end
end
end
return `mejorVecino`;

- **Evaluar Solución:** este problema se centra en maximizar media aritmética entre el valor de la química normalizado y el valor medio de TOPSIS para el conjunto de jugadores que forman la solución. Aunque si el precio de la plantilla es superior al máximo que impone el presupuesto la función objetivo devuelve un 0. Podemos ver la expresión matemática de la función objetivo en la Ecuación 4.7.

$$f_{\text{objetivo}} = \text{máx} \left(\frac{1}{2} \left(\frac{\text{química}}{33} + \frac{1}{n} \sum_{i=1}^n \text{TOPSIS}_i \right) \right) \quad (4.7)$$

f_{objetivo} : Valor de la función objetivo que se desea maximizar.

química : Valor total de la química del equipo (máximo 33).

TOPSIS_i : Puntuación normalizada del jugador i según el método TOPSIS.

n : Número total de jugadores seleccionados en la plantilla.

- **Cálculo de la Química:** para que nuestra función objetivo sea eficiente, no podemos calcular la química recorriendo el vector de jugadores múltiples veces por característica. Utilizamos tres mapas (equipos, ligas y nacionalidades) que se rellenan en un único recorrido, almacenando cuántos jugadores hay por equipo, liga y país. Esto permite calcular la química de cada jugador en tiempo constante ($O(1)$), evitando redundancias. Así consigui-

mos que el cálculo sea rápido y escalable incluso en escenarios con muchas combinaciones. Podemos observar la implementación de esta función en el Código 4.1.

```
1 private double evaluate(List<Jugador_Data> solution) {
2     double score_media_ponderada = 0.0;
3     double puntos_de_quimica[] = new double[solution.size()];
4     Map<String, Integer> equipos = new HashMap<>();
5     Map<String, Integer> ligas = new HashMap<>();
6     Map<String, Integer> nacionalidades = new HashMap<>();
7     double precio_total = 0.0;
8     double puntos_de_quimica_totales = 0.0;
9     Set<String> nombres_jugadores = new HashSet<>();
10
11     Arrays.fill(puntos_de_quimica, 0.0);
12
13     int i = 0;
14     for (Jugador_Data jugador: solution) {
15         if(this.ignorar_precio.equals("-1")) {
16             precio_total += jugador.price;
17         }
18
19         if (nombres_jugadores.contains(jugador.name) ||
20             precio_total > precio_maximo) {
21             return 0.0;
22         }
23         nombres_jugadores.add(jugador.name);
24
25         score_media_ponderada += jugador.puntuacion;
26
27         if (!(jugador.card_type.contains("Hero") || jugador.
28             card_type.contains("Heroes") || jugador.card_type.
29             equals("Icon"))) {
30             equipos.merge(jugador.club, 1, Integer::sum);
31         }
32         if (!jugador.card_type.equals("Icon")) {
33             ligas.merge(jugador.league, 1, Integer::sum);
34         }
35         if (!(jugador.card_type.contains("Hero") || jugador.
36             card_type.contains("Heroes"))) {
37             nacionalidades.merge(jugador.nation, 1, Integer::
```

```
        sum);
34     }
35
36     if (jugador.card_type.contains("Hero") || jugador.
        card_type.contains("Heroes")) {
37         puntos_de_quimica[i] = 3.0;
38         ligas.merge(jugador.league, 1, Integer::sum);
39     } else if (jugador.card_type.equals("Icon")) {
40         puntos_de_quimica[i] = 3.0;
41         nacionalidades.merge(jugador.league, 1, Integer::
            sum);
42     }
43     i++;
44 }
45
46 // Calcular química para todos los jugadores
47 for (i = 0; i < solution.size(); i++) {
48     if (puntos_de_quimica[i] != 3.0) {
49         Jugador_Data jugador = solution.get(i);
50         puntos_de_quimica[i] = evaluar_quimica_pais(
            nacionalidades.getDefault(jugador.nation, 0).
            doubleValue()) +
51             evaluar_quimica_liga(ligas.
                getDefault(jugador.
                    league, 0).doubleValue())
                +
52             evaluar_quimica_equipo(
                equipos.getDefault(
                    jugador.club, 0).
                    doubleValue());
53         if (puntos_de_quimica[i] > 3.0) {
54             puntos_de_quimica[i] = 3.0;
55         }
56     }
57     puntos_de_quimica_totales += puntos_de_quimica[i];
58 }
59
60 score_media_ponderada /= solution.size();
61 puntos_de_quimica_totales = puntos_de_quimica_totales / (
    solution.size() * 3);
62
```

```
63     return (score_media_ponderada * 0.5) + (  
64         puntos_de_quimica_totales * 0.5);  
    }
```

Listado 4.1: Función objetivo

Capítulo 5

ANÁLISIS DE RESULTADOS

5.1. Introducción

Con el objetivo de validar el correcto funcionamiento del sistema y garantizar una experiencia de usuario óptima, comprobaremos cómo el sistema resuelve los casos de uso más importantes:

- Crear una plantilla:
 - Definiendo un presupuesto máximo.
 - Seleccionando una formación a nuestro gusto.

- Configurar posición:
 - Configurando prioridades.
 - Configurando filtros.
 - Comparando jugadores visualizando gráficos comparativos y sus precios.

- Consultar al asistente:
 - Solicitando que analice nuestra plantilla.
 - Obteniendo información sobre el juego.
 - Creando un equipo automáticamente.

5.2. Consultar al Asistente

En primer lugar, imaginemos que somos un usuario con conocimientos muy básicos sobre FC Ultimate Team. Nuestro objetivo es conseguir un equipo con el que poder jugar con amigos, disponiendo para ello de un presupuesto de 20.000 monedas.

5.2.1. Consultar cuestiones sobre el juego

Lo primero que haremos será consultar al sistema aquellas cuestiones que no conozcamos, por ejemplo:

¿Cuál es la mejor formación para un usuario que está empezando a jugar?

Respuesta del asistente a la pregunta

La mejor formación para un usuario principiante en **FC 25 Ultimate Team** es la **4-4-2 clásica**. **¿Por qué la 4-4-2 es ideal para principiante?**

- **Equilibrio natural:** Líneas claras y roles definidos.
- **Fácil de ejecutar:** No requiere automatismos complejos.
- **Versatilidad táctica:** Útil para contraataques, juego directo o posesión básica.

Instrucciones individuales recomendadas

- **Delanteros:** *Desmarcarse*, ideal para contraataques.
- **Centrocampistas:** Apoyo equilibrado en transiciones.
- **Extremos:** Apoyo ofensivo moderado.
- **Defensas:** Instrucciones estándar para solidez táctica.

Roles y atributos clave por posición

Posición	Atributos clave	Rol recomendado
Portero	Reflejos, posicionamiento, control del área	Seguridad bajo palos
Centrales (2)	Fuerza, marcaje, anticipación, juego aéreo	Cortar ataques, despejar
Laterales (2)	Resistencia, velocidad, centros	Apoyo ofensivo y defensivo
MC Defensivo	Intercepciones, stamina, posicionamiento	Contención (doble pivote)
MC Creativo	Pase, visión, control	Distribución de juego
Extremos (2)	Velocidad, regate, centros	Desborde y asistencia
Delantero rápido	Desmarque, definición, aceleración	Ruptura de líneas
Delantero físico	Fuerza, juego de espaldas, pase	Apoyo y asociación

Estilo de juego recomendado

- **Transiciones rápidas (contraataque):** Aprovecha la velocidad.
- **Bloque medio defensivo:** Evita desorganización.
- **Juego directo:** Menos posesión, más verticalidad.

Factores clave de rendimiento

- **Stamina y recuperación:** Clave en extremos y laterales.
- **Química entre jugadores:** Mejora sinergias.
- **Compactación defensiva:** Para evitar huecos entre líneas.

Limitaciones y advertencias

- No se ha considerado plantilla concreta, por lo que es una guía general.
- La 4-4-2 puede sufrir frente a formaciones con superioridad numérica en el medio.

Recomendaciones prácticas

- Empieza con una 4-4-2 básica y estilo de juego directo.
- Asigna roles simples, coherentes con los atributos de cada jugador.
- Evalúa tras cada partido: posesión, xG, recuperaciones.
- Ajusta instrucciones e intensidad defensiva según el rival.
- Explora variantes como 4-2-3-1 o 4-1-2-1-2 cuando domines la base.

Una vez que ya sabemos la formación que tenemos que utilizar, podemos cuestionarnos cuál es el estilo de juego que queremos emplear jugando con esa formación. Por ello, se hace una consulta a la asistente en ese sentido, preguntándole, ¿cuáles son los estilos de juego que puedo utilizar cuando juego con la formación 4-4-2?

¿Cuáles son los estilos de juego que puedo utilizar cuando juego con la formación 4-4-2?**Respuesta del asistente a la pregunta**

Claro, la formación 4-4-2 en FC 25 Ultimate Team es muy versátil y se adapta a varios estilos de juego según el perfil de tus jugadores. Sin embargo, sin información sobre tu plantilla actual, te comparto los estilos más efectivos que puedes aplicar y cómo optimizarlos tácticamente.

Conclusión Principal:

La 4-4-2 funciona muy bien con estilos de juego como presión alta, contraataque o juego directo, siempre que combines correctamente los perfiles de tus delanteros, extremos y mediocampistas. Elegir el estilo adecuado depende del ritmo, resistencia y habilidades técnicas de tus jugadores.

Estilos de juego recomendados para 4-4-2

- **Juego Directo (ataques rápidos, verticales):** Ideal si tienes extremos rápidos y regateadores + delanteros móviles o cazagoles. Beneficia a equipos con ritmo alto y buena definición. Requiere laterales ofensivos que acompañen las bandas.
- **Contraataque:** Funciona si tus extremos y delanteros son veloces y con buen desmarque. Usa mediocampistas con buen pase largo y visión para lanzar transiciones. Mantén un bloque medio o bajo para recuperar y salir rápido.
- **Presión Alta:** Exige delanteros con agresividad y resistencia para presionar constantemente. Mediocampistas con ritmo y stamina son clave para sostener el bloque medio. Laterales deben tener buen ritmo y capacidad de recuperación.
- **Posesión (menos común, pero viable):** Requiere extremos pasadores y mediocampistas con buen pase corto y visión. Delanteros deben combinar bien (por ejemplo, un delantero móvil + uno de referencia). Menos efectiva si tus jugadores no tienen alta toma de decisiones y control.

Roles clave según estilo de juego

Posición	Juego Directo / Contraataque	Posesión
ST	Móvil + Cazagoles	Móvil + Hombre Objetivo
RM/LM	Regateador rápido	Pasador con visión
CM	Uno box-to-box + uno defensivo	Dos con pase y visión
LB/RB	Ofensivos con ritmo	Equilibrados o conservadores

Factores clave de rendimiento

- Ritmo y resistencia en CM: Vital para sostener presión o transiciones.
- Perfil técnico de extremos: Define si juegas por bandas o por dentro.
- Sincronización laterales–mediocampo: Si los laterales suben, necesitas CMs que cubran.
- Combinación de delanteros: Usa roles complementarios (ej. móvil + finalizador) para romper líneas.

Limitaciones de información

Actualmente, no se conoce el perfil específico de tus jugadores (atributos, estilo, química). Esto impide recomendar un estilo óptimo personalizado.

Recomendaciones adicionales

- Prueba primero con juego directo o contraataque si tienes extremos veloces.
- Usa un delantero móvil (con buen ritmo y regate) y otro cazagoles (con definición y posicionamiento).

- Configura tus laterales para que suban solo si tienen ritmo y stamina suficientes.
- Ajusta el bloque defensivo según tu estilo: alto para presión, medio para contraataque.

5.2.2. Crear un equipo Automáticamente

Ahora que ya sabemos qué formación vamos a utilizar y las estrategias que podemos emplear, quiero que el sistema me genere una plantilla para cada posible estilo de juego teniendo en cuenta mi presupuesto de 20000 monedas (ver Figuras 5.1, 5.2 y 5.3).

Contraataque



Figura 5.1: Plantilla generada automáticamente para jugar al contraataque con la formación 4-4-2 (20000 monedas)

He solicitado al asistente que también explique el porqué de las prioridades asignadas. Esta consulta se muestra en la Tabla 5.1.

Posición	Prioridades Clave	Sentido Táctico
Portero (GK-0)	Posicionamiento (90), Reflejos (90), Portero Líbero (80)	Seguridad en 1v1, reduce riesgos en balones filtrados, inicia contraataques con precisión.
Centrales (CB-1, CB-2)	Intercepciones (91), Conciencia Defensiva (90), Entrada Firme (85), Defensa con Balón (85)	Cortan ataques sin desorden, inician transiciones rápidas desde el pase corto.
Laterales (LB-3, RB-4)	Velocidad de Sprint (90), Resistencia (85), Centros (80), Lateral Ofensivo (85)	Suben rápido por banda y repliegan con intensidad, sosteniendo ritmo en transiciones.
Mediocentros (CM-5, CM-6)	Visión (91), Pase Corto (90), Resistencia (85), Box to Box (85)	Conectan defensa y ataque, filtran pases rápidos y cubren mucho campo.
Extremos (LM-7, RM-8)	Velocidad de Sprint (90), Regate (85), Centros (80), Extremo (85)	Rompen líneas en velocidad, asisten tras desbordes, generan desequilibrio.
Delanteros (ST-9, ST-10)	ST-9 (Poacher): Definición (91), Posicionamiento Ofensivo (90), Potencia de Tiro (85), Delantero Avanzado (85) ST-10 (False 9): Definición (90), Visión(70), Falso 9 (85)	El Poacher ataca el área, el False 9 genera espacios al arrastrar marcas, combinación ideal en transiciones rápidas.

Tabla. 5.1: Prioridades tácticas por posición en esquema 4-4-2 para contraataque

Presión Alta



Figura 5.2: Plantilla generada automáticamente para presionar desde arriba con la formación 4-4-2 (20000 monedas)

La Tabla 5.2 ofrece un desglose detallado de las prioridades asignadas a cada posición y su sentido táctico.

Posición	Prioridades Clave	Sentido Táctico
Portero (GK-0)	Reflejos (91), Posicionamiento (81), Portero Líbero (80)	Reacciona rápido en 1v1 tras pases filtrados por defensa adelantada. Menor prioridad a kicking al no buscar juego directo.
Centrales (CB-2, CB-3)	Fuerza (91), Intercepciones (91), Conciencia Defensiva (81), Defensa Directa (91)	Imprescindibles para ganar duelos físicos y cortar pases entre líneas. Despejan sin complicaciones si se rompe la presión alta.
Laterales (LB-1, RB-4)	Aceleración (81), Intercepciones (91), Resistencia (71), Lateral Ofensivo (81)	Cortan contras rivales por banda y permiten presión alta constante gracias a su aceleración y capacidad de retorno.
Mediocentros (CM-6, CM-7)	Visión (91), Pase Corto (81), Resistencia (71), Medio Centro (81)	Gestionan la reorganización tras recuperación en campo rival, filtran pases rápidos y mantienen intensidad durante el partido.
Extremos (LM-5, RM-8)	Velocidad de Sprint (81), Regate (81), Centros (71), Extremo (81)	Protagonistas en las transiciones ofensivas, ganan metros por fuera y generan superioridad tras recuperación alta.
Delanteros (ST-9, ST-10)	Definición (91), Posicionamiento Ofensivo (81), Velocidad de Sprint (71), Delantero Avanzado (91)	Presionan a la defensa rival, convierten errores en goles y atacan espacios vacíos tras robo.

Tabla. 5.2: Prioridades tácticas por posición en esquema 4-4-2 con presión alta

Posesión



Figura 5.3: Plantilla generada automáticamente para mantener la posesión con la formación 4-4-2 (20000 monedas)

La Tabla 5.3 recoge la explicación detallada de las prioridades establecidas para cada posición, enfocadas en mantener la posesión.

Posición	Prioridades Clave	Sentido Táctico
Portero (GK-0)	Reflejos (90), Posicionamiento (85), Portero Líbero (60)	Seguridad bajo presión es clave. El juego con los pies es secundario pero suficiente. <i>De Gea</i> encaja, aunque un portero con mejor pase corto potenciaría salidas seguras.
Centrales (CB-1, CB-2)	Intercepciones (90), Conciencia Defensiva (85), Defensa con Balón (80)	Solidez defensiva combinada con capacidad para iniciar jugadas desde atrás. <i>Hummels</i> lidera la salida, <i>Buchanan</i> complementa con solidez física.
Laterales (LB-3, RB-4)	Aceleración (85), Velocidad de Sprint (80), Centros (75), Lateral Ofensivo (85)	Aportan amplitud y apoyan ofensivamente. Su responsabilidad defensiva es moderada. <i>Catley</i> encaja perfectamente; <i>Abdulhamid</i> debe dosificarse defensivamente.
Mediocentros (CM-6, CM-7)	Visión (90), Pase Corto (85), Pase Largo (80), Box to Box (85)	Doble pivote mixto que controla el ritmo del partido. <i>Little</i> creativa; <i>Walsh</i> sostiene el equilibrio. Excelente pareja para posesión sostenida.
Extremos (LM-5, RM-8)	Aceleración (85), Velocidad de Sprint (80), Centros (80), Extremo (90)	Verticalidad medida. Generan amplitud y desequilibrio. <i>Foord</i> ideal por izquierda; <i>Ikone</i> veloz pero menos fiable en circulación.
Delanteros (ST-9, ST-10)	Posicionamiento Ofensivo (90), Definición (90), Potencia de Tiro (85), Delantero Referencia (80)	<i>Dovbyk</i> fija centrales como referencia (Poacher), <i>Asseyi</i> combina con mediocentros y bandas. Equilibrio entre finalización y apoyo.

Tabla. 5.3: Sentido de la asignación de prioridades en esquema 4-4-2 orientado a posesión

5.2.3. Evaluación de Plantillas

Se ha solicitado al asistente la evaluación de cada una de las plantillas previamente generadas, teniendo en cuenta tanto la formación como el estilo de juego asociado. Además, se ha especificado que las respuestas debían presentarse en formato de tabla para facilitar su análisis y comparación.

Contraataque

La evaluación generada por el asistente para la plantilla cuyo estilo de juego se basa en el contraataque se presenta en la Tabla 5.4.

Jugador	Fortalezas	Debilidades / Recomendación
Jonathan Tah (CB)	Defensa posicional, físico (92 fuerza)	Aceleración baja (53), necesita compañero rápido
Joseph Aidoo (CB)	Físico, buen tackle	Pase largo pobre, aceleración justa
David Raum (LB)	Rápido, stamina, centros	Defensa decente, mejor como carrilero o con apoyo
Lucas Vázquez (RB)	Equilibrado, stamina, versatilidad	Recomendado, ideal para el sistema
Luka Modrić (CM)	Visión, pase top	Ritmo bajo, físico débil. Necesita MC defensivo al lado
Mario Götze (CM)	Pase y control	Ritmo muy bajo, defensa deficiente. Mejor como CAM
Nico Williams (LM)	Ritmo, regate, 5 pierna mala	Perfecto para contraataque por banda
Karim Adeyemi (RM)	Velocidad, desborde, 4 skills	Débil defensivamente. Alternar con Iñaki o como ST
Laura Freigang (ST)	Finalización, posicionamiento, fuerza	No muy rápida, mejor como referencia
Iñaki Williams (ST)	Ritmo, stamina, salto	3 pierna mala, visión justa. Ideal para atacar espacios
Manuel Neuer (GK)	Pase, colocación, reacciones	Perfecto para defensa adelantada y contraataques
Recomendaciones generales		
<ul style="list-style-type: none"> ■ Mantener: Iñaki, Nico, Lucas Vázquez, Modrić (con un MC defensivo), Neuer. ■ Cambiar: Sustituir a Götze por un MC defensivo box-to-box o MCD puro. ■ Ajustar: Buscar un LB más defensivo o un CB más rápido; alternar Adeyemi según rival. 		

Tabla. 5.4: Evaluación resumida de la plantilla (4-4-2 Contraataque, 20000 monedas)

Presión Alta

La evaluación generada por el asistente para la plantilla cuyo estilo de juego se basa en mantener una presión alta se presenta en la Tabla 5.5.

Jugador	Fortalezas	Debilidades / Recomendación
Manuel Neuer (GK)	Pase, colocación, juego en largo	Velocidad baja (49), mantener como sweeper keeper
David Raum (LB)	Ritmo (85), stamina (91), centros	Perfecto para sistema de presión alta
Joseph Aidoo (CB)	Físico, tackles agresivos	Pase deficiente, riesgo en salida bajo presión
Emre Can (CB)	Agresividad, físico, buen pase corto	Mantener, buen central reconvertido para presión alta
Lucas Vázquez (RB)	Stamina (92), conducción, equilibrio	Ideal para sistema ofensivo y presión constante
Pedri (CM)	Pase corto (88), visión, stamina	Mantener, ideal como interior box-to-box
Mario Götze (CM)	Control, pase en corto	Ritmo bajo (55), físico débil → sustituir por MC físico
Nico Williams (LM)	Velocidad, regate, desborde por izquierda	Defiende poco (36), pero aporta en transiciones rápidas
Iñaki Williams (RM)	Ritmo (94), stamina (88), físico (84)	Puede alternar como ST, mantener
Benjamin Šeško (ST)	Finalización, físico correcto	Pase deficiente → sustituir por delantero más ágil
Jonas Hofmann (ST)	Técnica, visión, posicionamiento	Puede actuar como segundo punta o falso 9
Recomendaciones generales		
<ul style="list-style-type: none"> ▪ Mantener: Neuer, Raum, Emre Can, Pedri, Nico Williams, Iñaki, Hofmann. ▪ Cambiar: Sustituir a Götze por un MC físico (ej. Laimer, Gravenberch, Wirtz IF). ▪ Ajustar: Buscar un ST más adecuado para presión alta (ej. Werner, Adeyemi, Thuram). 		

Tabla. 5.5: Evaluación resumida de la plantilla (4-4-2 Presión alta, 20000 monedas)

Posesión

La evaluación generada por el asistente para la plantilla cuyo estilo de juego se basa en mantener la posesión se presenta en la Tabla 5.6.

Jugador	Fortalezas	Debilidades / Recomendación
Stefan Ortega (GK)	Pase (86), posicionamiento sólido	Ideal para salida en corto, mantener
Juan Miranda (LB)	Ritmo, apoyo ofensivo decente	Mejorable defensivamente; buscar opción más sólida si posible
Manuel Akanji (CB)	Completo, pase corto/largo excelente	Central ideal para posesión, mantener
Matthijs de Ligt (CB)	Defensa top, pase largo correcto	Lento (62), puede quedar expuesto si la línea sube
Lucas Vázquez (RB)	Muy ofensivo, buena técnica	Requiere instrucciones defensivas conservadoras
Luka Modrić (CM)	Pase, visión, control bajo presión	Baja stamina (76), cuidar esfuerzos
Mateo Kovačić (CM)	Técnica completa, control, pase	Perfecto complemento de Modrić, mantener
Sávio (LM)	Ritmo, regate, desequilibrio 1v1	No contribuye a circulación, débil en defensa; buscar alternativa más asociativa
Nico Williams (RM)	Velocidad, desborde, stamina decente	Poca visión, poco pase posicional. Similar problema que Sávio
Iñaki Williams (ST)	Velocidad brutal, ideal al espacio	Pase bajo, mala combinación. Mejor como segundo delantero o re-ulsivo
Marcus Rashford (ST)	Finalización, regate, algo mejor pase que Iñaki	Stamina baja (68), tampoco es un jugador asociativo
Recomendaciones generales		
<ul style="list-style-type: none"> ■ Mantener: Doble pivote Modrić + Kovačić, Akanji, Ortega. ■ Cambiar: Al menos uno de los extremos por un jugador con mejor pase/visión (ej.: Bernardo Silva, Foden, Ødegaard). ■ Ajustar: Instrucciones conservadoras a laterales, “volver a defender” a extremos. Buscar delantero con buen pase si se mantiene el 4-4-2 o considerar cambiar a 4-2-3-1. 		

Tabla. 5.6: Evaluación resumida de la plantilla (4-4-2 Posesión, 20000 monedas)

5.3. Creación de Equipos

A partir de la plantilla sugerida por el sistema para implementar un estilo de juego basado en la posesión del balón, se llevará a cabo la aplicación manual de todas las recomendaciones proporcionadas por el asistente, utilizando para ello la interfaz disponible

5.3.1. Configurar Posiciones

Una de las principales sugerencias consiste en sustituir a uno de los extremos por un jugador que presente mejores atributos en pase o visión de juego, con el objetivo de favorecer la circulación del balón y mantener el control del partido.

Elección del extremo que vamos a sustituir

Primeramente, debemos comparar los dos jugadores utilizando la interfaz gráfica y el asistente automáticamente nos dará su opinión (ver Figura 5.4).

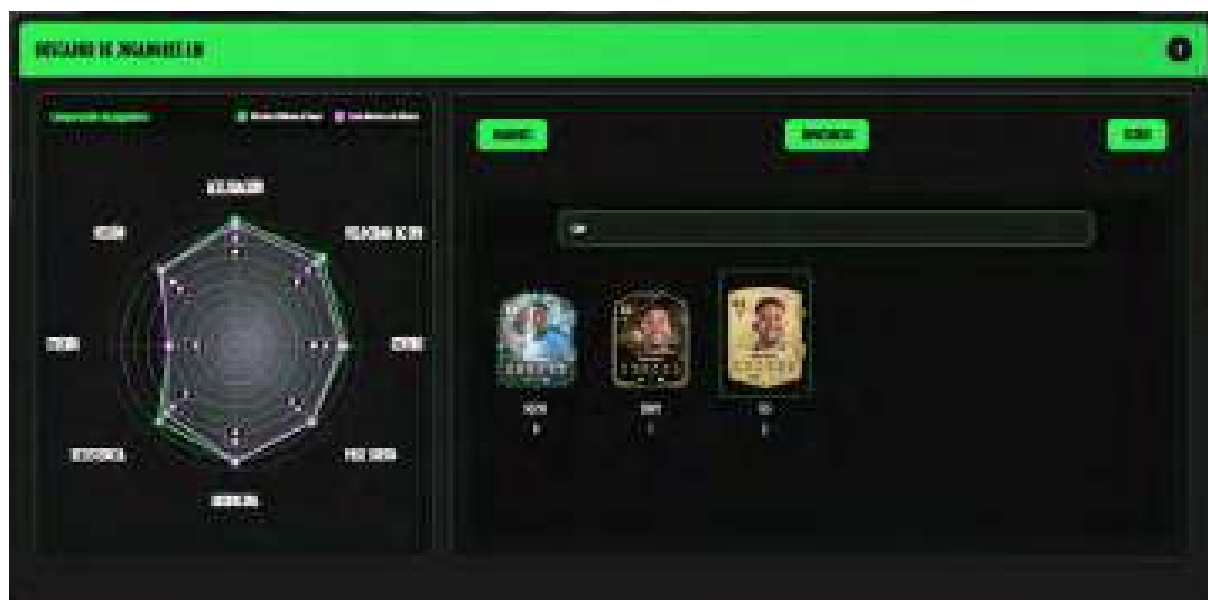


Figura 5.4: Comparación gráfica de Nico y Sávio para un sistema de posesión

Atributo clave	Nicholas Williams	Sávio Moreira
Aceleración / Sprint	94 / 93	90 / 87
Stamina	81	73
Dribbling total	88	87
Agilidad / Balance	92 / 84	88 / 83
Control de balón	85	84
Centros	83	80
Pase corto	82	81
Visión	80	79
Roles tácticos	Winger, Inside Forward	No definidos

Tabla. 5.7: Comparativa técnica entre Nicholas Williams y Sávio Moreira (LM en formación 4-4-2)

Opinión del asistente

La opinión del asistente se resume en que: para la posición de LM en una 4-4-2, **Nicholas Williams** es claramente superior a **Sávio Moreira** en velocidad, resistencia, regate 1v1 y adaptación táctica. Esto lo convierte en una opción más efectiva y fiable durante los 90 minutos.

Por lo tanto, vamos a intentar sustituir a Sávio Moreira. Siguiendo las indicaciones del asistente, vamos a intentar fichar a un extremo destinado específicamente a mantener la posesión, priorizando la visión y el pase.

Sustitución de Sávio Moreira

Para sustituirlo, primeramente nos dirigimos a la ventana de comparación de jugadores de su posición haciendo click sobre su carta (ver Figura 5.5).

Las prioridades que configuró el asistente automáticamente son las siguientes:



Figura 5.5: Prioridades configuradas por el asistente para la posición de LM

Debemos aumentar la prioridad de los atributos visión, pase, dribbling, agilidad y reducir la prioridad de la aceleración, la velocidad de sprint. De lo contrario, el modelo no reflejará correctamente nuestra preferencia por los primeros atributos. Además, deberemos situar la química como atributo más importante, ya que queremos mantener la sinergia del equipo.

Disponemos de los siguientes candidatos:

El candidato más interesante parece ser Matheus Nunes, ya que es el que no tiene un ritmo demasiado bajo. Pero al compararlo nos damos cuenta de lo siguiente:

Opinión del asistente comparando a Sávio Moreira con Matheus Nunes

- **Ritmo y stamina:** Sávio (88 ritmo, 73 stamina) domina claramente sobre Denis (60 ritmo, 55 stamina).
- **Técnica:** Denis tiene mejor pase largo, pero Sávio es superior en pase corto, dribbling y agilidad.
- **Conclusión:** Sávio es claramente mejor opción como LM ofensivo. Denis, solo como suplente puntual.

Podemos intentar bajar la química para ver si existen jugadores que aunque no nos permitan tener el 33 de química, nos saquen del apuro sin tener que aumentar el presupuesto.

Encontramos dos candidatos interesantes:

- **Paula Fernández Jiménez:** seleccionando esta jugadora la química del equipo desciende a 31.

Atributo	Peso (%)	Sávio	Paula	Mejor
Visión	100	79	81	Paula
Pase largo	100	70	80	Paula
Pase corto	66	81	85	Paula
Dribbling	62	87	80	Sávio
Agilidad	46	90	73	Sávio
Stamina	60	73	89	Paula
Aceleración	20	90	76	Sávio
Velocidad punta	20	87	77	Sávio
TOTAL	-	4 atributos	4 atributos	Empate (depende del estilo)

Tabla. 5.8: Comparativa LM: Paula Fernández Jiménez vs Sávio Moreira de Oliveira

Opinión del asistente comparando a Sávio Moreira con Matheus Nunes

- **Construcción y posesión:** Paula (mejor pase largo, visión y stamina).
- **Transiciones rápidas y desborde:** Sávio (mejor dribbling, agilidad y ritmo).
- **Conclusión:** Paula si juegas a posesión o presión alta; Sávio si priorizas verticalidad y contraataques.

- **Thomas Lemar:** seleccionando a este jugador, la química del equipo desciende a 32.

Atributo Clave	Lemar	Sávio	Ganador
Visión / Pase largo	81 / 78	79 / 70	Lemar
Pase corto	81	81	Empate
Dribbling / Agilidad	83 / 90	87 / 90	Sávio
Ritmo (Aceleración/Sprint)	85 / 77	90 / 87	Sávio
Stamina	77	73	Lemar
Estilo de química ideal	Maestro/Artista	Motor/Cazador	Depende
TOTAL	3 factores	2 factores	Depende del estilo

Tabla. 5.9: Comparativa LM: Thomas Lemar vs Sávio Moreira de Oliveira

Opinión del asistente comparando a Sávio Moreira con Thomas Lemar

- **Para construcción y posesión:** Lemar (mejor visión, pase largo y stamina).
- **Para transiciones rápidas y desborde:** Sávio (mejor dribbling y ritmo total).
- **Estilos de química recomendados:** Lemar → *Maestro / Artista*; Sávio → *Motor / Cazador*.
- **Conclusión:**
 - **Lemar** → posesión, distribución desde banda, apoyo defensivo.
 - **Sávio** → transiciones rápidas, desequilibrio en banda.

Por último nos quedaría comparar esas dos opciones, para ver que jugador es mejor para nuestro equipo:

Atributo	Lemar	Paula	Ganador
Visión (100 %)	81	81	Empate
Pase largo (100 %)	78	80	Paula
Pase corto (66 %)	81	85	Paula
Dribbling (62 %)	83	80	Lemar
Agilidad (46 %)	90	73	Lemar
Stamina (60 %)	77	89	Paula
Aceleración (20 %)	85	76	Lemar
Sprint Speed (20 %)	77	77	Empate
Química (12 %)	–	–	No disponible
Total	4 atributos	3 atributos	Depende del estilo

Tabla. 5.10: Comparativa LM: Thomas Lemar vs Paula Fernández (Formación 4-4-2)

Opinión del asistente comparando a Paula Fernández con Thomas Lemar

- **Perfil ofensivo creativo y desequilibrante: Thomas Lemar**
- **Perfil conservador y resistente: Paula Fernández**
- **Estilo de química recomendado:**
 - Lemar → *Motor (Engine), Cazador (Hunter)*
 - Paula → *Sombra (Shadow), Motor (Engine)*
- **Recomendación final:**
 - **Para una 4-4-2 ofensiva por bandas → Lemar**
 - **Para solidez y aguante → Paula**

Si ese punto o esos puntos perdidos de química nos parecen demasiado, debemos plantearnos entonces una ampliación de presupuesto, orientada a mejorar esa posición exclusivamente. Y no es hasta que aumentamos el presupuesto a 27000 monedas, que aparece una candidata mejor que nuestro titular actual.

Finalmente, comparamos a Lemar con nuestra nueva candidata Hemp para comprobar si nos merece la pena la ampliación de presupuesto:

Atributo	Peso (%)	Lemar	Hemp	Ganador
Pase corto (Short Pass)	33	81	81	Empate
Dribbling	31	83	87	Hemp
Agilidad (Agility)	23	90	90	Empate
Stamina	30	77	88	Hemp
Aceleración	10	85	91	Hemp
Velocidad (Sprint)	10	77	90	Hemp
Visión (Vision)	50	81	83	Hemp
Pase largo (Long Pass)	50	78	79	Hemp
Total				Hemp (6-2)

Tabla. 5.11: Comparativa LM: Lauren Hemp vs Thomas Lemar (Formación 4-4-2)

Opinión del asistente comparando a Lauren Hemp con Thomas Lemar

- **Conclusión principal:** Lauren Hemp es claramente superior para el rol de LM ofensivo en 4-4-2.
- **Ventajas:** Mayor dribbling, stamina, aceleración y velocidad punta.
- **Estilo de química recomendado:** *Motor (Engine)* → mejora ritmo, pase y regate.
- **Consideración táctica:** Si Lemar tiene enlaces de química significativamente mejores, puede ser alternativa puntual.
- **Recomendación final:** **Hemp** es la mejor opción si la química lo permite.

Evaluación de la plantilla tras realizar la sustitución

Solicitamos al asistente que evalúe la plantilla para ver si hemos corregido el problema incorporando a Lauren Hemp (ver Figura 5.6).

Evaluación por parte del asistente de la plantilla 4-4-2 después de realizar cambios

Extremos (LM: Hemp / RM: Nico Williams)

- Roles: Inside Forward / Winger
- Pase corto: Hemp (81), Nico (82)
- Visión: Hemp (83), Nico (80)
- Dribbling: ambos 87+
- Stamina: Hemp (88), Nico (81)
- Compostura: Hemp (70), Nico (78)

Conclusiones:

- Muy buenos para mantener amplitud y combinar.
- Defensa limitada: Nico (def. awareness 36), Hemp (56).



Figura 5.6: Plantilla para jugar al contraataque con la formación 4-4-2 (27000 monedas) después de realizar modificaciones manuales

Capítulo 6

CONCLUSIONES

En este capítulo se presentan las conclusiones extraídas a lo largo del desarrollo del proyecto, con el objetivo de realizar una valoración final tanto desde una perspectiva técnica como personal. Asimismo, se proponen diversas líneas de trabajo futuro que permitirían ampliar o mejorar los resultados obtenidos, fomentando la continuidad y evolución de esta propuesta.

6.1. Valoración Final del Proyecto

El sistema FC Assistant ha superado con creces todos los objetivos inicialmente planteados, cumpliendo de manera satisfactoria tanto con los requisitos funcionales como con los no funcionales establecidos. Esta solución ofrece beneficios tangibles a los usuarios, los cuales no se habían materializado en propuestas previas, y contribuye significativamente a uno de los grandes retos actuales en el ámbito de la informática: lograr sistemas accesibles, adaptables al usuario y explicables en su funcionamiento.

Desde una perspectiva personal, la realización de este proyecto ha constituido una experiencia profundamente enriquecedora para mi desarrollo como ingeniero informático. Ha puesto a prueba los conocimientos adquiridos a lo largo de la carrera en áreas como los sistemas de información web, los sistemas de apoyo a la decisión, la minería web, los sistemas de recuperación de información, las bases de datos distribuidas, las metaheurísticas y la concurrencia, entre otros, todos ellos aplicados en un caso práctico de gran complejidad. Esta experiencia me ha permitido comprender, de forma directa, las dificultades inherentes al desarrollo de una solución partiendo desde cero, con el objetivo de generar un impacto real en la sociedad.

En definitiva, este proyecto ha supuesto un verdadero punto de inflexión en mi formación, consolidando mi vocación por el desarrollo de soluciones tecnológicas útiles, accesibles y orientadas a mejorar la experiencia de los usuarios.

6.2. Trabajos Futuros

El presente proyecto abre la puerta a múltiples líneas de trabajo que podrían desarrollarse en fases posteriores. A continuación, se proponen algunas de las posibles extensiones y mejoras:

- *Adaptación de la interacción con el asistente para personas con discapacidad intelectual.* En la actualidad, resulta fundamental diseñar sistemas accesibles para personas neurodivergentes, especialmente considerando el progresivo envejecimiento de la población y el incremento asociado de trastornos cognitivos. La incorporación de interacción asistida puede mejorar notablemente la experiencia y el acceso de estos usuarios.
- *Diseño específico para público infantil.* Si bien el diseño actual resulta funcional y adecuado, la intervención de especialistas en pedagogía o educación infantil permitiría adaptar la interfaz y la experiencia de usuario a las necesidades y capacidades cognitivas de los niños, mejorando así su eficacia y usabilidad.
- *Implementación de un modelo de negocio y sistema de monedas virtuales (Assistant Coins).* Este componente permitiría introducir elementos de gamificación, incentivar el uso continuado del asistente y abrir nuevas posibilidades de monetización.
- *Integración de modelos de simulación de partidos.* Incorporar simulaciones realistas permitiría analizar la interacción entre jugadores y mejorar la capacidad predictiva del asistente, proporcionando recomendaciones más precisas y fundamentadas en el comportamiento esperado durante el juego.
- *Aplicación de modelos específicos para el análisis de series temporales.* El uso de modelos avanzados en este ámbito permitiría realizar predicciones más fiables sobre la evolución del valor de los jugadores, identificando posibles alzas o descensos de precio en función de patrones históricos y variables contextuales.
- *Mejora de la interacción con el asistente.* Actualmente, la interacción con el asistente es satisfactoria y este reconoce como contexto el equipo que se está formando. No obstante, sería deseable que el asistente alcanzara un mayor grado

de autonomía y comprensión contextual, permitiéndole actuar de forma proactiva y coherente con la situación.

- *Implementación de un sistema de gestión de usuarios y analítica web.* Se propone desarrollar un sistema que permita la gestión de usuarios, así como incorporar mecanismos para la captura y estudio de datos analíticos con el fin de comprender mejor el comportamiento de los usuarios y optimizar la experiencia de uso.

Bibliografía

- [1] Marco Dorigo and Luca Maria Gambardella. Ant colonies for the travelling salesman problem. *biosystems*, 43(2):73–81, 1997.
- [2] Sapna Katiyar, N Ibraheem, and Abdul Quaiyum Ansari. Ant colony optimization: a tutorial review. In *National conference on advances in power and control*, pages 99–110, 2015.
- [3] M. Sánchez. La industria del videojuego, ganador sector entretenimiento, s.f. URL <https://www.msanchez.es/la-industria-del-videojuego-ganador-sector-entretenimiento>.
- [4] Audiencia anual de esports a nivel mundial, s.f. URL <https://es.statista.com/estadisticas/711557/audiencia-anual-de-esports-a-nivel-mundial/>.
- [5] SomosXbox. El simulador de fútbol definitivo ahora cuesta un 30 URL <https://www.somosxbox.com/el-simulador-de-futbol-definitivo-ahora-cuesta-un-30-menos-consigue-ea-sports-fc-25-en-oferta/>.
- [6] Xataka. Llevo 20 años tocando videojuego de fútbol y he probado EA Sports FC 25: esta ha sido mi experiencia, 2024. URL <https://www.xataka.com/videojuegos/llevo-20-anos-tocar-videojuego-futbol-he-probado-ea-sports-fc-25-esta-ha-sido-mi-experiencia>.
- [7] The game awards 2024 - best sports/racing, 2024. URL <https://thegameawards.com/nominees/best-sports-racing>.
- [8] Sakshi Sachdeva. Scrum methodology. *Int. J. Eng. Comput. Sci*, 5(16792):16792–16800, 2016.
- [9] Janet Kuhn. Decrypting the moscow analysis. *The workable, practical guide to Do IT Yourself*, 5, 2009.
- [10] Ministerio de Trabajo y Economía Social. Resolución de 17 de noviembre de 2023, de la dirección general de trabajo, por la que se registra y publica el

- convenio colectivo estatal de empresas de consultoría, tecnologías de la información y estudios de mercado y de la opinión pública, 2023. URL <https://www.boe.es/boe/dias/2023/11/29/pdfs/BOE-A-2023-24388.pdf>. BOE núm. 285, de 29 de noviembre de 2023.
- [11] J. Hernández Moreno. Fútbol: concepto e investigación. *EFDeportes*, 148, 1993. URL <https://efdeportes.com/efd148/futbol-concepto-e-investigacion.htm>.
- [12] Roukaya Ben Jeddou. *Football Selection Optimization through the Integration of Management Theories, AI and Multi-criteria Decision Making*. PhD thesis, Université Bourgogne Franche-Comté, 2024.
- [13] Zhiqiang Pu, Yi Pan, Shijie Wang, Boyin Liu, Min Chen, Hao Ma, and Yixiong Cui. Orientation and decision-making for soccer based on sports analytics and ai: A systematic review. *IEEE/CAA Journal of Automatica Sinica*, 11(1):37–57, 2024.
- [14] Milton Rivas Borbón and Erick Sánchez Alvarado. Guía didáctica del curso: Táctica y estrategia en futbol. *MHSalud: Movimiento Humano y Salud*, 10(1):6–40, 2013.
- [15] FIFA U Team. Todos los atributos en fifa 21, 2020. URL <https://es.fifauteam.com/atributos-en-fifa-21/>. Accedido: 20 de junio de 2025.
- [16] Claudia López de Munain, Marcela Torrent, Florencio Molina, Sebastián Schanz, Anahi Sandoval, Juan Pablo Sfiligoy, and Marcos Litterio Sassaroli. Sistemas de apoyo a la toma de decisiones. In *XVI Workshop de Investigadores en Ciencias de la Computación*, 2014.
- [17] Hamed Taherdoost and Atefeh Mohebi. Using smart method for multi-criteria decision making: Applications, advantages, and limitations. *Archives of Advanced Engineering Science*, 2(4):190–197, 2024. doi: 10.47852/bonviewAAES42022765. Research Article.
- [18] Robert H. Bonczek, Clyde W. Holsapple, and Andrew B. Whinston. *Foundations of Decision Support Systems*. Academic Press, New York, 1981. ISBN 0-12-113050-9.
- [19] Ching-Lai Hwang, Kwangsun Yoon, Ching-Lai Hwang, and Kwangsun Yoon. Methods for multiple attribute decision making. *Multiple attribute decision making: methods and applications a state-of-the-art survey*, pages 58–191, 1981.
- [20] Pengyu Chen. Effects of normalization on the entropy-based topsis method. *Expert Systems with Applications*, 136:33–41, 2019.

- [21] Eiram Turban. *Jay E. Aronson Decision Support System and Intelligent Systems*. Prentice Hall, 2001.
- [22] Daniel J. Power. Specifying an expanded framework for classifying and describing decision support systems. *Communications of the Association for Information Systems*, 13:158–166, 2004. doi: 10.17705/1CAIS.01313.
- [23] Ralph Kimball, Margy Ross, Bob Becker, Joy Mundy, and Warren Thornthwaite. *Kimball's Data Warehouse Toolkit Classics: The Data Warehouse Toolkit, ; The Data Warehouse Lifecycle, ; The Data Warehouse ETL Toolk*. Wiley Publishing, 2009.
- [24] Edgar F Codd. Providing olap (on-line analytical processing) to user-analysts: An it mandate. <http://www.arborsoft.com/papers/coddTOC.html>, 1993.
- [25] Jacek Boroń. Application of ms excel® solver in solving structural and management problems.
- [26] Jay W Forrester. Industrial dynamics: a major breakthrough for decision makers. In *The Roots of Logistics*, pages 141–172. Springer, 2012.
- [27] MA Kramers, CGM Conijn, and C Bastiaansen. Exsys, an expert system for diagnosing flowerbulb diseases, pests and non-parasitic disorders. *Agricultural systems*, 58(1):57–85, 1998.
- [28] Bill Cope and Mary Kalantzis. A little history of e-learning: finding new ways to learn in the plato computer education system, 1959–1976. *History of Education*, 52(6):905–936, 2023.
- [29] JR JAY F NUNAMAKER. Future research in group support systems: needs, some questions and possible directions. *International Journal of Human-Computer Studies*, 47(3):357–385, 1997.
- [30] R Brent Gallupe, Gerardine DeSanctis, and Gary W Dickson. The impact of computer-based support on the process and outcomes of group decision making. 1986.
- [31] Aleem Akhtar. Evolution of ant colony optimization algorithm—a brief literature review. *arXiv preprint arXiv:1908.08007*, 2019.
- [32] M. Dorigo, V. Maniezzo, and A. Colorni. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 26(1):29–41, 1996. doi: 10.1109/3477.484436.

- [33] Giuseppe Lugano. Virtual assistants and self-driving cars. In *2017 15th international conference on ITS telecommunications (ITST)*, pages 1–5. IEEE, 2017.
- [34] Reza Rawassizadeh, Taylan Sen, Sunny Jung Kim, Christian Meurisch, Hamidreza Keshavarz, Max Mühlhäuser, and Michael Pazzani. Manifestation of virtual assistants and robots into daily life: Vision and challenges. *CCF Transactions on Pervasive Computing and Interaction*, 1:163–174, 2019.
- [35] José Miguel Palacios Bayas, Victor Alejandro Bosquez Barcenas, and Ángel Ramiro Palacios Bayas. Integración de un asistente virtual en ambientes de vida asistida por computador para personas con discapacidad física. *Revista de Investigación Talentos*, 7(1):48–61, 2020.
- [36] José M Rubio, Tanya Neira-Peña, Danilo Molina, and Cristian Vidal-Silva. Proyecto ubot: asistente virtual para entornos virtuales de aprendizaje. *Información tecnológica*, 33(4):85–92, 2022.
- [37] Juan Pablo Crespo Obaco and Jeovanny Benavides Bailón. Beneficios y desafíos de los asistentes virtuales en el aprendizaje: Benefits and challenges of virtual assistants in learning. *LATAM Revista Latinoamericana de Ciencias Sociales y Humanidades*, 5(2):685–700, 2024.
- [38] Guido Trujillo Valdiviezo, Desmond Mejía Ayala, and Lino Rodríguez Alegre. Los asistentes virtuales y la calidad de servicio al cliente. *TECHNO REVIEW. International Technology, Science and Society Review/Revista Internacional de Tecnología, Ciencia y Sociedad*, 13(4):1–10, 2023.
- [39] Joseph Weizenbaum. Eliza—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45, 1966.
- [40] Luke Swartz. *Why people hate the paperclip: Labels, appearance, behavior, and social responses to user interface agents*. PhD thesis, Citeseer, 2003.
- [41] Francisco Javier Martínez Méndez. *Recuperación de información: modelos, sistemas y evaluación*. JMC Kiosko., 2004.
- [42] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020.

- [43] Jisoo Jang and Wen-Syan Li. Au-rag: Agent-based universal retrieval augmented generation. In *Proceedings of the 2024 Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region*, pages 2–11, 2024.
- [44] Aditi Singh, Abul Ehtesham, Saket Kumar, and Tala Talaei Khoei. Agentic retrieval-augmented generation: A survey on agentic rag. *arXiv preprint arXiv:2501.09136*, 2025.
- [45] Alan Cooper. *The inmates are running the asylum*. Springer, 1999.
- [46] Kobe W. Does index use b-tree or b+ tree?, 2023. URL <https://www.mongodb.com/community/forums/t/does-index-use-btree-or-b-tree/248053>. MongoDB Developer Community Forum.
- [47] Kenneth Enevoldsen, Isaac Chung, Imene Kerboua, Márton Kardos, Ashwin Mathur, David Stap, Jay Gala, Wissam Sibli, Dominik Krzemiński, Genta Indra Winata, Saba Sturua, Saiteja Utpala, Mathieu Ciancone, Marion Schaeffer, Gabriel Sequeira, Diganta Misra, Shreeya Dhakal, Jonathan Rystrom, Roman Solomatin, Ömer Çağatan, Akash Kundu, Martin Bernstorff, Shitao Xiao, Akshita Sukhlecha, Bhavish Pahwa, Rafał Poświata, Kranthi Kiran GV, Shawon Ashraf, Daniel Auras, Björn Plüster, Jan Philipp Harries, Loïc Magne, Isabelle Mohr, Mariya Hendriksen, Dawei Zhu, Hippolyte Gisserot-Boukhlef, Tom Aarsen, Jan Kostkan, Konrad Wojtasik, Taemin Lee, Marek Šuppa, Crystina Zhang, Roberta Rocca, Mohammed Hamdy, Andrianos Michail, John Yang, Manuel Faysse, Aleksei Vatolin, Nandan Thakur, Manan Dey, Dipam Vasani, Pranjal Chitale, Simone Tedeschi, Nguyen Tai, Artem Snegirev, Michael Günther, Mengzhou Xia, Weijia Shi, Xing Han Lù, Jordan Clive, Gayatri Krishnakumar, Anna Maksimova, Silvan Wehrli, Maria Tikhonova, Henil Panchal, Aleksandr Abramov, Malte Ostendorff, Zheng Liu, Simon Clematide, Lester James Miranda, Alena Fenogenova, Guangyu Song, Ruqiya Bin Safi, Wen-Ding Li, Alessia Borghini, Federico Cassano, Hongjin Su, Jimmy Lin, Howard Yen, Lasse Hansen, Sara Hooker, Chenghao Xiao, Vaibhav Adlakha, Orion Weller, Siva Reddy, and Niklas Muenighoff. Mmteb: Massive multilingual text embedding benchmark, 2025. URL <https://arxiv.org/abs/2502.13595>.

Apéndice A

MANUAL DE INSTALACIÓN

Este apéndice constituye una guía orientativa para la puesta en marcha del sistema. Comienza con la instalación del software común a todos los entornos y, posteriormente, se centra en la configuración específica de cada uno de los componentes.

A.1. Software Común

En esta sección se describen los procedimientos necesarios para instalar el conjunto de herramientas y entornos que comparten la mayoría de los equipos implicados en la ejecución del sistema. Estas herramientas proporcionan el entorno base sobre el cual se ejecuta el resto de componentes.

A.1.1. Windows Subsystem for Linux (WSL)

En caso de disponer de un sistema operativo Windows, es posible aprovechar la funcionalidad de WSL. WSL es una característica de Windows que permite ejecutar un entorno GNU/Linux completo de manera nativa, sin necesidad de una máquina virtual o un arranque dual.

La instalación puede realizarse fácilmente desde la Microsoft Store¹ o desde la página oficial de Ubuntu para WSL². Una vez instalado, se recomienda comprobar la versión del sistema operativo Linux instalado mediante el siguiente Código A.1.

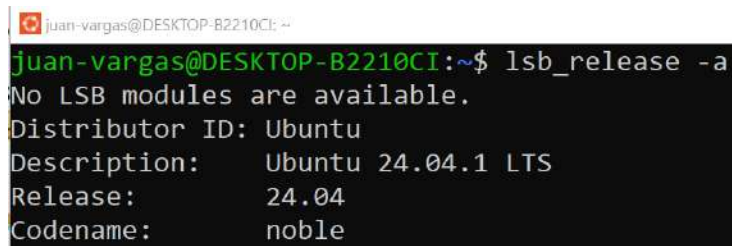
¹<https://apps.microsoft.com/detail/9pdxgncfsczv?hl=es-ES&gl=ES>

²<https://ubuntu.com/desktop/wsl>

```
1 lsb_release -a
```

Listado A.1: Consulta de la versión instalada de Ubuntu en WSL

La salida del comando anterior permitirá verificar que se está utilizando la versión deseada de Ubuntu, como se ilustra en la Fig. A.1.



```
juan-vargas@DESKTOP-B2210CI: ~$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 24.04.1 LTS
Release:        24.04
Codename:       noble
```

Figura A.1: Consulta de la versión de Ubuntu instalada en WSL

A.1.2. Visual Studio Code

Para instalar Visual Studio Code en Ubuntu, se deben ejecutar los comandos indicados en el fragmento de código Código A.2.

```
1 sudo apt update
2 sudo apt install wget gpg -y
3 wget -qO- https://packages.microsoft.com/keys/microsoft.asc | gpg --dearmor > microsoft.
  gpg
4 sudo install -o root -g root -m 644 microsoft.gpg /etc/apt/trusted.gpg.d/
5 sudo sh -c 'echo "deb [arch=amd64] https://packages.microsoft.com/repos/vscode stable
  main" > /etc/apt/sources.list.d/vscode.list'
6 sudo apt update
7 sudo apt install code
```

Listado A.2: Comandos para instalar Visual Studio Code en Ubuntu desde terminal

Una vez instalado, puede ejecutarse simplemente mediante el comando siguiente (ver Código A.3):

```
1 code
```

Listado A.3: Comando para ejecutar Visual Studio Code

A.1.3. Node.js y npm

Para instalar Node.js y npm en Ubuntu se puede utilizar el siguiente comando (ver Código A.4):

```
1 sudo apt install nodejs npm -y
```

Listado A.4: Instalación de Node.js y npm en Ubuntu

Una vez completada la instalación, es recomendable verificar que ambas herramientas se han instalado correctamente consultando sus versiones respectivas, como se muestra en Código [A.5](#):

```
1 node -v  
2 npm -v
```

Listado A.5: Comprobación de versiones instaladas de Node.js y npm

A.1.4. Python

Para instalar Python 3 y su gestor de paquetes `pip`, se deben ejecutar los comandos mostrados en el fragmento Código [A.6](#).

```
1 sudo apt update  
2 sudo apt install python3 python3-pip -y
```

Listado A.6: Comandos para instalar Python 3 y pip en Ubuntu

Tras la instalación, es posible comprobar la versión instalada tanto de Python como de pip mediante los comandos indicados en el fragmento Código [A.7](#).

```
1 python3 --version  
2 pip3 --version
```

Listado A.7: Verificación de versiones de Python 3 y pip

A.2. MRAD

A.2.1. Base de Datos Distribuida

Cabe recordar que, durante el diseño de este componente, se especificó que su simulación se llevaría a cabo mediante Docker. Por lo tanto, una vez seleccionado el equipo adecuado, es necesario proceder con la instalación de Docker en dicho equipo. El comando para instalar Docker en Ubuntu se presenta en el Código [A.8](#).

```
1 sudo apt update
2 sudo apt install docker.io -y
3 sudo systemctl start docker
4 sudo systemctl enable docker
```

Listado A.8: Comandos para instalar Docker en Ubuntu

Una vez instalado Docker, nos descargamos la imagen más reciente de MongoDB con el comando que aparece en el Código [A.9](#).

```
1 sudo docker pull mongo:latest
```

Listado A.9: Descargar imagen oficial de MongoDB

Ahora creamos una red para el clúster MongoDB, con un rango de subred definido, como se muestra en el Código [A.10](#).

```
1 docker network create clusternet --subnet=192.168.0.0/16
```

Listado A.10: Crear red personalizada para Docker

A continuación, procedemos a crear tres servidores de configuración para MongoDB, cambiando el nombre del contenedor y mapeando los puertos apropiadamente. Estos comandos se muestran en el Código [A.11](#).

```
1 docker run -d -it --net clusternet --name mongoserver1 -p 27101:27017 mongodb/mongodb-
  community-server mongod --port 27017 --configsvr --replSet replicasetbdd
2 docker run -d -it --net clusternet --name mongoserver2 -p 27102:27017 mongodb/mongodb-
  community-server mongod --port 27017 --configsvr --replSet replicasetbdd
3 docker run -d -it --net clusternet --name mongoserver3 -p 27103:27017 mongodb/mongodb-
  community-server mongod --port 27017 --configsvr --replSet replicasetbdd
```

Listado A.11: Crear servidores de configuración MongoDB

Una vez arrancados los servidores de configuración, inicializamos el replica set desde cualquiera de estos contenedores con el siguiente comando MongoDB, mostrado en el Código [A.12](#).

```
1 rs.initiate({
2   _id: "replicasetbdd",
3   members: [
4     { _id: 0, host: "mongoserver1:27017" },
5     { _id: 1, host: "mongoserver2:27017" },
6     { _id: 2, host: "mongoserver3:27017" }
7   ]
8 })
```

Listado A.12: Inicializar replica set para servidores de configuración

Seguidamente, creamos las máquinas nodos de shards, cambiando nombres y puertos para cada nodo, como se muestra en el Código A.13.

```
1 docker run -d --net clusternet --name mongoshard1nodo1 -p 27111:27017 mongodb/mongodb-
  community-server mongod --port 27017 --shardsvr --replSet replicasetshard1
2 docker run -d --net clusternet --name mongoshard1nodo2 -p 27121:27017 mongodb/mongodb-
  community-server mongod --port 27017 --shardsvr --replSet replicasetshard1
3 docker run -d --net clusternet --name mongoshard1nodo3 -p 27131:27017 mongodb/mongodb-
  community-server mongod --port 27017 --shardsvr --replSet replicasetshard1
4
5 docker run -d --net clusternet --name mongoshard2nodo1 -p 27112:27017 mongodb/mongodb-
  community-server mongod --port 27017 --shardsvr --replSet replicasetshard2
6 docker run -d --net clusternet --name mongoshard2nodo2 -p 27122:27017 mongodb/mongodb-
  community-server mongod --port 27017 --shardsvr --replSet replicasetshard2
7 docker run -d --net clusternet --name mongoshard2nodo3 -p 27132:27017 mongodb/mongodb-
  community-server mongod --port 27017 --shardsvr --replSet replicasetshard2
8
9 docker run -d --net clusternet --name mongoshard3nodo1 -p 27113:27017 mongodb/mongodb-
  community-server mongod --port 27017 --shardsvr --replSet replicasetshard3
10 docker run -d --net clusternet --name mongoshard3nodo2 -p 27123:27017 mongodb/mongodb-
  community-server mongod --port 27017 --shardsvr --replSet replicasetshard3
11 docker run -d --net clusternet --name mongoshard3nodo3 -p 27133:27017 mongodb/mongodb-
  community-server mongod --port 27017 --shardsvr --replSet replicasetshard3
12
13 docker run -d --net clusternet --name mongoshard4nodo1 -p 27114:27017 mongodb/mongodb-
  community-server mongod --port 27017 --shardsvr --replSet replicasetshard4
14 docker run -d --net clusternet --name mongoshard4nodo2 -p 27124:27017 mongodb/mongodb-
  community-server mongod --port 27017 --shardsvr --replSet replicasetshard4
15 docker run -d --net clusternet --name mongoshard4nodo3 -p 27134:27017 mongodb/mongodb-
  community-server mongod --port 27017 --shardsvr --replSet replicasetshard4
16
17 docker run -d --net clusternet --name mongoshard5nodo1 -p 27115:27017 mongodb/mongodb-
  community-server mongod --port 27017 --shardsvr --replSet replicasetshard5
18 docker run -d --net clusternet --name mongoshard5nodo2 -p 27125:27017 mongodb/mongodb-
  community-server mongod --port 27017 --shardsvr --replSet replicasetshard5
19 docker run -d --net clusternet --name mongoshard5nodo3 -p 27135:27017 mongodb/mongodb-
  community-server mongod --port 27017 --shardsvr --replSet replicasetshard5
20
21 docker run -d --net clusternet --name mongoshard6nodo1 -p 27116:27017 mongodb/mongodb-
  community-server mongod --port 27017 --shardsvr --replSet replicasetshard6
22 docker run -d --net clusternet --name mongoshard6nodo2 -p 27126:27017 mongodb/mongodb-
  community-server mongod --port 27017 --shardsvr --replSet replicasetshard6
23 docker run -d --net clusternet --name mongoshard6nodo3 -p 27136:27017 mongodb/mongodb-
  community-server mongod --port 27017 --shardsvr --replSet replicasetshard6
24
25 docker run -d --net clusternet --name mongoshard7nodo1 -p 27117:27017 mongodb/mongodb-
  community-server mongod --port 27017 --shardsvr --replSet replicasetshard7
26 docker run -d --net clusternet --name mongoshard7nodo2 -p 27127:27017 mongodb/mongodb-
  community-server mongod --port 27017 --shardsvr --replSet replicasetshard7
27 docker run -d --net clusternet --name mongoshard7nodo3 -p 27137:27017 mongodb/mongodb-
  community-server mongod --port 27017 --shardsvr --replSet replicasetshard7
```

Listado A.13: Crear nodos de shards MongoDB

Para cada conjunto de nodos shard, iniciamos el replica set correspondiente con los comandos MongoDB indicados en el Código A.14.

```
1 rs.initiate({
2   _id: "replicasetshard1",
3   members: [
4     { _id: 0, host: "mongoshard1nodo1:27017" },
5     { _id: 1, host: "mongoshard1nodo2:27017" },
6     { _id: 2, host: "mongoshard1nodo3:27017" }
7   ]
8 })
9 rs.initiate({
10  _id: "replicasetshard2",
11  members: [
12    { _id: 0, host: "mongoshard2nodo1:27017" },
13    { _id: 1, host: "mongoshard2nodo2:27017" },
14    { _id: 2, host: "mongoshard2nodo3:27017" }
15  ]
16 })
17 rs.initiate({
18  _id: "replicasetshard3",
19  members: [
20    { _id: 0, host: "mongoshard3nodo1:27017" },
21    { _id: 1, host: "mongoshard3nodo2:27017" },
22    { _id: 2, host: "mongoshard3nodo3:27017" }
23  ]
24 })
25 rs.initiate({
26  _id: "replicasetshard4",
27  members: [
28    { _id: 0, host: "mongoshard4nodo1:27017" },
29    { _id: 1, host: "mongoshard4nodo2:27017" },
30    { _id: 2, host: "mongoshard4nodo3:27017" }
31  ]
32 })
33 rs.initiate({
34  _id: "replicasetshard5",
35  members: [
36    { _id: 0, host: "mongoshard5nodo1:27017" },
37    { _id: 1, host: "mongoshard5nodo2:27017" },
38    { _id: 2, host: "mongoshard5nodo3:27017" }
39  ]
40 })
41 rs.initiate({
42  _id: "replicasetshard6",
43  members: [
44    { _id: 0, host: "mongoshard6nodo1:27017" },
45    { _id: 1, host: "mongoshard6nodo2:27017" },
46    { _id: 2, host: "mongoshard6nodo3:27017" }
47  ]
48 })
49 rs.initiate({
50  _id: "replicasetshard7",
51  members: [
52    { _id: 0, host: "mongoshard7nodo1:27017" },
53    { _id: 1, host: "mongoshard7nodo2:27017" },
54    { _id: 2, host: "mongoshard7nodo3:27017" }
55  ]
56 })
```

Listado A.14: Inicializar replica sets de shards

Seguidamente lanzamos un router Mongo que unirá toda la infraestructura, configurando el router para que utilice los servidores de configuración, como se muestra en el Código A.15.

```
1 docker run -d --net clusternet --name mongos -p 27017:27017 mongodb/mongodb-community-  
server mongos --configdb replicasetbdd/mongoserver1:27017,mongoserver2:27017,  
mongoserver3:27017 --bind_ip_all
```

Listado A.15: Lanzar router Mongo para el cluster

Finalmente, desde el router (mongos) podemos conectar clientes para realizar consultas y operaciones en la base de datos distribuida. Es importante añadir los shards al router para que se incluyan en el clúster con los comandos MongoDB que se muestran en el Código A.16.

```
1 sh.addShard("replicasetshard1/mongoshard1nodo1:27017,mongoshard1nodo2:27017,  
mongoshard1nodo3:27017")  
2 sh.addShard("replicasetshard2/mongoshard2nodo1:27017,mongoshard2nodo2:27017,  
mongoshard2nodo3:27017")  
3 sh.addShard("replicasetshard3/mongoshard3nodo1:27017,mongoshard3nodo2:27017,  
mongoshard3nodo3:27017")  
4 sh.addShard("replicasetshard4/mongoshard4nodo1:27017,mongoshard4nodo2:27017,  
mongoshard4nodo3:27017")  
5 sh.addShard("replicasetshard5/mongoshard5nodo1:27017,mongoshard5nodo2:27017,  
mongoshard5nodo3:27017")  
6 sh.addShard("replicasetshard6/mongoshard6nodo1:27017,mongoshard6nodo2:27017,  
mongoshard6nodo3:27017")  
7 sh.addShard("replicasetshard7/mongoshard7nodo1:27017,mongoshard7nodo2:27017,  
mongoshard7nodo3:27017")
```

Listado A.16: Añadir shards al router mongos

A.2.2. Conjunto de Scrappers

Primeramente debemos seleccionar una máquina que se encargue de realizar el webscraping para posteriormente ejecutar el algoritmo de scrapping en ella.

Una vez dispongamos de la máquina adecuada y el código, tenemos que instalar las bibliotecas necesarias para la ejecución del código, las cuales podemos instalar utilizando el Código A.17.

```
1 pip3 install pymongo requests lxml playwright
```

Listado A.17: Instalación de bibliotecas necesarias de Python

Finalmente solo tendremos que ejecutarlo con este comando, donde deberemos detallar la dirección IP de la base de datos, junto al nombre del host y el puerto de acceso como se puede ver en el Código A.18. Opcionalmente le podremos pasar el nombre de usuario y la contraseña.

```
1 python3 ./main.py --host [Direccion IP de la base de datos] --port [Puerto] --username [Nombre de usuario] --password[Contraseña]
```

Listado A.18: Ejecutar algoritmo de web scrapping

A.3. MGP

A.3.1. Servidor Web

Para comenzar, es necesario tener instalados Node.js y npm. Una vez instalados, se puede proceder a la instalación global de Angular CLI mediante el comando que se muestra en el Código A.19:

```
1 npm install -g @angular/cli
```

Listado A.19: Instalación de Angular CLI

A continuación, accedemos al directorio del proyecto y editamos el archivo `app-ctes.ts`, ubicado en la raíz del mismo, modificando el valor de la constante `API_URL` para que apunte a la URL correspondiente del servidor back-end.

Una vez dentro del proyecto, es necesario instalar todas las dependencias definidas en el archivo `package.json`. Este paso se lleva a cabo mediante la ejecución del

comando indicado en el Código [A.20](#):

```
1 npm install
```

Listado A.20: Instalación de dependencias del proyecto Angular

Posteriormente, arrancamos el servidor para que puedan visualizarlo en toda nuestra red con el comando que aparece en el Código [A.21](#).

```
1 ng serve --host 0.0.0.0 --allowed-host fcassistant.ddns.net
```

Listado A.21: Compilación del proyecto Angular en modo producción

A.3.2. Servidor Back-end

El servidor back-end ha sido desarrollado utilizando el framework NestJS, por lo que, al igual que el cliente, requiere tener instalados previamente Node.js y npm. Una vez configurado el entorno base, se recomienda instalar globalmente la interfaz de línea de comandos de NestJS mediante el comando que se muestra en el Código [A.22](#):

```
1 npm install -g @nestjs/cli
```

Listado A.22: Instalación de NestJS CLI

A continuación, se accede al directorio raíz del proyecto back-end y se procede a instalar las dependencias definidas en el archivo `package.json`, utilizando el comando indicado en el Código [A.23](#):

```
1 npm install
```

Listado A.23: Instalación de dependencias del servidor NestJS

Una vez instaladas las dependencias, se debe acceder al archivo `.env`, ubicado en la raíz del proyecto, donde se definen las constantes de configuración necesarias para el correcto funcionamiento del servidor. Un ejemplo representativo de dicho archivo se presenta en el Código [A.24](#):

```
1 APP_PORT=3000
2 DB_HOST=localhost
3 DB_PORT=27141
4 DB_USER=admin
5 DB_PASSWORD=password
6 DB_NAME=jugadores
7 REDIS_HOST=192.168.1.38
8 REDIS_PORT=6379
```

Listado A.24: Ejemplo de configuración de constantes para el servidor back-end

Para el despliegue en un entorno de producción, se recomienda compilar previamente la aplicación y ejecutar el código resultante. Este procedimiento se realiza en dos fases. En primer lugar, se lleva a cabo la compilación del proyecto mediante el comando recogido en el Código [A.25](#):

```
1 npm run build
```

Listado A.25: Compilación del proyecto NestJS

Una vez completada la compilación, se ejecuta la aplicación utilizando el código transpilado, tal como se indica en el Código [A.26](#):

```
1 node dist/main
```

Listado A.26: Ejecución del servidor NestJS en producción

A.3.3. Servidor Redis

Una forma sencilla y eficiente de desplegar Redis es mediante el uso de contenedores Docker. Para ello, es necesario contar con Docker instalado en el sistema. Una vez instalado, se puede obtener la imagen oficial de Redis desde el repositorio de Docker Hub ejecutando el comando que se muestra en el Código [A.27](#):

```
1 docker pull redis
```

Listado A.27: Descarga de la imagen oficial de Redis desde Docker Hub

Una vez descargada la imagen, se puede iniciar un contenedor de Redis con la configuración deseada. Por ejemplo, para ejecutarlo en segundo plano y exponer el puerto por defecto (6379), se puede utilizar el comando del Código [A.28](#):

```
1 docker run -d --name redis-server -p 6379:6379 redis
```

Listado A.28: Ejecución de un contenedor Redis en segundo plano

Este comando crea y ejecuta un contenedor denominado `redis-server`, mapeando el puerto local 6379 al puerto del contenedor, lo cual permite que el servidor back-end pueda conectarse a Redis a través de la dirección indicada en la variable `REDIS_HOST` del archivo `.env`.

En caso de que sea necesario detener el contenedor, se puede utilizar el siguiente comando:

```
1 docker stop redis-server
```

Listado A.29: Parada del contenedor Redis

Y para reiniciarlo posteriormente:

```
1 docker start redis-server
```

Listado A.30: Reinicio del contenedor Redis

A.4. MRT

A.4.1. Base de Datos Local Reducida

Para generar la base de datos local reducida, es necesario ejecutar el algoritmo de extracción de datos, proporcionando los parámetros requeridos para establecer la conexión con la base de datos MongoDB y definir las características del proceso de extracción y almacenamiento. Los argumentos que debe recibir el script son los siguientes:

- `hostdb`: dirección del host de la base de datos.
- `port`: puerto de conexión.
- `dbname`: nombre de la base de datos.
- `path`: directorio destino para almacenar los documentos extraídos.
- `nthreads`: número de hilos concurrentes a utilizar.
- `diasmediana`: número de días considerados para el cálculo de la mediana de los precios de los jugadores.
- `username`: usuario para autenticación.
- `password`: contraseña correspondiente.

A modo de ejemplo, la ejecución del script desde línea de comandos podría realizarse de la manera que se indica en el Código [A.31](#).

```
1 python extractor_datos.py 192.168.1.38 27017 jugadores /ruta/a/documentos 8 30 --username  
admin --password secret
```

Listado A.31: Ejemplo de ejecución del script de extracción de datos

A.4.2. Servidores Worker

La instalación de los servidores workers sigue un procedimiento análogo al empleado para el servidor back-end (ver Sec. A.3.2). La principal diferencia radica en la configuración del archivo `.env`, donde se deben especificar las rutas absolutas de los archivos `.jar` correspondientes a los algoritmos de toma de decisión y de optimización.

A continuación, se muestra un ejemplo de configuración en el archivo `.env`, donde además se indican las credenciales para la conexión con Redis y la ruta a la base de datos local reducida:

```
1 REDIS_HOST=192.168.1.38  
2 REDIS_PORT=6379  
3 SELECCION_DE_JUGADORES_PATH=C:\\Users\\jucav\\Documents\\NetBeansProjects\\Topsis_Jugador  
  \\target\\Topsis_Jugador-1.0-SNAPSHOT.jar  
4 CREACION_DE_EQUIPOS_AUTOMATICA_PATH=C:\\Users\\jucav\\Documents\\NetBeansProjects\\  
  ColoniasHormigas\\target\\ColoniasHormigas-1.0-SNAPSHOT.jar  
5 BASE_DATOS_LOCAL_REDUCIDA_PATH=C:\\Users\\jucav\\Desktop\\TFG\\CÓDIGO\\servidor-para-  
  tareas\\Extractor_de_Datos\\datos
```

Listado A.32: Configuración del archivo `.env` para servidores worker

A.5. MA

Para iniciar el MA, es necesario ejecutar en primer lugar el archivo `main.py`. Esta acción generará automáticamente una carpeta denominada `coleccion`, en la cual deberán almacenarse los chunks en formato `.txt`.

El proceso de creación de la colección se compone de las siguientes etapas:

1. **Transcripción de los audios:** es necesario transcribir previamente todos los archivos de audio disponibles. Para ello, se debe ejecutar el script `transcribir.py`, indicando como argumentos la carpeta de origen que contiene los audios y la carpeta de destino donde se almacenarán las transcripciones resultantes.

2. **Agrupación de las transcripciones:** una vez finalizado el proceso de transcripción, los archivos generados deben reunirse en una única carpeta, que se utilizará posteriormente como fuente para la generación de los chunks.
3. **Generación de chunks:** finalmente, se ejecuta el algoritmo encargado de dividir los textos transcritos en fragmentos o chunks, los cuales serán almacenados automáticamente en la carpeta `coleccion`, llamado `crear_coleccion.py`.

Antes de poner en marcha el servidor MA, es imprescindible instalar las bibliotecas necesarias para garantizar su correcto funcionamiento. En el Código [A.33](#), se muestran los comandos necesarios para instalar las dependencias principales mediante `pip`.

```
1 pip install redis
2 pip install fastapi
3 pip install uvicorn
4 pip install pydantic
5 pip install sentence-transformers
6 pip install numpy
7 pip install scipy
8 pip install requests
9 pip install markdown
10 pip install scikit-learn
```

Listado A.33: Instalación de dependencias para el servidor MA

Una vez instaladas las dependencias, se debe ejecutar nuevamente el archivo `main.py`, esta vez especificando como parámetros la dirección IP y el puerto del servidor Redis. A modo ilustrativo, la ejecución del servidor desde la línea de comandos puede realizarse de la siguiente manera (ver Código [A.34](#)):

```
1 python servidor_ma.py 192.168.1.38 6379
```

Listado A.34: Ejemplo de ejecución del servidor MA

Apéndice B

Manual de Usuario

Este apéndice presenta el manual de usuario de FC Assistant, el cual ofrece una explicación detallada y estructurada sobre la navegación por la interfaz, la creación y personalización de plantillas. Asimismo, se describe el funcionamiento de la ventana modal de comparación y del asistente inteligente. El manual se complementa con capturas ilustrativas que contribuyen a una mejor comprensión.

B.1. Página Principal

Desde la página principal de la aplicación, puedes acceder fácilmente a la interfaz para crear plantillas. Solo tienes que hacer clic en el enlace que aparece en la barra superior de navegación (ver Figura [B.1](#)).

B.2. Página de Creación de Plantillas

Una vez dentro de la página de creación de plantillas, encontrarás varias secciones diseñadas para ayudarte a construir y personalizar tu equipo de manera rápida e intuitiva (ver Figura [B.2](#)).



Figura B.1: Página principal de FC Assistant



Figura B.2: Página de creación de plantillas de FC Assistant

Sección de la química

En esta parte verás un gráfico que representa cómo se calcula la química de tu equipo. Cada símbolo verde indica cuántos jugadores de tu plantilla comparten club, liga o nacionalidad.

Cuando varios símbolos verdes aparecen seguidos, los jugadores relacionados reciben un punto extra en su química. Esta visualización te ayudará a entender de forma rápida cómo encajan los jugadores entre sí (ver Figura [B.3](#)).

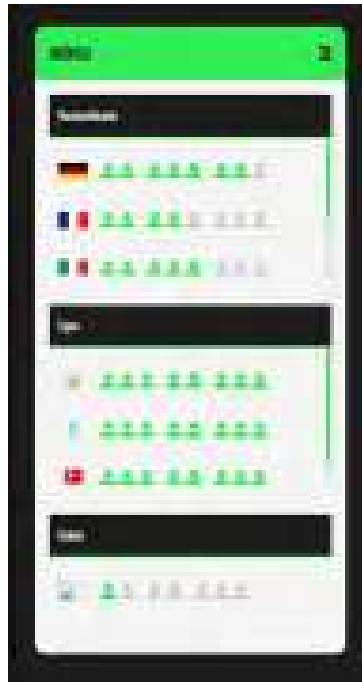


Figura B.3: Gráfico de la química en FC Assistant

Sección de la plantilla

Aquí podrás ver cómo está formada actualmente tu plantilla, con una representación gráfica de cada jugador en su posición correspondiente (ver Figura B.4).



Figura B.4: Representación gráfica de la plantilla en FC Assistant

Al hacer clic en una posición concreta, se abrirá un menú con varias opciones útiles (ver Figura B.5):



Figura B.5: Menú de opciones de una posición de plantilla en FC Assistant

- **Comparador:** abre una ventana para comparar jugadores y seleccionar el más adecuado.
- **Copiar Importancias:** copia tus preferencias de esa posición para aplicarlas en otra.
- **Copiar Filtros:** copia los filtros usados en esa posición para reutilizarlos fácilmente.
- **Ignorar Precio / Contemplar Precio:** decide si quieres que el precio del jugador influya o no en las valoraciones.
- **Eliminar:** quita al jugador asignado en esa posición.

Sección de construcción del equipo

Desde esta sección puedes establecer el presupuesto máximo total de tu plantilla. Una vez definido, pulsa el botón “Construir” para generar automáticamente un equipo que se ajuste a tus preferencias y limitaciones (ver Figura B.6).

Sección de ajustes

Por último, en la sección de ajustes (ver Figura B.7) puedes cambiar la formación de tu equipo y, si lo necesitas, borrar todos los jugadores asignados con un solo clic para empezar desde cero.



Figura B.6: Sección para construir automáticamente el equipo en FC Assistant

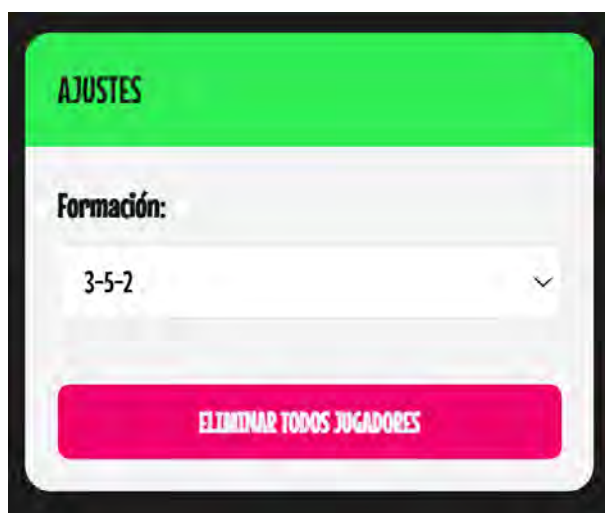


Figura B.7: Sección de ajustes en FC Assistant

B.3. Ventana Modal de Comparación de Jugadores

Cuando hagas clic en la opción *Comparador* dentro del menú de una posición concreta, se abrirá una ventana que te permitirá comparar jugadores de forma visual y sencilla. Esta ventana está dividida en varias secciones que te ayudarán a tomar la mejor decisión.

Sección Jugadores

Nada más entrar en la ventana modal, verás una lista con los jugadores disponibles para la posición seleccionada. Estos aparecen ordenados de mayor a menor según su precio, lo que te ayudará a identificar rápidamente qué opciones se ajustan mejor a tu

presupuesto, especialmente si estás limitado por un coste máximo (ver Figura B.8).



Figura B.8: Ventana modal sin haber seleccionado un jugador en FC Assistant

Para ayudarte a tomar decisiones, puedes pasar el ratón por encima de un jugador para ver su carta ampliada a la izquierda. Si ya tienes un jugador seleccionado y colocas el ratón sobre otro, verás una comparación directa entre ambos. Esta vista comparativa destaca las diferencias clave entre los dos, lo que te facilita elegir al que mejor se adapta a tu estrategia (ver Figura B.9 y la Figura B.10).

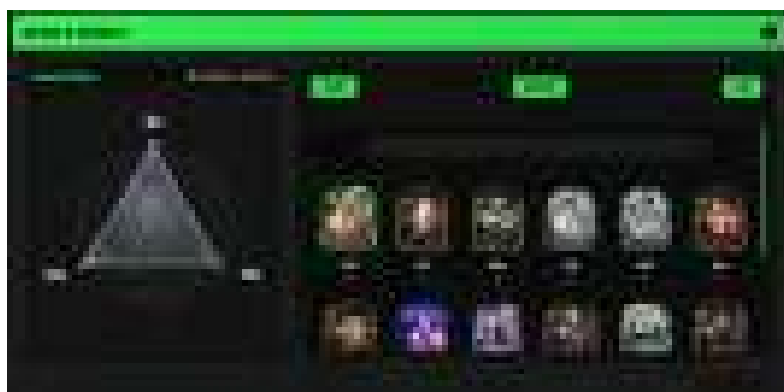


Figura B.9: Ventana modal comparando jugadores en FC Assistant

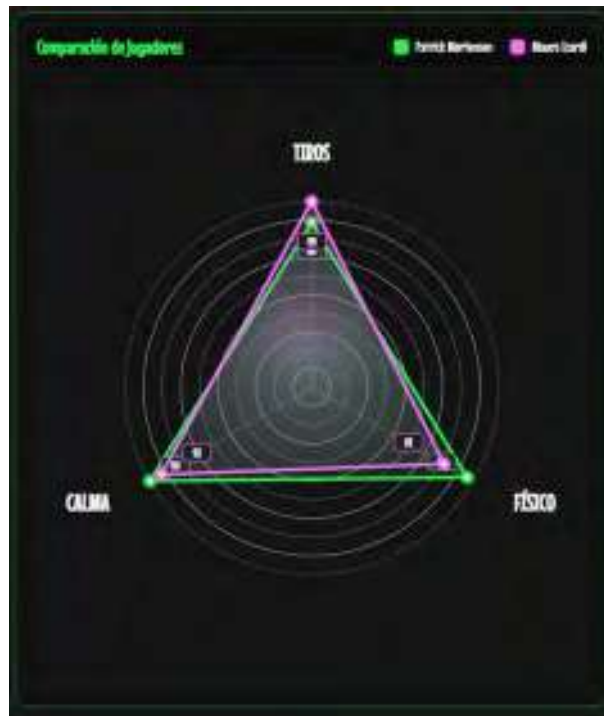


Figura B.10: Ventana modal gráfico comparativo en FC Assistant

Cuando encuentres al jugador que deseas, simplemente haz clic sobre él y será añadido automáticamente a tu plantilla en la posición que habías seleccionado.

Sección Importancias

Aquí puedes indicar qué aspectos te parecen más importantes a la hora de comparar jugadores para una posición concreta (ver Figura B.11). Puedes escoger entre varios criterios como los atributos del jugador, su rol, su estilo de juego, o detalles más específicos como el precio, la altura o la química. Incluso puedes configurar si le das importancia a que ese atributo sea menor o mayor.



Figura B.11: Añadir importancias en ventana modal para una posición en FC Assistant

Cada vez que añadas unas características, aparecerá en la parte derecha con una barra que te permite ajustar cuánto valor le das frente al resto (ver Figura B.12). Cuanto más separado esté un control del resto, más peso tendrá en la comparación.



Figura B.12: Importancias añadidas para una posición en FC Assistant

Para que todo esto sea más fácil de entender, en la parte izquierda verás un gráfico circular que se actualiza al momento (ver Figura B.13). Este gráfico te muestra de forma visual cómo se distribuye la importancia entre los distintos aspectos que has elegido.

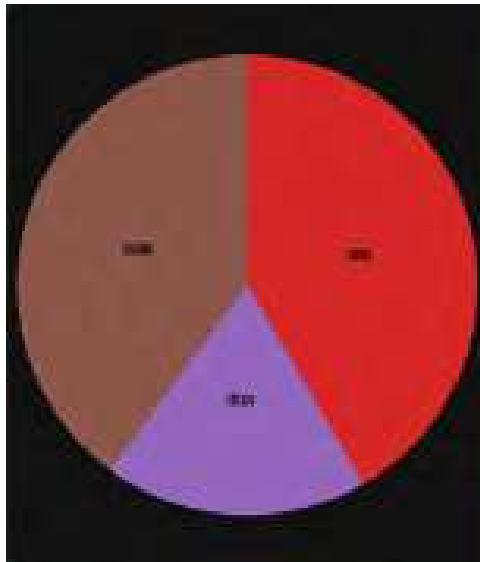


Figura B.13: Gráfico de importancias en FC Assistant

Sección Filtros

Esta sección te permite afinar aún más tu búsqueda. Puedes elegir diferentes filtros desde un menú desplegable (por ejemplo, nacionalidad, liga o estilo de juego), y luego configurar cada uno en detalle desde la parte derecha (ver Figura B.14). Así podrás centrarte solo en los jugadores que realmente te interesan.



Figura B.14: Añadir filtros en la ventana modal para una posición en FC Assistant

B.4. Ventana del Asistente

La ventana del asistente, situada en la esquina inferior derecha de la interfaz de usuario, proporciona un canal de comunicación directa con el sistema de ayuda inteligente integrado en FC Assistant (Figura B.15).

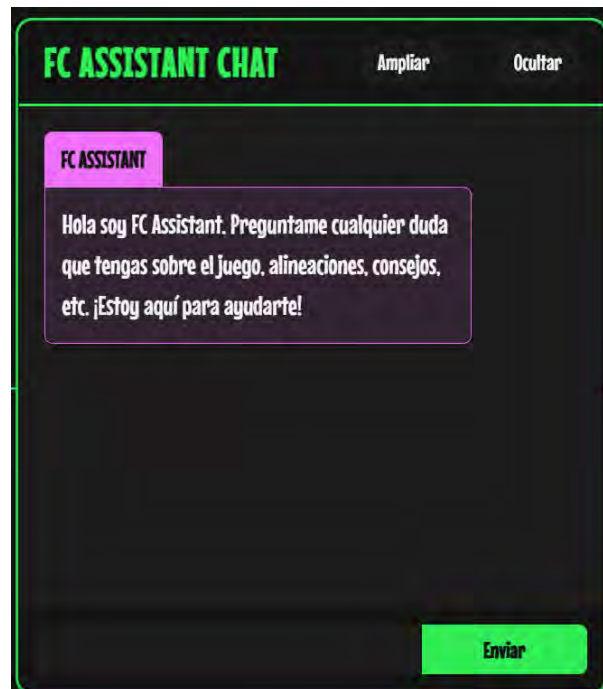


Figura B.15: Ventana de conversación con el asistente en FC Assistant

Además de su funcionalidad principal de asistencia conversacional, esta ventana ofrece varias opciones diseñadas para mejorar la usabilidad del sistema:

- **Ampliar:** al pulsar este botón, la ventana se despliega hacia la izquierda, ocupando todo el ancho de la pantalla. Esta funcionalidad resulta especialmente útil para examinar respuestas extensas o complejas con mayor comodidad.
- **Ocultar:** al utilizar esta opción, la ventana se minimiza en la parte inferior, permitiendo al usuario interactuar con otras áreas de la aplicación sin interrupciones visuales.

La interacción con el asistente se realiza mediante el campo de entrada de texto, situado en la parte inferior de la ventana (resaltado en color verde). Una vez redactada la consulta, basta con pulsar el botón “Enviar” para recibir una respuesta personalizada.

El asistente está capacitado para responder a diversas solicitudes. Por ejemplo, puede analizar la plantilla actual del usuario sugiriendo estrategias específicas, e incluso generar automáticamente un equipo completo a partir de prompts, sin necesidad de configuración manual previa.

