TRABAJO FIN DE GRADO

ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Informática

IA COMO DUNGEON MÁSTER

Autor/a:

Manuel Andrés Pallero Duque

Director/a:

D. Juan Bonastre Egea

Murcia, junio de 2025

TRABAJO FIN DE GRADO

ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Informática

IA COMO DUNGEON MÁSTER

Autor/a:

Manuel Andrés Pallero Duque

Director/a:

D. Juan Bonastre Egea

Murcia, junio de 2025

Agradecimientos

Este Trabajo de Fin de Grado no habría sido posible sin el apoyo inestimable de las personas más importantes en mi vida. A toda mi familia, gracias por vuestra confianza, por animarme en cada paso y por entender las horas dedicadas a este proyecto.

Un agradecimiento especial y lleno de cariño para mi mujer, Inma. Tu paciencia, tu apoyo constante en el día a día, y tu capacidad para motivarme incluso en los momentos más complicados han sido cruciales. Gracias por creer en mí y por toda la ayuda que me has brindado para que pudiera dedicarme a mis estudios y poder cumplir esta meta. Este logro también es tuyo.

Listado de Abreviatura

**API:** Application Programming Interface (Interfaz de Programación de Aplicaciones)

**CPU:** Central Processing Unit (Unidad Central de Procesamiento)

**D&D:** Dragones y Mazmorras

**DM:** Dungeon Master

**GPU:** Graphics Processing Unit (Unidad de Procesamiento Gráfico)

**IA:** Inteligencia Artificial

**LLM:** Large Language Model (Modelo de Lenguaje Grande)

**NVMe:** Non-Volatile Memory Express (Memoria Exprés No Volátil)

**PLN:** Procesamiento del Lenguaje Natural

**PNJ:** Personaje No Jugador

**RAG:** Retrieval Augmented Generation (Generación Aumentada por Recuperación)

**RAM:** Random Access Memory (Memoria de Acceso Aleatorio)

**SSD:** Solid State Drive (Unidad de Estado Sólido)

**TFG:** Trabajo de Fin de Grado

**TTRPG:** Tabletop Role-Playing Game (Juego de Rol de Mesa)

**VRAM:** Video Random Access Memory (Memoria de Acceso Aleatorio de Video)

**ÍNDICE**

[RESUMEN 19](#_Toc200615065)

[1. INTRODUCCIÓN 23](#_Toc200615066)

[1.1. Motivación 23](#_Toc200615067)

[1.2. Definición 24](#_Toc200615068)

[1.3. Objetivos propuestos (generales y específicos) 25](#_Toc200615069)

[2. ESTUDIO DEL MERCADO (O ESTADO DEL ARTE) 27](#_Toc200615070)

[2.1. Conceptos relevantes del dominio de aplicación. 27](#_Toc200615071)

[2.2. Relación con proyectos con la misma funcionalidad 28](#_Toc200615072)

[Nombre de la herramienta 29](#_Toc200615073)

[Funcionalidad Principal 29](#_Toc200615074)

[Modelo de Acceso 29](#_Toc200615075)

[2.3. Estudio de viabilidad 30](#_Toc200615076)

[2.3.1. Alcance del proyecto 30](#_Toc200615077)

[2.3.2. Estudio de la situación actual 31](#_Toc200615078)

[2.3.3. Estudio y valoración de las alternativas de solución 33](#_Toc200615079)

[2.3.4. Selección de la solución 34](#_Toc200615080)

[3. METODOLOGÍAS USADAS (MÉTRICA, UML, ETC.) 37](#_Toc200615081)

[3.1. Visión General de las Metodologías de Desarrollo de Software 37](#_Toc200615082)

[3.2. Selección Metodológica para este TFG: El Modelo en Cascada Iterativa 38](#_Toc200615083)

[3.2.1. Plan de Trabajo Detallado como Eje Metodológico 38](#_Toc200615084)

[3.3. Gestión de Requisitos 40](#_Toc200615085)

[3.4. Modelado del Sistema con UML 41](#_Toc200615086)

[3.5. Estrategias de Prueba y Evaluación 42](#_Toc200615087)

[3.6. Justificación del Enfoque Metodológico Seleccionado 42](#_Toc200615088)

[4. TECNOLOGÍAS Y HERRAMIENTAS UTILIZADAS EN EL PROYECTO 45](#_Toc200615089)

[4.1. Entorno de Desarrollo y Programación 45](#_Toc200615090)

[4.1.1. Python (Lenguaje de Programación Principal) 45](#_Toc200615091)

[4.1.2. Entorno Virtual (venv) 45](#_Toc200615092)

[4.1.3. Git (Control de Versiones) 46](#_Toc200615093)

[4.2. Interacción con Discord 46](#_Toc200615094)

[4.2.1. discord.py (Librería para la API de Discord) 46](#_Toc200615095)

[4.2.2. Discord Developer Portal 46](#_Toc200615096)

[4.3. Inteligencia Artificial y Modelo de Lenguaje (LLM) 46](#_Toc200615097)

[4.3.1. Mistral (Gran Modelo de Lenguaje) 47](#_Toc200615098)

[4.3.2. Hugging Face transformers (Librería para LLMs) 47](#_Toc200615099)

[4.3.3. torch / tensorflow (Frameworks de Deep Learning) 47](#_Toc200615100)

[4.4. Sistema de Recuperación Aumentada por Generación (RAG) 47](#_Toc200615101)

[4.4.1. sentence-transformers (Generación de Embeddings) 47](#_Toc200615102)

[4.4.2. faiss-cpu (Índice de Similitud Vectorial) 48](#_Toc200615103)

[4.4.3. Dataset "Fireball" (Fuente de Conocimiento) 48](#_Toc200615104)

[4.5. Gestión de Datos y Persistencia 48](#_Toc200615105)

[4.5.1. SQLite (Base de Datos Local) 48](#_Toc200615106)

[4.5.2. datasets (Manejo de Conjuntos de Datos) 48](#_Toc200615107)

[4.6. Herramientas y Librerías Auxiliares 49](#_Toc200615108)

[4.6.1. python-dotenv (Gestión de Variables de Entorno) 49](#_Toc200615109)

[4.6.2. asyncio.to\_thread (Programación Asíncrona) 49](#_Toc200615110)

[4.7. Documentación y Gestión del Proyecto 49](#_Toc200615111)

[4.7.1. README.md (Documentación del Repositorio) 49](#_Toc200615112)

[4.7.2. Plan de Trabajo Semanal (Herramienta de Seguimiento) 49](#_Toc200615113)

[5. ESTIMACIÓN DE RECURSOS Y PLANIFICACIÓN 51](#_Toc200615114)

[5.1. Estimación del Esfuerzo del Proyecto 51](#_Toc200615115)

[5.2. Realizar una planificación temporal del proyecto (diagrama de Pert, diagrama de Gantt, etc) 53](#_Toc200615116)

[5.3. Realizar una valoración de la dedicación y el coste económico 55](#_Toc200615117)

[6. DESARROLLO DEL CONTENIDO DEL PROYECTO (EXPLICACIÓN DETALLADA DEL ANÁLISIS, DISEÑO Y SOLUCIONES APORTADAS). 59](#_Toc200615118)

[6.1. Análisis de Requisitos y Alcance Funcional 59](#_Toc200615119)

[6.1.1. Objetivos y Alcance General 59](#_Toc200615120)

[6.1.2. Actores del Sistema 60](#_Toc200615121)

[6.1.3. Requisitos Funcionales 60](#_Toc200615122)

[6.1.4. Requisitos No Funcionales 61](#_Toc200615123)

[6.2. Diseño del Sistema 62](#_Toc200615124)

[6.2.1. Arquitectura General del Sistema 62](#_Toc200615125)

[6.2.2. Diseño del Bot de Discord 63](#_Toc200615126)

[6.2.3. Diseño del Módulo de Inteligencia Artificial 64](#_Toc200615127)

[6.2.4. Diseño del Módulo de Gestión de Datos (Hojas de Personaje) 64](#_Toc200615128)

[6.2.5. (Opcional) Diseño del Módulo de Combate 65](#_Toc200615129)

[6.3. Implementación y Soluciones Aportadas 65](#_Toc200615130)

[6.3.1. Configuración del Entorno de Desarrollo (Semana 1) 65](#_Toc200615131)

[6.3.2. Desarrollo del Bot Básico e Integración Inicial con LLM (Semana 1) 65](#_Toc200615132)

[6.3.3. Implementación del Sistema RAG (Semana 2) 66](#_Toc200615133)

[6.3.4. Integración del Núcleo del DM (RAG + LLM + Historial) (Semana 3) 66](#_Toc200615134)

[6.3.5. Desarrollo de Expansiones (Semanas 4-5) 66](#_Toc200615135)

[6.3.6. Desafíos y Decisiones Clave durante la Implementación 67](#_Toc200615136)

[6.4. Pruebas y Verificación del Sistema 67](#_Toc200615137)

[6.4.1. Estrategia y Casos de Prueba 67](#_Toc200615138)

[6.4.2. Resultados de las Pruebas y Corrección de Errores 68](#_Toc200615139)

[6.4.3. Verificación del Cumplimiento de Requisitos 69](#_Toc200615140)

[7. DESPLIEGUE Y PRUEBA DE LA SOLUCIÓN 71](#_Toc200615141)

[7.1. Plan de Pruebas 71](#_Toc200615142)

[7.1.1. Objetivos de las Pruebas 71](#_Toc200615143)

[7.1.2. Alcance de las Pruebas 72](#_Toc200615144)

[7.1.3. Tipos de Pruebas y Estrategia 72](#_Toc200615145)

[7.1.4. Entorno de Pruebas 74](#_Toc200615146)

[7.1.5. Criterios de Aceptación 75](#_Toc200615147)

[7.2. Indicadores del Desempeño y/o Utilidad 75](#_Toc200615148)

[7.3. Escalabilidad / Extensión / Mantenimiento / Soporte 77](#_Toc200615149)

[7.3.1. Escalabilidad 77](#_Toc200615150)

[7.3.2. Extensión (Líneas de Trabajo Futuro) 77](#_Toc200615151)

[7.3.3. Mantenimiento 78](#_Toc200615152)

[7.3.4. Soporte 79](#_Toc200615153)

[7.4. Plan de Formación de Usuarios (Guía de Usuario) 79](#_Toc200615154)

[7.4.1. Público Objetivo 79](#_Toc200615155)

[7.4.2. Contenido de la Guía de Usuario 80](#_Toc200615156)

[7.4.3. Formato y Distribución 81](#_Toc200615157)

[8. CONCLUSIONES 83](#_Toc200615158)

[8.1. Objetivos Alcanzados 83](#_Toc200615159)

[8.2. Conclusiones del Trabajo y Personales 85](#_Toc200615160)

[8.2.1. Conclusiones del Trabajo 85](#_Toc200615161)

[8.2.2. Conclusiones Personales 86](#_Toc200615162)

[8.3. Vías Futuras 87](#_Toc200615163)

[9. BIBLIOGRAFÍA 91](#_Toc200615164)

[10. ANEXOS 95](#_Toc200615165)

[10.1. Manuales de instalación (en caso de ser necesarios) 95](#_Toc200615166)

[10.2. Manuales de usuario (en caso de ser necesarios) 95](#_Toc200615167)

[10.3. Otros documentos, manuales 95](#_Toc200615168)

ÍNDICE DE ELEMENTOS GRÁFICOS(Sí hay en el TFG)

Clicar en (no se encuentran…) tras haber citado automáticamente (como en el ejemplo de tabla) y actualizar índice en:

Referencias > Actualizar tabla

**FIGURA**

No se encuentran elementos de tabla de ilustraciones.

**TABLA**

[Tabla 1 Comparativa simplificada de Herramientas de IA para DM 27](#_Toc198021689)

**GRÁFICO**

No se encuentran elementos de tabla de ilustraciones.

**IMAGEN**

No se encuentran elementos de tabla de ilustraciones.

**Para crear un Índice de una categoría nueva**

habrá que ir a:

Referencias > Insertar Tabla de ilustraciones

Y en “rótulo de título” elegir la categoría (Tabla, figura, ecuación…)

que hayamos creado previamente

RESUMEN

Este trabajo aborda la creación de un Dungeon Máster (DM) mediante Inteligencia Artificial (IA), motivado por los recientes avances en Modelos de Lenguaje Grandes (LLMs) y la creciente popularidad de los juegos de rol como Dungeons & Dragons (D&D). El proyecto busca ofrecer una solución innovadora para dirigir partidas y enriquecer la experiencia de juego, respondiendo a la escasez de DMs y explorando el potencial de la IA en este ámbito lúdico y narrativo.

El objetivo principal del presente Trabajo de Fin de Grado es desarrollar un sistema de IA funcional, accesible a través de un bot de Discord. Este sistema está diseñado para actuar como DM en partidas de Dungeons & Dragons, siendo capaz de generar narrativas dinámicas y coherentes, interpretar las acciones de los jugadores, aplicar reglas del juego mediante técnicas de Generación Aumentada por Recuperación (RAG) con el modelo de lenguaje Mistral y el dataset 'Fireball', y gestionar interacciones básicas con personajes no jugadores (PNJs).

Para la consecución de estos objetivos, se empleó una metodología híbrida adaptada, que combina principios de planificación estructurada con prácticas ágiles para el desarrollo iterativo de los componentes de IA. Las tecnologías clave utilizadas incluyen Python como lenguaje de programación principal, la librería discord.py para la interacción con la plataforma Discord, el LLM Mistral para la generación de contenido narrativo, un sistema RAG implementado con sentence-transformers y FAISS para la contextualización basada en el dataset "Fireball", y SQLite para la persistencia de datos como las hojas de personaje.

Como resultado de este proceso de desarrollo, se ha obtenido un prototipo funcional del DM IA integrado en Discord. El sistema es capaz de generar narrativas contextualizadas en respuesta a las acciones de los jugadores, gestionar información básica de los personajes a través de hojas de personaje digitales, y mantener un flujo de partida coherente, demostrando así la viabilidad de la solución técnica y el cumplimiento de los principales hitos funcionales propuestos.

Se concluye que la integración de Modelos de Lenguaje Grandes con técnicas de Generación Aumentada por Recuperación constituye una aproximación prometedora y efectiva para simular las funciones de un DM. El diseño de *prompts* se ha revelado como un factor crítico para guiar adecuadamente al LLM. Si bien se reconocen las limitaciones inherentes a la tecnología actual, como la coherencia narrativa a muy largo plazo y la complejidad en la evaluación objetiva de la calidad de la experiencia de rol, el proyecto desarrollado establece una base sólida y evidencia un considerable potencial para futuras extensiones que podrían enriquecer la interactividad, la profundidad de las mecánicas de juego y la personalización de las partidas de rol asistidas por IA

Palabras claves: Inteligencia Artificial, Dungeon Master, Juegos de Rol, Modelos de Lenguaje Grandes, LLM, Mistral, Generación Aumentada por Recuperación, RAG, Discord, Python, Procesamiento del Lenguaje Natural, PLN.

ABSTRACT(Opcional - párrafo único)

Recent advancements in Artificial Intelligence (AI), particularly in Large Language Models (LLMs), coupled with the enduring popularity of tabletop role-playing games (RPGs) like Dungeons & Dragons, offer a unique opportunity to innovate the gaming experience. This project focuses on the development of an AI-driven Dungeon Master designed to operate within the Discord platform, aiming to address the common challenge of DM availability and to explore new dimensions of interactive storytelling. The primary objective was to create a functional prototype capable of generating dynamic narratives, interpreting player actions, and managing game context using the Mistral LLM, enhanced by a Retrieval Augmented Generation (RAG) system leveraging the "Fireball" dataset for specific knowledge grounding. The methodology involved a hybrid approach, integrating structured planning with agile, iterative development of AI components, utilizing Python, discord.py, sentence-transformers for embeddings, FAISS for similarity search, and SQLite for data persistence such as character sheets. The implemented system successfully demonstrates the AI's ability to produce contextualized narrative responses and manage basic game mechanics via the Discord bot. This work concludes that the synergy of LLMs with RAG techniques presents a viable and promising pathway for creating AI DMs, while also highlighting the critical importance of prompt engineering and acknowledging areas for future development in terms of long-term narrative coherence, complex rule adherence, and overall user immersion.

Keywords: Artificial Intelligence, Dungeon Master, Role-Playing Games, Large Language Models, LLM, Mistral, Retrieval Augmented Generation, RAG, Discord, Python, Natural Language Processing, NLP.

# INTRODUCCIÓN

En los últimos años, la IA, más concretamente los modelos de lenguajes más grandes conocidos como LLMs, han demostrado unas capacidades sorprendentes en la generación de texto, abriendo un mundo de posibilidades.

Al mismo tiempo los juegos de mesa y rol están teniendo un auge en su popularidad. En el caso de los juegos de rol Dragones y Mazmorras (D&D) ha tenido un gran aumento de popularidad debido a los videojuegos como Baldur`s Gate 3 que fue el mejor juego del año en su lanzamiento y siendo un movimiento de masas hacia el sector de los juegos de mesa y rol.

Otras como series como “Stranger Things”, D&D es un punto importante de su historia ya que los protagonistas juegan a dicho juego y creen que las cosas que suceden en la vida real están relacionadas con el juego.

También existen series como “La leyenda de Vox Machina” que son adaptaciones de una campaña de rol que se ha pasado al formato televisivo como serie de animación.

Con este contexto se presenta una oportunidad para explorar como la IA puede asistir o simular las funciones de un director de juego o DM, mejorando la experiencia de juego haciendo que sea más inmersivo permitiendo crear mejores historias sin tener que preparar demasiado contexto siendo DM.

## Motivación

La motivación central de este TFG brota de la sinergia entre los recientes y exponenciales avances en IA generativa y la creciente necesidad de innovar en el ámbito de los juegos de rol, un sector en clara expansión. Como aficionado y DM experimentado, he sido testigo del profundo atractivo de *Dungeons & Dragons* (D&D), así como de las dificultades inherentes a la figura del DM, cuyo rol es crucial pero exigente en tiempo y dedicación.

El auge de los LLMs presenta una oportunidad única para abordar un desafío común: la escasez de personas dispuestas a dirigir partidas. Esta situación, a menudo, impide que los grupos disfruten de estas experiencias narrativas compartidas. La idea de desarrollar una IA capaz de asumir o facilitar las tareas del DM surge, por tanto, como una solución prometedora.

Este proyecto busca explorar cómo la IA puede no solo aligerar la carga del DM, sino también enriquecer la dinámica de juego y, fundamentalmente, democratizar el acceso a los juegos de rol. Se pretende crear una herramienta que, partiendo de mi interés personal por esta convergencia tecnológica y lúdica, contribuya a solventar una barrera real para muchos jugadores, permitiendo que más personas puedan sumergirse en el fascinante mundo del rol.

## Definición

En el presente trabajo se abordá el diseño y desarrollo de un sistema de IA cuyo objetivo es actuar como DM en partidas de rol de D&D. El núcleo del proyecto reside en la creación de un bot para la plataforma Discord, permitiendo la interacción de los usuarios mediante dicha plataforma.

Para lograr la funcionalidad de DM, el sistema integrará un Modelo de Lenguaje Grande (LLM), específicamente una variante de Mistral, responsable de generar respuestas narrativas y diálogos para los personajes no jugadores (PNJs). Asimismo, se implementará una técnica de Generación Aumentada por Recuperación (RAG), utilizando el dataset 'Fireball' como base de conocimiento. Esta aproximación permitirá al LLM acceder a información contextual relevante para enriquecer las narrativas, asegurar la coherencia con las reglas del juego y mejorar la calidad de las interacciones y encuentros. El objetivo final es entregar un sistema funcional que demuestre estas capacidades, explorando el potencial de la IA para dinamizar y hacer más accesibles las partidas de rol.

Diagrama

El contenido generado por IA puede ser incorrecto.

Ilustración : Diagrama de la arquitectura del TFG

## Objetivos propuestos (generales y específicos)

Desarrollar un sistema de DM funcional, accesible a través de un bot de Discord, que actúe como Asistente de DM para partidas de *Dungeons & Dragons* (D&D). Este sistema deberá ser capaz de generar narrativas, interpretar acciones de los jugadores, aplicar reglas del juego mediante técnicas de RAG con el LLM Mistral, y gestionar interacciones básicas con personajes no jugadores (PNJs), con el fin de enriquecer y facilitar la experiencia de los juegos de rol.

Para ello, es necesario cumplir una serie de objetivos específicos:

1. **Objetivo específico 1:** implementar una interfaz de comunicación a través de un bot de Discord, permitiendo a los jugadores introducir sus decisiones y descripciones de sus acciones mediante comandos de texto.
2. **Objetivo específico 2:** Implementar la capacidad en la IA para generar respuestas narrativas dinámicas y coherentes, incluyendo descripciones de escenarios, eventos y diálogos de personajes no jugadores (PNJs), utilizando el modelo de lenguaje Mistral.
3. **Objetivo específico 3:** Aplicar las reglas del juego mediante datasets como "Fireball" como base fundamental del mundo virtual gestionado por la IA, asegurando que las mecánicas del juego y la información contextual sean accesibles y utilizadas por el sistema mediante técnicas RAG.
4. **Objetivo específico 4:** Establecer un control del flujo de juego que se base en una secuencia de turnos o interacciones lógicas, donde la IA gestione la progresión de la partida en función de las acciones de los jugadores y los eventos narrativos.
5. **Objetivo específico 5:** Capacitar a la IA para el manejo básico de Personajes No Jugadores (PNJs), permitiendo que estos interactúen de forma creíble con los jugadores y contribuyan al desarrollo de la narrativa.

# ESTUDIO DEL MERCADO

En este capítulo se presenta el marco contextual del proyecto "IA como Dungeon Master". Se comenzará definiendo los conceptos tecnológicos y del dominio de aplicación más relevantes que sustentan el desarrollo propuesto. A continuación, se analizarán proyectos y herramientas existentes con funcionalidades similares o que aborden problemáticas parecidas, con el fin de identificar el estado actual de la técnica y las oportunidades de innovación que este trabajo pretende explorar. Finalmente, se realizará un estudio de viabilidad detallado que abarcará el alcance preciso del proyecto, un análisis de la situación actual de las tecnologías a emplear, y una valoración de las alternativas consideradas para la selección final de la solución técnica a implementar.

## Conceptos relevantes del dominio de aplicación.

La IA es un campo de la informática dedicado a la creación de sistemas capaces de realizar tareas que típicamente requieren inteligencia humana. Estas tareas incluyen el aprendizaje, el razonamiento, la resolución de problemas, la percepción, la comprensión del lenguaje y la toma de decisiones. Dentro de la IA, el Procesamiento del Lenguaje Natural (PLN) es una subdisciplina crucial que se enfoca en la interacción entre las computadoras y el lenguaje humano, permitiendo a las máquinas comprender, interpretar y generar texto o habla de manera significativa.

En los últimos años, una de las innovaciones más disruptivas dentro del PLN ha sido el desarrollo de los Modelos de Lenguaje Grandes (LLMs, por sus siglas en inglés *Large Language Models*). Los LLMs son modelos de IA entrenados con ingentes cantidades de datos textuales, utilizando arquitecturas de red neuronal profunda, predominantemente la arquitectura Transformer (Vaswani et al., 2017). Estos modelos aprenden a predecir la siguiente palabra en una secuencia, lo que les confiere una notable capacidad para generar texto coherente y contextualmente relevante, traducir idiomas, responder preguntas, resumir información y mantener conversaciones.

Para el presente proyecto, se ha seleccionado un modelo de la familia Mistral AI, conocidos por su eficiencia y su rendimiento competitivo en diversas tareas de PLN, además de ofrecer modelos con licencias permisivas que facilitan su uso en investigación y desarrollo. La elección específica recae en su capacidad para la generación de texto creativo y la comprensión contextual, aspectos fundamentales para simular las funciones de un DM y en los requisitos básicos de carga del sistema en la GPU.

## Relación con proyectos con la misma funcionalidad

El uso de IA para asistir o emular al DM en juegos de rol de mesa (TTRPGs) es un campo en crecimiento. Las herramientas existentes varían desde simples generadores de contenido hasta sistemas complejos que intentan gestionar partidas completas. Con la llegada de los Modelos de Lenguaje Extensos (LLMs), el enfoque se ha ampliado hacia la narración interactiva y la gestión de personajes no jugadores (PNJs).

Las principales categorías de estas herramientas son:

* **Generadores de Contenido Específico:** Crean elementos puntuales para las partidas.
  + **CharGen :** Genera paisajes, personajes con arte, tiendas, tabernas y monstruos, usando modelos LoRA especializados para D&D.
  + **Toolbaz AI Roleplay Scenario Generator :** Herramienta web gratuita para crear escenarios de rol básicos.
* **Plataformas de Creación de Juegos de Rol con IA:** Permiten diseñar experiencias de juego más completas.
  + **Rosebud AI RPG Maker :** Crea videojuegos de rol 2D/3D, donde la IA asiste en la generación de mundos, personajes y código.
* **Sistemas Completos de IA como DM:** Intentan simular integralmente las funciones del DM.
  + **HyperWrite AI Dungeon Master :** Crea y gestiona campañas de D&D, adaptándose a las decisiones del jugador.
  + **AI Dungeon :** Pionero en aventuras de texto interactivas, ha utilizado diversos LLMs (GPT-2/3, Jurassic-1 de AI21 Labs , modelos propios como Wayfarer ).
  + **AI Game Master - Dungeon RPG (App) :** Aplicación móvil que usa IA (basada en GPT) como DM, con generación de imágenes.

* **Proyectos Open Source:** Ofrecen transparencia y flexibilidad para la experimentación.
  + **DungeonLM :** Agente DM basado en modelos GPT de OpenAI, con interfaz en Streamlit.
  + **ai-dungeon-master (Discord Bot) :** Bot para Discord que usa GPT-4 y LlamaIndex para dirigir partidas.

La percepción de los usuarios sobre estas herramientas es mixta. Se valora la creatividad y la inmersión , pero persisten problemas como la falta de coherencia a largo plazo , la repetición y la dificultad para adherirse estrictamente a las reglas del juego. Muchas herramientas funcionan más como asistentes avanzados que como sustitutos completos del DM humano.

Tabla Comparativa simplificada de Herramientas de IA para DM

|  |  |  |
| --- | --- | --- |
| **Nombre de la herramienta** | **Funcionalidad Principal** | **Modelo de Acceso** |
| CharGen | Generación de activos de juego (personajes, mapas, etc.) | Suscripción |
| HyperWrite AI Dungeon Master | Creación y gestión de campañas de D&D | Prueba gratuita, luego suscripción |
| AI Dungeon | Generación de aventuras de texto interactivas | Niveles gratuitos y de suscripción |
| AI Game Master - Dungeon RPG | App móvil con DM por IA y generación de imágenes | Gratuito con compras en la app (tokens) |
| DungeonLM | Agente DM open source (GPT) con interfaz Streamlit | Open Source (requiere clave API OpenAI) |
| ai-dungeon-master | Bot de Discord como AI DM (GPT-4) | Open Source (requiere claves API) |

## Estudio de viabilidad

Este apartado profundiza en la viabilidad del proyecto "IA como Dungeon Master", comenzando por definir su alcance, analizando el estado actual de las tecnologías clave que se emplearán, valorando diferentes alternativas de solución y concluyendo con la selección de la más adecuada para los objetivos y limitaciones de un TFG.

### Alcance del proyecto

* **Objetivo General:** Diseñar, desarrollar y evaluar un prototipo funcional de IA que actúe como DM para una experiencia de juego de rol textual para un solo jugador, utilizando un LLM de Mistral AI.
* **Funcionalidades Principales Que Implementar:**
* Generación de narrativa dinámica en respuesta a acciones del jugador.
* Control básico de diálogos y acciones de Personajes No Jugadores (PNJs).
* Interpretación de entradas del jugador en lenguaje natural (texto).
* Gestión sencilla de reglas y estado del juego ( simulación de dados, actualización de estados básicos), explorando la "llamada a funciones" de Mistral AI.
* **Tecnologías Clave Propuestas:**
* LLM: Modelo de Mistral AI (Mistral 7B Instruct o un modelo Small/Large vía API con llamada a funciones ).
* Lenguaje de Programación: Python.
* Interfaz de Usuario: Discord.

* **Entregables Principales:**
* Prototipo funcional del AI Dungeon Master.
* Memoria de TFG (incluyendo diseño, implementación, pruebas).
* Código fuente del prototipo.
* Informe breve de pruebas de usuario cualitativas.
* **Limitaciones del Proyecto (Qué NO se Abordará) :**
* No se implementará un sistema de reglas completo de TTRPGs comerciales.
* La coherencia narrativa a muy largo plazo y memoria detallada serán desafíos.
* Prototipo puramente textual, sin gráficos avanzados ni multijugador.
* Control de PNJs básico, sin simulaciones de comportamiento profundas.
* No se realizará *fine-tuning* extensivo del LLM.
* Evaluación principalmente cualitativa y de alcance limitado.

### Estudio de la situación actual

#### El Papel de los LLMs en el DM con IA

Los Modelos de Lenguaje Extensos (LLMs) son cruciales para las herramientas avanzadas de IA para DMs, permitiendo la generación de texto creativo, gestión de diálogos y comprensión del lenguaje natural, esenciales para simular las funciones de un DM.

#### Análisis de Modelos LLM Relevantes

Mistral AI

Mistral AI destaca por su enfoque en la innovación abierta y modelos de alto rendimiento. Su filosofía de liberar modelos con pesos abiertos bajo licencias permisivas como Apache 2.0 (Mistral 7B, series Mixtral) fomenta su uso en investigación, ideal para un TFG. Modelos como **Mistral 7B Instruct** son accesibles y eficientes para experimentación local, afinados para seguir instrucciones. Modelos más avanzados como **Mistral Small** o **Mistral Large**, disponibles vía API, ofrecen capacidades superiores como ventanas de contexto más amplias (hasta 128k tokens) y soporte nativo para **llamada a funciones (*function calling*)**. Esta última es especialmente relevante para un AI DM, ya que permite al LLM interactuar con lógica de juego externa de forma estructurada.

**Otros Modelos LLM Destacados**

* **OpenAI (GPT-3.5, GPT-4, GPT-4o):** Muy potentes y versátiles , pero son propietarios, con acceso vía API de pago y posibles restricciones de contenido.
* **Anthropic (Claude 3):** Conocidos por su manejo de contextos extensos y enfoque en seguridad. Buenos en creatividad narrativa, pero pueden fallar en mecánicas de juego.
* **Google (Gemini):** Nativamente multimodales, con grandes ventanas de contexto. Algunos modelos muestran buena adherencia a mecánicas.
* **Meta (Llama 2, Llama 3):** Modelos abiertos influyentes, excelentes para *fine-tuning* y desarrollo comunitario, alternativa a APIs propietarias.
* **AI21 Labs (Jurassic-1, Jurassic-2):** Jurassic-1 fue usado por AI Dungeon para superar restricciones de OpenAI y optimizar costes.

La tendencia al *fine-tuning* específico del dominio (como en AI Dungeon o la tesis de ) subraya la importancia de adaptar los modelos base para tareas especializadas como el DM.

#### Infraestructura Necesaria

* **Ejecución Local de LLMs (Mistral 7B):**
  + **GPU:** Mínimo 12GB VRAM (NVIDIA RTX 3060).
  + **RAM:** Mínimo 16GB, recomendable 32GB+.
  + **CPU:** Multinúcleo moderna.
  + **Almacenamiento:** SSD (preferiblemente NVMe) con espacio suficiente ( Mistral 7B base ~13GB).
  + **Software:** Python, PyTorch, Transformers (Hugging Face), CUDA (para NVIDIA GPUs).
* **Alternativa (API):** Usar LLMs vía API ( Mistral AI "La Plateforme" , OpenAI) reduce requisitos de hardware local pero implica costes por uso y dependencia de terceros.

### Estudio y valoración de las alternativas de solución

Existen tres enfoques principales para desarrollar un AI DM:

* **1. Sistemas Basados en Reglas:**
  + **Descripción:** Operan con reglas predefinidas que dictan el comportamiento de la IA ( *storylets* en *Fallen London* o *Hades*).
  + **Fortalezas:** Predecibilidad, control autoral, coherencia garantizada (dentro de las reglas).
  + **Debilidades:** Fragilidad ante entradas no previstas, alto coste de creación/mantenimiento para narrativas complejas, falta de creatividad emergente.

* **2. Sistemas Basados en LLMs Puros:**
  + **Descripción:** Confían principalmente en un LLM para generar la narrativa y controlar el juego a partir de un *prompt* inicial ( AI Dungeon inicial).
  + **Fortalezas:** Alta creatividad y flexibilidad, potencial para narrativas emergentes, menor esfuerzo de autoría explícita inicial.
  + **Debilidades:** Problemas de coherencia a largo plazo , alucinaciones , dificultad para adherirse a reglas del juego , falta de control narrativo preciso (a menudo aceptan cualquier acción del jugador).
* **3. Sistemas Híbridos (LLM + Reglas/Estructura):**
  + **Descripción:** Combinan la flexibilidad de los LLMs con el control de los sistemas basados en reglas.
  + **Fortalezas:** Equilibrio entre creatividad y control, mejora potencial de coherencia y adherencia a reglas, modularidad.
  + **Debilidades:** Mayor complejidad de diseño e implementación, posible carga cognitiva al definir la interacción entre componentes.

### Selección de la solución

Considerando los objetivos y limitaciones de un TFG, se selecciona un **enfoque híbrido ligero** para el desarrollo del prototipo de "IA como Dungeon Master".

**Componentes de la Solución Propuesta:**

* **Modelo de Lenguaje Extenso (LLM) de Mistral AI:**
  + Se encargará de la generación de narrativa (descripciones, trama, diálogos de PNJs) y la interpretación de las acciones del jugador.
  + Se priorizará un modelo de Mistral AI con pesos abiertos y licencia permisiva ( Mistral 7B Instruct ) si el hardware lo permite , o un modelo eficiente vía API ( Mistral Small ) que soporte **llamada a funciones**.
* **Lógica de Juego mediante Llamada a Funciones (*Function Calling*):**
  + Se definirán funciones en Python para gestionar mecánicas de juego estructuradas ( simular tiradas de dados, actualizar Puntos de Vida de PNJs, gestionar inventario básico).
  + El LLM decidirá cuándo invocar estas funciones y con qué argumentos, basándose en la narrativa y la entrada del jugador. El resultado de la función se devolverá al LLM para integrarlo en la historia.

**Justificación de la Selección:**

* **Equilibrio:** Combina la creatividad del LLM para la narrativa con la consistencia de la lógica programada para las mecánicas del juego.
* **Mitigación de Debilidades:** La llamada a funciones ayuda a superar la dificultad de los LLMs puros para adherirse estrictamente a las reglas del juego.
* **Viabilidad para TFG:** Implementar un conjunto limitado de funciones externas es un objetivo técnico alcanzable , permitiendo explorar una característica avanzada de los LLMs modernos sin la complejidad de sistemas híbridos más elaborados (como arquitecturas multiagente completas o sistemas de *storylets* complejos ).
* **Tecnología Relevante:** Utiliza modelos de Mistral AI, alineándose con el interés del proyecto y aprovechando sus avances en modelos abiertos y capacidades como la llamada a funciones.

Esta solución busca un prototipo que sea narrativamente interesante y mecánicamente coherente dentro de un alcance factible para un TFG.

# METODOLOGÍAS USADAS

La selección y aplicación de una metodología de desarrollo de software robusta es un pilar fundamental para la consecución exitosa de cualquier proyecto tecnológico, incluyendo el presente TFG. Una metodología adecuada no solo estructura el proceso de desarrollo, sino que también facilita la gestión de tareas, la optimización de recursos y la consecución de los objetivos planteados dentro de los plazos establecidos. Dada la naturaleza de este TFG, que combina la investigación y aplicación de técnicas de IA con el desarrollo de un sistema software funcional, un bot DM IA para Discord, se requiere un enfoque que equilibre la necesidad de exploración y flexibilidad inherente a la IA con el rigor y la estructura que demanda un proyecto académico.

## Visión General de las Metodologías de Desarrollo de Software

El desarrollo de software es una disciplina que ha evolucionado significativamente, dando lugar a diversos enfoques y marcos de trabajo diseñados para gestionar su complejidad inherente. Las metodologías proporcionan una hoja de ruta estructurada, guiando a los equipos (o en este caso, al desarrollador individual) a través de las distintas fases del ciclo de vida del proyecto, desde la concepción y el análisis de requisitos hasta la implementación, pruebas y documentación final. La elección metodológica impacta directamente en la planificación, la ejecución de tareas, el seguimiento del progreso y la calidad del producto resultante.

Tradicionalmente, se distinguen dos grandes paradigmas: el tradicional (o en cascada) y el Ágil. El primero, representado por modelos como Waterfall, se caracteriza por un enfoque lineal y secuencial, donde cada fase debe completarse antes de iniciar la siguiente. Este modelo presume una definición clara y estable de los requisitos desde el inicio, con un fuerte énfasis en la documentación exhaustiva. En contraste, el paradigma Ágil, con marcos como Scrum o Kanban, promueve un desarrollo iterativo e incremental. Se valora la flexibilidad, la colaboración, la entrega temprana y continua de valor, y la capacidad de adaptación a los cambios, siendo especialmente útil en proyectos con requisitos evolutivos o alto grado de incertidumbre.

## Selección Metodológica para este TFG: El Modelo en Cascada Iterativa

Considerando la naturaleza específica de este TFG –el desarrollo de un DM IA para Discord – que involucra la exploración de modelos de IA (Mistral), técnicas de RAG y la implementación de funcionalidades concretas en un entorno de desarrollo individual y con plazos definidos, se ha optado por un modelo en Cascada Iterativa.

Este modelo combina la claridad estructural del modelo en cascada con la flexibilidad del desarrollo iterativo. El proyecto se descompone en una secuencia de iteraciones, y dentro de cada iteración, se siguen las fases clásicas del modelo en cascada:

* **Planificación y Análisis de Requisitos de la Iteración:** Se definen los objetivos específicos y funcionalidades a desarrollar para esa iteración concreta.
* **Diseño:** Se diseña la arquitectura y los componentes necesarios para cumplir los objetivos de la iteración.
* **Implementación:** Se desarrolla el código de los componentes diseñados.
* **Pruebas:** Se verifica la funcionalidad implementada en la iteración.
* **Entrega (Implícita):** Al final de cada iteración, se obtiene un incremento funcional del software, que es más completo que el de la iteración anterior.

### Plan de Trabajo Detallado como Eje Metodológico

El "Plan de Trabajo TFG Actualizado: DM IA en Discord (Objetivo Finalización: 12 de Junio de 2025)" es el documento central que guiará la ejecución de este proyecto. Este plan desglosa el TFG en fases semanales, cada una con objetivos claros, tareas técnicas específicas y tareas paralelas para la redacción de la memoria.

**Fases Principales del Plan de Trabajo:**

* **Iteración 1 (Semana 1):** Configuración y Bases. Incluye la planificación inicial, el análisis de requisitos para el bot básico, el diseño de su estructura inicial, la implementación de la conexión a Discord y la primera interacción con el LLM, seguido de pruebas de esta funcionalidad elemental.
* **Iteración 2 (Semana 2**): Manejo de Datos (RAG). Se analizan los requisitos para el RAG, se diseña su implementación (carga de dataset, generación de embeddings, índice), se implementan estas funciones y se prueba su efectividad de forma aislada.
* **Siguientes Iteraciones (Semanas 3-5):** Se enfocan en la integración del núcleo del DM, la adición de expansiones como las hojas de personaje y, opcionalmente, el combate básico o refinamiento de UX. Cada una sigue un ciclo de definición, diseño, implementación y prueba para las funcionalidades de esa iteración.

Este enfoque permite una planificación general inicial y una estructura documental clara (características del modelo cascada), mientras que el desarrollo incremental por iteraciones permite abordar la complejidad y la naturaleza exploratoria de los componentes de IA de manera progresiva y controlada. Al final de cada iteración, se pueden realizar ajustes basados en los resultados obtenidos, que informarán la planificación detallada de la siguiente.

El modelado con UML (Lenguaje Unificado de Modelado) se utilizará pragmáticamente en la fase de diseño de aquellas iteraciones que lo requieran para clarificar la arquitectura, las clases o los flujos de interacción. Las estrategias de prueba se aplicarán al final de cada iteración para el incremento desarrollado y de forma más exhaustiva al finalizar todas las iteraciones planificadas (Semana 6).

Diagrama

El contenido generado por IA puede ser incorrecto.Este plan detallado, con sus hitos técnicos y de documentación semanales, asegura un progreso constante y estructurado, mientras que la naturaleza iterativa del desarrollo de IA y las funcionalidades del bot se abordan dentro de cada ciclo semanal y mediante puntos de decisión explícitos.

## Gestión de Requisitos

La gestión de requisitos en el modelo de Cascada Iterativa adoptado se aborda de la siguiente manera:

**Definición Inicial de Alto Nivel:** Los objetivos generales y el alcance del proyecto, definidos en la introducción y en la planificación general del TFG, sirven como los requisitos globales. Estos se establecen en la fase inicial del proyecto.

**Detalle Iterativo de Requisitos:** Al comienzo de cada iteración (cada semana o bloque temático del plan de trabajo), se detallan los requisitos funcionales y no funcionales específicos que se abordarán en ese ciclo. Por ejemplo, las tareas técnicas semanales como "Funciones load\_llm() y generate\_dm\_response()" o "Comandos: !register, !setHP, !sheet" definen explícitamente las funcionalidades a implementar en sus respectivas iteraciones.

**Requisitos No Funcionales:** Aspectos como la usabilidad, el rendimiento y la mantenibilidad se consideran en cada iteración y se planifican mejoras específicas como tareas (ej. "Mejora UX / Refinamiento", "Implementar asyncio.to\_thread para llamadas LLM").

**Adaptación entre Iteraciones:** Aunque cada iteración sigue un flujo secuencial, los resultados y aprendizajes de una iteración pueden influir en la priorización o el detalle de los requisitos de la siguiente. Puntos de decisión explícitos en el plan, como "Decidir alcance de Semana 5", permiten esta adaptación.

## Modelado del Sistema con UML

El uso del Lenguaje Unificado de Modelado (UML) se integra en la fase de diseño de las iteraciones que lo requieran, de forma pragmática para aportar claridad. No se espera una sobrecarga de diagramación, sino un uso enfocado en:

**Diagrama de Casos de Uso (Conceptual):** Para ilustrar las interacciones principales del usuario con el sistema, especialmente en las primeras iteraciones para definir el alcance funcional básico.

**Diagrama de Clases (Simplificado):** Cuando se diseñen módulos con estructuras de datos significativas, como la gestión de hojas de personaje, para modelar las entidades principales.

**Diagrama de Secuencia o Actividad:** Para detallar flujos de interacción complejos dentro de una iteración, como el proceso RAG o la lógica de un comando específico. Estos diagramas servirán de apoyo para el diseño, la implementación y, fundamentalmente, para la documentación clara y visual en la memoria del TFG.

## Estrategias de Prueba y Evaluación

Las pruebas son un componente crucial en cada iteración y al final del proyecto:

Pruebas de Iteración: Al final de cada ciclo iterativo (semanalmente según el plan), se realizarán pruebas funcionales para verificar que los objetivos de la iteración se han cumplido y que el incremento de software funciona según lo esperado. Esto incluye probar los comandos específicos, las integraciones de IA desarrolladas o las funcionalidades de gestión de datos añadidas en esa iteración.

Pruebas del Componente IA: La evaluación de la calidad de respuesta del LLM y la efectividad del RAG será un proceso continuo a lo largo de las iteraciones donde estos componentes se desarrollen y refinen. Esto implicará pruebas cualitativas y ajustes en los prompts o la lógica de recuperación.

Pruebas de Integración: A medida que se completan las iteraciones, se probará la integración de los nuevos componentes con los ya existentes.

Fase de Pruebas Exhaustivas (Semana 6): Una vez completadas todas las iteraciones de desarrollo principales, se dedicará una fase específica a pruebas más completas del sistema integrado. Esto incluirá la verificación del cumplimiento de todos los requisitos, pruebas de usabilidad y la búsqueda de errores residuales.

Plan de Evaluación: La redacción del "plan de Evaluación" (Semana 4) y su refinamiento (Semana 5) formalizarán estas estrategias.

## Justificación del Enfoque Metodológico Seleccionado

El modelo en Cascada Iterativa se considera el más apropiado para este TFG por las siguientes razones:

* **Estructura para Requisitos Académicos:** La naturaleza secuencial dentro de cada iteración y la planificación general inicial facilitan la documentación estructurada y el cumplimiento de los requisitos formales de un TFG, permitiendo definir fases claras de análisis, diseño, implementación y prueba para cada bloque de trabajo.
* **Gestión de la Complejidad y Exploración en IA:** El desarrollo de un sistema con componentes de IA como LLMs y RAG implica un grado de exploración. El enfoque iterativo permite abordar esta exploración en ciclos manejables. Cada iteración permite construir, probar y refinar una parte del sistema, aprendiendo de los resultados antes de pasar a la siguiente, lo cual es crucial para ajustar *prompts* o la lógica del RAG.
* **Desarrollo Incremental y Entrega Progresiva:** El proyecto se construye pieza por pieza. Cada iteración culmina con un incremento funcional del software. Esto permite ver progreso tangible de forma regular (ej. bot básico, luego bot con RAG, luego con hojas de personaje), lo cual es motivador y permite una evaluación temprana de los componentes.
* **Adaptabilidad Moderada:** Si bien no es tan flexible como las metodologías puramente ágiles, el modelo de Cascada Iterativa permite realizar ajustes y refinar los planes para las iteraciones futuras basándose en los resultados y desafíos encontrados en las iteraciones previas. Los puntos de decisión del plan de trabajo son un ejemplo de esto.
* **Adecuación a un Proyecto Individual con Planificación Definida:** El "Plan de Trabajo TFG Actualizado" se alinea naturalmente con una estructura iterativa, donde cada semana o fase representa una iteración con objetivos claros y entregables definidos. Esto facilita la autogestión y el seguimiento del progreso sin la sobrecarga de ceremonias de metodologías ágiles diseñadas para equipos.
* **Mitigación de Riesgos:** Permite abordar los aspectos más riesgosos o técnicamente desafiantes en iteraciones tempranas, proporcionando una mejor comprensión de la viabilidad y ajustando el alcance si es necesario.

En conclusión, la Cascada Iterativa ofrece un equilibrio entre la planificación y estructura necesarias para un TFG, y la flexibilidad y el desarrollo incremental requeridos para un proyecto que involucra la implementación y experimentación con tecnologías de IA. El plan de trabajo detallado sirve como la hoja de ruta para estas iteraciones, guiando el desarrollo progresivo del DM IA

# TECNOLOGÍAS Y HERRAMIENTAS UTILIZADAS EN EL PROYECTO

La selección adecuada de tecnologías y herramientas es crucial para el desarrollo eficiente y exitoso de cualquier proyecto de software, especialmente aquellos que, como el presente TFG, incorporan componentes de IA y requieren la integración de múltiples servicios. A continuación, se detallan las principales tecnologías, lenguajes de programación, librerías y plataformas empleadas para la creación del DM IA para Discord.

## Entorno de Desarrollo y Programación

La base del desarrollo se asienta sobre un conjunto de herramientas y un lenguaje de programación que facilitan la creación, gestión y mantenimiento del código.

### Python (Lenguaje de Programación Principal)

Python ha sido seleccionado como el lenguaje de programación principal debido a su sintaxis clara, su vasta colección de librerías orientadas a la ciencia de datos y la IA, y su robusto soporte para el desarrollo de aplicaciones de red y bots. Su facilidad de aprendizaje y la activa comunidad de desarrolladores lo convierten en una opción idónea para este proyecto. Todas las funcionalidades del DM IA, desde la lógica del bot hasta la interacción con el modelo de lenguaje, serán implementadas en Python.

### Entorno Virtual (venv)

Para gestionar las dependencias del proyecto y asegurar un entorno de desarrollo aislado y reproducible, se utilizará venv. Esta herramienta estándar de Python permite crear entornos virtuales que contienen una instalación específica de Python y un conjunto de paquetes necesarios para el proyecto, evitando conflictos entre las dependencias de diferentes proyectos.

### Git (Control de Versiones)

El control de versiones es esencial para un desarrollo estructurado y para el seguimiento de los cambios en el código fuente. Se utilizará Git, el sistema de control de versiones distribuido más popular, para gestionar el historial del proyecto. Se realizarán *commits* frecuentes para documentar el progreso y facilitar la posible reversión a estados anteriores del código si fuera necesario. El proyecto se alojará en un repositorio Git.

## Interacción con Discord

La funcionalidad principal del proyecto es un bot que opera dentro de la plataforma Discord.

### discord.py (Librería para la API de Discord)

Para la interacción con la API de Discord, se empleará la librería discord.py. Esta librería asíncrona facilita la creación de bots, permitiendo manejar eventos de Discord (como la recepción de mensajes), registrar comandos (incluyendo los modernos *Slash Commands*), enviar mensajes (incluyendo formatos enriquecidos como *Embeds*) y gestionar la lógica general del bot dentro de los servidores y canales de Discord.

### Discord Developer Portal

La creación, configuración y gestión de la aplicación del bot (incluyendo la obtención del token de autenticación) se realizará a través del Portal de Desarrolladores de Discord. Esta plataforma es el punto de partida para registrar el bot y obtener las credenciales necesarias para su funcionamiento.

## Inteligencia Artificial y Modelo de Lenguaje (LLM)

El núcleo del DM IA reside en su capacidad para generar respuestas narrativas y gestionar la partida, lo cual se logrará mediante un Gran Modelo de Lenguaje (LLM).

### Mistral (Gran Modelo de Lenguaje)

Se utilizará un modelo de la familia Mistral como el LLM encargado de generar las respuestas del DM. La elección de Mistral se basa en su reconocida capacidad para la generación de texto coherente y creativo. El acceso al modelo se realizará mediante su API o, si la viabilidad técnica y los recursos lo permiten, podría explorarse una configuración local. Las funciones load\_llm() y generate\_dm\_response() serán las encargadas de gestionar esta interacción.

### Hugging Face transformers (Librería para LLMs)

La librería transformers de Hugging Face será fundamental para interactuar con el modelo Mistral (especialmente si se opta por una carga local o se requiere un manejo más granular que una simple API). Esta librería proporciona una interfaz estandarizada para acceder y utilizar una amplia variedad de modelos preentrenados, facilitando tareas como la carga de modelos, la tokenización del texto y la generación de respuestas.

### torch / tensorflow (Frameworks de Deep Learning)

La librería transformers opera sobre un framework de *deep learning* como PyTorch (torch) o TensorFlow (tensorflow). Se seleccionará uno de estos frameworks según la compatibilidad y los requisitos específicos del modelo Mistral y la librería transformers. Estos frameworks proveen las herramientas fundamentales para la computación tensorial y la ejecución de modelos de IA.

## Sistema de Recuperación Aumentada por Generación (RAG)

Para dotar al LLM de conocimiento específico del contexto de la partida o de un módulo de aventura (como "Fireball"), se implementará un sistema de Recuperación Aumentada por Generación (RAG).

### sentence-transformers (Generación de Embeddings)

La librería sentence-transformers se utilizará para generar *embeddings* (representaciones vectoriales densas) del texto extraído del *dataset* "Fireball". Estos *embeddings* capturan el significado semántico del texto, permitiendo realizar búsquedas por similitud.

### faiss-cpu (Índice de Similitud Vectorial)

FAISS (Facebook AI Similarity Search) en su versión para CPU (faiss-cpu) será la herramienta empleada para construir un índice eficiente a partir de los *embeddings* generados. Este índice permitirá realizar búsquedas rápidas de similitud semántica, encontrando los fragmentos de texto del *dataset* "Fireball" más relevantes para una consulta o contexto dado. La función build\_or\_load\_index() gestionará la creación y carga de este índice, y search\_relevant\_info() implementará la búsqueda.

### Dataset "Fireball" (Fuente de Conocimiento)

El *dataset* "Fireball" (cuyo contenido específico se definirá en el desarrollo) actuará como la base de conocimiento externa para el sistema RAG. La función load\_fireball\_dataset() se encargará de cargar, procesar y extraer el texto relevante de este conjunto de datos para su posterior indexación.

## Gestión de Datos y Persistencia

Para funcionalidades como las hojas de personaje, se requerirá un sistema de persistencia de datos.

### SQLite (Base de Datos Local)

Se utilizará SQLite como sistema de gestión de bases de datos relacional ligera para almacenar y gestionar información como las hojas de personaje de los jugadores. SQLite es una excelente opción para aplicaciones que requieren una base de datos embebida, sin necesidad de un servidor de base de datos separado. Se integrará mediante la librería sqlite3 de Python.

### datasets (Manejo de Conjuntos de Datos)

La librería datasets de Hugging Face podría ser utilizada para facilitar la carga y manipulación del *dataset* "Fireball" u otros conjuntos de datos que se puedan emplear durante el desarrollo o la evaluación del sistema.

## Herramientas y Librerías Auxiliares

Diversas herramientas y librerías complementarias facilitarán el desarrollo.

### python-dotenv (Gestión de Variables de Entorno)

Para manejar de forma segura información sensible como el token del bot de Discord o claves de API para el modelo Mistral, se utilizará python-dotenv. Esta librería permite cargar variables de entorno desde un archivo .env, manteniéndolas separadas del código fuente.

### asyncio.to\_thread (Programación Asíncrona)

Dado que las llamadas a modelos de lenguaje pueden ser bloqueantes y discord.py es una librería asíncrona, se empleará asyncio.to\_thread (disponible en Python 3.9+) o un mecanismo similar para ejecutar las operaciones de inferencia del LLM en un hilo separado. Esto evitará que el bot se bloquee mientras espera la respuesta del modelo, mejorando la responsividad.

## Documentación y Gestión del Proyecto

La documentación y una planificación clara son esenciales para el TFG.

### README.md (Documentación del Repositorio)

Se creará un archivo README.md detallado en el repositorio Git. Este archivo incluirá una descripción del proyecto, instrucciones claras para la configuración del entorno, la instalación de dependencias y la ejecución del bot.

### Plan de Trabajo Semanal (Herramienta de Seguimiento)

El "Plan de Trabajo TFG Actualizado" con sus tareas semanales y objetivos actúa como la principal herramienta de planificación y seguimiento del progreso del proyecto. Aunque no es una herramienta software *per se*, su estructura y el seguimiento disciplinado de las tareas son fundamentales para la gestión del tiempo y el cumplimiento de los plazos.

Este conjunto de tecnologías y herramientas ha sido seleccionado para proporcionar una base sólida y moderna para el desarrollo del DM IA, equilibrando la potencia necesaria para las tareas de IA con la facilidad de uso y la robustez requeridas para un proyecto de esta envergadura

# ESTIMACIÓN DE RECURSOS Y PLANIFICACIÓN

La selección adecuada de tecnologías y herramientas es crucial para el desarrollo eficiente y exitoso de cualquier proyecto de software, especialmente aquellos que, como el presente TFG, incorporan componentes de IA y requieren la integración de múltiples servicios. A continuación, se detallan las principales tecnologías, lenguajes de programación, librerías y plataformas empleadas para la creación del DM IA para Discord.

## Estimación del Esfuerzo del Proyecto

La estimación del esfuerzo en un proyecto de software busca predecir la cantidad de trabajo requerida para su desarrollo. Existen diversos modelos y técnicas para esta tarea, como el análisis por Puntos de Función o modelos algorítmicos como COCOMO**.**

* Análisis por Puntos de Función (FPA): Esta técnica mide el tamaño funcional del software desde la perspectiva del usuario, basándose en las funciones que el sistema proporciona (entradas, salidas, consultas, archivos lógicos internos y archivos de interfaz externos). Si bien un análisis formal de Puntos de Función ofrece una medida objetiva del tamaño, su aplicación detallada puede ser compleja para la escala y naturaleza de un TFG como el presente. No obstante, el "Plan de Trabajo TFG Actualizado" implícitamente desglosa funcionalidades clave a entregar en cada fase ( bot básico, sistema RAG, gestión de hojas de personaje, mecánicas de combate), lo que constituye una forma de identificar y dimensionar las entregas funcionales del proyecto.
* COCOMO (Constructive Cost Model):Es un modelo algorítmico que estima el esfuerzo y la duración del desarrollo de software basándose principalmente en el tamaño del producto (expresado en líneas de código o KLOC) y diversos factoresde coste (atributos del producto, del hardware, del personal y del proyecto). La aplicación rigurosa de COCOMO, especialmente en sus versiones Intermedia o Detallada, requiere una calibración y una estimación precisa del tamaño que pueden ser difíciles de obtener en las etapas iniciales de un TFG con componentes de IA, donde la exploración y la experimentación son significativas.

Para el presente TFG, y considerando su naturaleza individual y la metodología híbrida adoptada, la estimación del esfuerzo se basa primordialmente en un desglose detallado de tareas y una asignación temporal específica para cada fase del proyecto, tal como se articula en el "Plan de Trabajo TFG Actualizado" (ver sección 6.2). Este enfoque pragmático permite una planificación y seguimiento directo del esfuerzo semanal.

Tomando como referencia el plan de 6.5 semanas de desarrollo intensivo (desde el 28 de Abril hasta el 12 de Junio de 2025), y asumiendo una dedicación promedio de 25 horas semanales (considerando la naturaleza intensiva del plan y la necesidad de cubrir tanto tareas técnicas como de redacción de memoria), el esfuerzo directo estimado para estas fases principales sería:

Esfuerzo desarrollo​=6.5 semanas×25 horas/semana=162.5 horas

A esto se suma el tiempo dedicado a la configuración inicial, la investigación profunda para el estado del arte, la redacción final intensiva y el período de buffer, lo que podría llevar la estimación total de dedicación del estudiante a un rango de 250-350 horas. Esta estimación es coherente con la carga de trabajo esperada para un TFG que involucra desarrollo práctico e investigación. Una cuantificación más formal, si fuera requerida, podría alinearse con los créditos ECTS asignados al TFG (donde 1 ECTS suele equivaler a 25-30 horas de trabajo del estudiante).

A esto se suma el tiempo dedicado a la configuración inicial, la investigación profunda para el estado del arte, la redacción final intensiva y el período de buffer, lo que podría llevar la estimación total de dedicación del estudiante a un rango de 250-350 horas. Esta estimación es coherente con la carga de trabajo esperada para un TFG que involucra desarrollo práctico e investigación. Una cuantificación más formal, si fuera requerida, podría alinearse con los créditos ECTS asignados al TFG (donde 1 ECTS suele equivaler a 25-30 horas de trabajo del estudiante).

## Realizar una planificación temporal del proyecto (diagrama de Pert, diagrama de Gantt, etc)

L a planificación temporal es crucial para asegurar que el proyecto se complete dentro de los plazos establecidos. Para este TFG, la planificación se articula a través del "Plan de Trabajo TFG Actualizado: DM IA en Discord", con fecha de inicio el 28 de Abril de 2025 y fecha de entrega al tutor el 16 de Junio de 2025.

El plan se estructura en las siguientes fases principales, derivadas de las agrupaciones semanales:

**Fase 1: Configuración, Bases y Fundamentación Teórica (Semana 1: 28 Abril - 4 Mayo)**

* + *Actividades Principales:* Configuración del entorno, bot básico, primera interacción con Mistral, inicio de Introducción y Estado del Arte.
  + *Duración:* 1 semana.

**Fase 2: Implementación del Sistema RAG y Diseño Metodológico (Semana 2: 5 Mayo - 11 Mayo)**

* + *Actividades Principales:* Carga dataset Fireball, construcción sistema RAG, avance/finalización Estado del Arte, borrador Metodología y Diseño.
  + *Duración:* 1 semana.

**Fase 3: Integración del Núcleo del DM e Implementación Inicial (Semana 3: 12 Mayo - 18 Mayo)**

* + *Actividades Principales:* Conexión RAG+LLM, historial básico, finalización borrador Metodología y Diseño, inicio sección Implementación.
  + *Duración:* 1 semana.

**Fase 4: Desarrollo de Funcionalidades Extendidas y Plan de Evaluación (Semanas 4-5: 19 Mayo - 1 Junio)**

* + *Actividades Principales:* Implementación de expansión 1 (Hojas de Personaje), decisión sobre expansión 2 o refinamiento, implementación de expansión 2 o refinamiento UX. Detalle de implementación en memoria, borrador y refinamiento plan de Evaluación, inicio borrador Resultados.
  + *Duración:* 2 semanas.

**Fase 5: Pruebas Exhaustivas, Congelación de Código y Redacción Avanzada (Semana 6: 2 Junio - 8 Junio)**

* + *Actividades Principales:* Testeo intensivo, corrección de errores, limpieza y congelación de código. Redacción avanzada de Resultados, Conclusiones y Trabajo Futuro.
  + *Duración:* 1 semana.

**Fase 6: Finalización de Memoria y Preparación de Entrega (9 Junio - 12 Junio)**

* + *Actividades Principales:* Redacción de Resumen, Bibliografía, Anexos. Revisión global intensiva. Preparación de código y demo.
  + *Duración:* 4 días.

**Fase 7: Buffer, Últimos Retoques y Entrega (13 Junio - 16 Junio)**

* + *Actividades Principales:* Margen para imprevistos, revisión final y entrega al tutor.
  + *Duración:* 3-4 días.

## Realizar una valoración de la dedicación y el coste económico

La valoración de la dedicación y el coste económico de un TFG permite contextualizar el esfuerzo invertido y los recursos consumidos, aunque muchos de estos no supongan un desembolso directo para el estudiante.

**Dedicación (Esfuerzo del Estudiante):** Como se estimó en la sección 6.1, la dedicación total del estudiante para este proyecto se sitúa en un rango de **250-350 horas**. Esta cifra incluye todas las actividades: investigación, aprendizaje de nuevas tecnologías (Mistral, RAG, discord.py), diseño, codificación, pruebas, y la redacción exhaustiva de la memoria del TFG. El "Plan de Trabajo TFG Actualizado" refleja una alta intensidad de dedicación semanal para cumplir con los plazos establecidos.

**Coste Económico Estimado:**

Se desglosan los costes estimados en diferentes categorías:

* **Costes de Hardware:**
  + Se utilizará el ordenador personal del estudiante (Portátil/Sobremesa).
  + *Valor de Referencia (Amortización/Coste):* 2000,00 € *(Este valor es un ejemplo, el estudiante debe ajustarlo a su equipo).*
* **Costes de Software:**
  + Sistema Operativo (Linux, Windows): Gratuito (Linux) o preinstalado.
  + Python y sus librerías (discord.py, transformers, sentence-transformers, faiss-cpu, sqlite3, etc.): Código abierto, gratuitas. Coste: 0,00 €.
  + Modelo LLM Mistral: Se asume el uso de una posible capa gratuita de API o un modelo local para fines académicos. Coste: 0,00 € (para el TFG).
  + Herramientas de desarrollo (Git, IDE como VS Code/Spyder): Gratuitas. Coste: 0,00 €.
  + *Total Coste de Software:* **0,00 €**.
* **Otros Costes:**
  + Conexión a Internet (aprox. 2 meses de desarrollo principal): 40,00 €/mes \* 2 meses = 80,00 €.
  + Consumo eléctrico estimado: 10,00 €/mes \* 2 meses = 20,00 €.
  + *Total, Otros Costes:* **100,00 €**.
* **Costes de Desarrollo (Valoración del esfuerzo del estudiante):**
  + Este es un coste "nocional" que refleja el valor del tiempo invertido por el estudiante si se contratase a un perfil junior.
  + Horas totales estimadas: 300 horas (tomando un valor medio del rango o basado en ECTS).
  + Tarifa horaria simbólica (Programador Junior/Becario): 12,00 €/hora.
  + *Coste de Desarrollo (Estudiante):* 300 horas \* 12,00 €/hora = **3.600,00 €**.
* **Coste Total Estimado del Proyecto:**
  + Coste Hardware: 2000,00 €
  + Coste Software: 0,00 €
  + Otros Costes: 100,00 €
  + Coste Desarrollo (Estudiante): 3.600,00 €
  + **Subtotal:** 5.700,00 €
  + Costes Indirectos (e.j. , 15-20% para gastos generales, material de oficina, etc., si aplica académicamente): 20% de 5.700,00 € = 900,00 €
  + **COSTE TOTAL ESTIMADO (Valoración Académica):** **5.400,00 €**

**Valoración General:** El principal "coste" de este TFG es la intensiva dedicación de tiempo y esfuerzo intelectual por parte del estudiante. Los costes monetarios directos son mínimos gracias al uso de hardware personal y software de código abierto o con licencias educativas/gratuitas. La planificación temporal es ajustada, lo que exige un alto grado de compromiso y disciplina. Las "Claves para este plan ajustado" (disciplina con la escritura, priorización constante y comunicación con el tutor) son fundamentales para mitigar los riesgos de desviación. La inclusión de un periodo de buffer y puntos de decisión estratégicos en el plan son medidas clave para gestionar la incertidumbre inherente al desarrollo con IA y el alcance del proyecto. La viabilidad del proyecto, dentro de esta estimación de recursos y planificación, depende críticamente de la ejecución diligente del plan de trabajo.

# DESARROLLO DEL CONTENIDO DEL PROYECTO

Este capítulo constituye el corazón técnico del TFG. En él se describe en detalle el proceso seguido para la concepción, diseño e implementación del DM basado en IA para la plataforma Discord. Se abordarán desde el análisis de los requisitos hasta las soluciones técnicas aportadas, la configuración del entorno y la implementación de cada uno de los componentes clave del sistema, siguiendo la planificación temporal establecida.

## Análisis de Requisitos y Alcance Funcional

Antes de acometer el diseño y la implementación, es fundamental establecer una comprensión clara de lo que el sistema debe hacer. Esta sección detalla los requisitos funcionales y no funcionales del DM IA.

### Objetivos y Alcance General

* **Objetivo Principal:** Desarrollar un bot de Discord que actúe como un DM IA, capaz de dirigir una partida de rol narrativa, utilizando un Gran Modelo de Lenguaje (LLM) como Mistral y un sistema de Recuperación Aumentada por Generación (RAG) para contextualizar sus respuestas con información específica de una aventura (dataset "Fireball").
* **Alcance:**
  + Interacción principal a través de comandos de texto en Discord.
  + Generación de narrativa y respuestas del DM por parte del LLM.
  + Uso de RAG para mejorar la coherencia y especificidad del LLM.
  + Gestión básica del historial de conversación.
  + Implementación de mecánicas de juego como hojas de personaje.
  + (Opcional, según progreso) Implementación de un sistema de combate básico.
  + Mejoras en la experiencia de usuario (UX) mediante formatos de mensaje enriquecidos y comandos modernos.

### Actores del Sistema

* **Jugador:** Usuario de Discord que interactúa con el DM IA para participar en la partida de rol.
* **DM IA (Bot de Discord):** El sistema desarrollado, que interpreta las acciones de los jugadores, genera la narrativa, gestiona el estado del juego y aplica las mecánicas.

### Requisitos Funcionales

Los requisitos funcionales definen las tareas específicas que el sistema debe poder realizar:

* **RF01:** El bot deberá conectarse y permanecer operativo en un servidor de Discord.
* **RF02:** El bot deberá responder a un comando básico de prueba (ej. !ping) para verificar su actividad.
* **RF03:** El sistema deberá ser capaz de cargar y utilizar un LLM (Mistral) para generar respuestas narrativas en el rol de DM.
* **RF04:** El sistema deberá implementar un mecanismo RAG que:
  + Cargue y procese un conjunto de datos de aventura específico ("Fireball").
  + Genere *embeddings* del contenido del dataset.
  + Construya un índice de similitud (FAISS) con dichos *embeddings*.
  + Recupere información relevante del índice basada en la entrada del jugador o el contexto actual.
* **RF05:** Las respuestas del LLM deberán incorporar la información recuperada por el sistema RAG y el historial de conversación para mejorar la contextualización.
* **RF06:** El sistema deberá mantener un historial básico de la conversación de la partida actual.
* **RF07:** El bot deberá permitir limpiar el historial de conversación mediante un comando (ej. !clearhistory).
* **RF08:** El sistema deberá permitir la gestión de Hojas de Personaje para los jugadores, incluyendo:
  + Registro de nuevos personajes (ej. !register).
  + Modificación de atributos básicos (ej. Puntos de Vida con !setHP).
  + Visualización de la hoja de personaje (ej. !sheet).
  + Almacenamiento persistente de los datos de personaje (SQLite).
* **RF09 (Opcional):** El sistema podría gestionar un sistema de combate básico, incluyendo:
  + Inicio y unión a un combate (ej. !startCombat, !joinCombat).
  + Gestión de turnos e iniciativa (ej. !initiative, !nextTurn).
  + Lógica básica del DM para acciones de monstruos/enemigos.
* **RF10:** El bot deberá permitir la interacción mediante comandos de prefijo (ej. !comando) y, posteriormente, migrar a *Slash Commands* (/comando) para una mejor UX.
* **RF11:** El bot deberá utilizar formatos de mensaje enriquecidos de Discord (*Embeds*) para presentar información clave de forma clara y atractiva.

### Requisitos No Funcionales

Los requisitos no funcionales describen las cualidades y restricciones del sistema:

* **RNF01 (Usabilidad):** La interacción con el bot deberá ser intuitiva para usuarios familiarizados con Discord y juegos de rol. Las respuestas del DM IA deben ser claras y fáciles de entender.
* **RNF02 (Rendimiento):** El tiempo de respuesta del bot a las acciones del jugador, especialmente aquellas que involucran al LLM, deberá ser razonable para no interrumpir el flujo de la partida (se buscará optimizar mediante asyncio.to\_thread).
* **RNF03 (Coherencia Narrativa):** El DM IA deberá mantener un grado aceptable de coherencia en la narrativa generada y en la aplicación de la información contextual.
* **RNF04 (Mantenibilidad):** El código fuente deberá estar bien estructurado, comentado y ser modular para facilitar futuras modificaciones o expansiones.
* **RNF05 (Configurabilidad):** El token del bot y otras configuraciones sensibles (ej. claves API de Mistral) deberán gestionarse de forma segura (ej. mediante archivos .env).
* **RNF06 (Estabilidad):** El bot deberá manejar errores de forma adecuada sin caídas inesperadas.

## Diseño del Sistema

Esta sección detalla la arquitectura y el diseño de los componentes principales del DM IA. Se recomienda el uso de diagramas UML (como diagramas de componentes, clases simplificados, secuencia o actividad) para ilustrar los conceptos aquí descritos.

### Arquitectura General del Sistema

* Describir los componentes principales y cómo interactúan:
  + **Interfaz de Usuario (Discord):** Plataforma a través de la cual el jugador interactúa.
  + **Bot de Discord (discord.py):** Recibe los comandos del usuario, procesa la lógica de la aplicación, interactúa con los otros módulos y envía respuestas a Discord.
  + **Módulo de Procesamiento de Lenguaje Natural (PLN):**
    - **Motor LLM (Mistral):** Responsable de la generación de texto narrativo y decisiones del DM.
    - **Módulo RAG:** Proporciona contexto relevante al LLM. Incluye el *Dataset Loader*, *Embedding Generator*, *Vector Store (FAISS)* y *Retriever*.
  + **Módulo de Gestión de Datos:**
    - **Base de Datos (SQLite):** Almacena datos persistentes como las hojas de personaje.
  + **Módulo de Lógica de Juego:** Gestiona el historial de conversación, mecánicas de personaje y (opcionalmente) combate.
* Incluir un **diagrama de arquitectura general** mostrando estos componentes y sus flujos de comunicación principales.

### Diseño del Bot de Discord

* **Estructura de Comandos:** Detallar cómo se manejarán los comandos (inicialmente con prefijo, luego migración a *Slash Commands*).
* **Manejo de Eventos:** Explicar cómo se utilizan los eventos de discord.py (ej. on\_message para comandos de prefijo, interacciones para *Slash Commands*, on\_ready).
* **Organización del Código:** (Ej. uso de Cogs en discord.py para modularidad, si aplica).
* **Formato de Respuestas:** Diseño del uso de *Embeds* para presentar la información del DM de manera atractiva.

### Diseño del Módulo de Inteligencia Artificial

#### Interacción con el LLM (Mistral)

* **Estrategia de Acceso:** Detallar si se usará API o un modelo local, y las consideraciones de cada enfoque.
* **Diseño de *Prompts*:** Este es un punto crucial. Explicar la estructura base del *prompt* enviado a Mistral. Detallar cómo se incorporará el contexto del RAG, el historial de la conversación, y la información del personaje (si aplica) para guiar al LLM a actuar como DM. Ilustrar el proceso iterativo de refinamiento de *prompts*.
* **Gestión del Historial de Conversación:** Cómo se almacenará (ej. dict en memoria), cuál será su tamaño límite, y cómo se incluirá en los *prompts*.

#### Diseño del Sistema RAG

* **Flujo de Datos del RAG:** Desde la consulta del usuario/contexto del juego hasta la recuperación de fragmentos relevantes.
* **Procesamiento del Dataset "Fireball":** Formato esperado, cómo se extraerá el texto y se segmentará (si es necesario) para la generación de *embeddings*.
* **Generación y Almacenamiento de *Embeddings*:** Uso de sentence-transformers y cómo/dónde se guardará el índice FAISS para su carga posterior (build\_or\_load\_index).
* **Lógica de Recuperación:** Cómo se formularán las consultas al índice FAISS y cómo se seleccionarán los fragmentos más relevantes (search\_relevant\_info).

### Diseño del Módulo de Gestión de Datos (Hojas de Personaje)

* **Esquema de la Base de Datos SQLite:** Definir las tablas (ej. Personajes, Atributos\_Personaje), columnas y relaciones necesarias para almacenar la información de las hojas de personaje.
* **Operaciones CRUD:** Describir cómo se realizarán las operaciones de Crear, Leer, Actualizar y Borrar (CRUD) sobre los datos de los personajes mediante la librería sqlite3.

### (Opcional) Diseño del Módulo de Combate

* Si se implementa, describir las estructuras de datos para gestionar el estado del combate (participantes, iniciativa, salud de enemigos, etc.).
* Lógica de los comandos de combate y flujo de turnos.

## Implementación y Soluciones Aportadas

Esta sección describe el proceso de construcción del sistema, detallando la implementación de cada componente según el plan de trabajo. Se recomienda incluir fragmentos de código relevantes y explicar las decisiones técnicas tomadas.

### Configuración del Entorno de Desarrollo (Semana 1)

* Breve resumen de la configuración del entorno Python con venv.
* Listado de librerías principales instaladas y sus versiones (ej. requirements.txt).
* Configuración inicial del repositorio Git.

### Desarrollo del Bot Básico e Integración Inicial con LLM (Semana 1)

* Implementación de la conexión del bot a Discord usando discord.py.
* Código del comando !ping.
* Implementación de las funciones load\_llm() y generate\_dm\_response() para interactuar con Mistral (API o local).
* Ejemplos de las primeras pruebas de generación de texto y los *prompts* iniciales utilizados.

### Implementación del Sistema RAG (Semana 2)

* **load\_fireball\_dataset():** Detalles de la implementación para cargar y preprocesar el texto del dataset "Fireball".
* **build\_or\_load\_index():** Código y explicación de cómo se generan los *embeddings* con sentence-transformers y se construye/guarda/carga el índice FAISS.
* **search\_relevant\_info():** Implementación de la función de búsqueda por similitud.
* Resultados de pruebas aisladas del sistema RAG (ej. consultas de ejemplo y fragmentos recuperados).

### Integración del Núcleo del DM (RAG + LLM + Historial) (Semana 3)

* Modificaciones en on\_message (o el manejador de comandos) para integrar el flujo: Input del Usuario -> RAG -> *Prompt* Aumentado -> LLM -> Output a Discord.
* **Evolución del Diseño de *Prompts*:** Mostrar ejemplos concretos de cómo se refinaron los *prompts* para incorporar el contexto RAG y el historial, y el impacto en la calidad de las respuestas del DM.
* Implementación del sistema de historial de conversación y del comando !clearhistory.

### Desarrollo de Expansiones (Semanas 4-5)

#### Implementación de Hojas de Personaje

* Creación de las tablas en la base de datos SQLite (mostrar esquemas SQL o código de creación).
* Implementación de los comandos de gestión de personajes (!register, !setHP, !sheet) y su interacción con la base de datos.
* (Si aplica) Cómo se integró la información de los personajes en los *prompts* del LLM.

#### (Opcional) Implementación de Combate Básico o Refinamiento UX

* **Combate:** Detallar las estructuras de datos, la lógica de los comandos (!startCombat, etc.) y el manejo básico de la IA de los monstruos.
* **Refinamiento UX:**
  + Ejemplos de uso de *Embeds* para mejorar la presentación.
  + Proceso de migración a *Slash Commands* y ejemplos de su implementación.
  + Implementación de asyncio.to\_thread para las llamadas al LLM y su impacto en la responsividad.

### Desafíos y Decisiones Clave durante la Implementación

* Describir cualquier desafío técnico significativo encontrado durante el desarrollo (ej. limitaciones de la API, problemas de rendimiento, dificultades en el diseño de *prompts*, gestión de la coherencia del RAG).
* Explicar las soluciones implementadas y justificar las decisiones técnicas importantes que se tomaron (ej. por qué se eligió una estructura de *prompt* específica, cómo se optimizó una función, etc.).

## Pruebas y Verificación del Sistema

Una vez completado el desarrollo principal (hacia la Semana 6), se debe realizar una fase de pruebas exhaustivas para asegurar la calidad y el correcto funcionamiento del DM IA.

### Estrategia y Casos de Prueba

* Detallar el plan de pruebas ejecutado:
  + **Pruebas Funcionales:**
    - Tabla de casos de prueba para cada comando del bot (entrada esperada, acción, salida esperada).
    - Pruebas de secuencias de interacción (ej. crear personaje, modificarlo, usarlo en una consulta al DM).
  + **Pruebas del Componente IA:**
    - *Calidad del RAG:* Consultas de prueba al RAG para verificar la relevancia de la información recuperada.
    - *Calidad de Generación del LLM:* Escenarios de prueba con diferentes *prompts* (con y sin contexto RAG/historial) para evaluar la coherencia, creatividad y adecuación de las respuestas del DM.
    - *Adherencia a Reglas (si aplica):* Pruebas específicas para las mecánicas de combate.
  + **Pruebas de Usabilidad:**
    - Evaluación de la facilidad de uso de los comandos.
    - Claridad y presentación de la información (uso de *Embeds*).
    - (Idealmente) Pruebas con algunos usuarios beta (amigos, compañeros).
  + **Pruebas de Casos Límite y Errores:** Probar entradas inesperadas, comandos malformados, etc., para verificar el manejo de errores.

### Resultados de las Pruebas y Corrección de Errores

* Presentar un resumen de los resultados obtenidos durante las pruebas.
* Mencionar los *bugs* o problemas más significativos que se encontraron y cómo fueron solucionados.
* Explicar cómo se refinaron los *prompts* del LLM o la lógica del RAG basándose en los resultados de las pruebas para mejorar la calidad del DM.

### Verificación del Cumplimiento de Requisitos

* Concluir esta sección evaluando en qué medida el sistema desarrollado cumple con los requisitos funcionales (RF) y no funcionales (RNF) identificados en la sección 7.1. Argumentar cada punto.

# **DESPLIEGUE Y PRUEBA DE LA SOLUCIÓN**

Una vez finalizada la fase de desarrollo e implementación detallada en el capítulo anterior, este capítulo se centra en el proceso de pruebas llevado a cabo para asegurar la calidad y el correcto funcionamiento del DM IA, y la evaluación de su desempeño y utilidad. Adicionalmente, se discuten aspectos cruciales como la escalabilidad, las posibilidades de extensión futura, el mantenimiento previsto y una propuesta para la guía de los usuarios finales del DM IA en Discord. Aunque el "despliegue" en el contexto de este TFG se refiere principalmente a la ejecución del bot en un entorno de pruebas (un servidor de Discord), se abordarán las consideraciones para un uso funcional.

## Plan de Pruebas

El objetivo fundamental de esta fase es verificar que el sistema DM IA cumple con los requisitos definidos, funciona de manera robusta y ofrece una experiencia de usuario satisfactoria.

### Objetivos de las Pruebas

* Validar que todas las funcionalidades implementadas (comandos del bot, interacción con LLM, sistema RAG, gestión de hojas de personaje, etc.) operan según lo especificado en los requisitos funcionales.
* Asegurar la correcta integración entre los distintos componentes del sistema (bot de Discord, LLM Mistral, RAG, base de datos SQLite).
* Evaluar la calidad de las respuestas generadas por el DM IA en términos de coherencia narrativa, relevancia contextual y creatividad.
* Identificar y corregir errores (bugs) en el software.
* Verificar el cumplimiento de los requisitos no funcionales clave, como la usabilidad y un rendimiento aceptable.

### Alcance de las Pruebas

El alcance de las pruebas cubrirá los siguientes aspectos del sistema:

* **Funcionalidad del Bot de Discord:** Conexión, recepción y procesamiento de comandos, envío de mensajes y *Embeds*.
* **Módulo de Inteligencia Artificial:**
  + Interacción con el LLM (Mistral): Calidad y pertinencia de la generación de texto.
  + Sistema RAG: Precisión en la recuperación de información del dataset "Fireball" y su impacto en las respuestas del LLM.
  + Gestión del historial de conversación.
* **Mecánicas de Juego:** Funcionalidad de las hojas de personaje (creación, modificación, visualización) y su persistencia en la base de datos SQLite. (Si se implementó, también las mecánicas de combate básico).
* **Experiencia de Usuario (UX):** Facilidad de interacción, claridad de los mensajes y comandos.

### Tipos de Pruebas y Estrategia

Se aplicará una combinación de los siguientes tipos de pruebas:

* **Pruebas Funcionales (Caja Negra):**
  + **Estrategia:** Se diseñarán casos de prueba basados en los requisitos funcionales (ver Sección 7.1.3). Cada caso de prueba especificará una acción a realizar, la entrada proporcionada y el resultado esperado.
  + **Ejemplo:** *Verificación de comandos de gestión de personaje, comandos de interacción con el DM, comando !clearhistory, etc.*
  + Se creará una tabla de casos de prueba que incluirá: ID del Caso, Descripción, Pasos a Realizar, Datos de Entrada, Resultado Esperado, Resultado Obtenido, Estado (Pasa/Falla), Observaciones.
* **Pruebas de Integración:**
  + **Estrategia:** Se verificarán las interfaces y la comunicación entre los componentes principales:
    - Integración Bot (discord.py) ↔ Motor LLM (Mistral).
    - Integración Bot ↔ Sistema RAG (recuperación de contexto).
    - Integración Sistema RAG ↔ Motor LLM (uso del contexto recuperado en *prompts*).
    - Integración Bot ↔ Base de Datos SQLite (para hojas de personaje).
  + **Ejemplo:** *Probar que una acción del jugador que requiere información del RAG se traduce en un prompt aumentado correctamente y una respuesta contextualizada del LLM.*
* **Pruebas Específicas del Componente IA:**
  + **Estrategia:** Dada la naturaleza de los sistemas de IA generativa, estas pruebas combinarán evaluaciones cuantitativas simples (donde sea posible) y, principalmente, cualitativas.
  + **Calidad del RAG:** Se realizarán consultas de prueba representativas al sistema RAG para evaluar la relevancia de los fragmentos del dataset "Fireball" recuperados.
  + **Calidad de Generación del LLM:**
    - Se utilizarán escenarios de juego de prueba para evaluar la coherencia narrativa, la creatividad, la adecuación al rol de DM y la correcta utilización del contexto RAG y el historial.
    - Se podría definir una pequeña rúbrica con criterios como: "Coherencia", "Relevancia", "Creatividad", "Adherencia al tono de DM".
  + **Robustez del Prompting:** Probar cómo el sistema maneja entradas de jugador ambiguas o inesperadas.
* **Pruebas de Usabilidad (UX):**
  + **Estrategia:** Se realizarán pruebas desde la perspectiva del jugador, evaluando la facilidad para entender cómo interactuar con el bot, la claridad de los comandos y la presentación de la información (uso de *Embeds*, migración a *Slash Commands*).
  + **Método:** Autoevaluación y, si es posible, pruebas informales con algunos compañeros o amigos que actúen como jugadores beta para recoger feedback.
* **Pruebas de Rendimiento (Observacionales):**
  + **Estrategia:** Se observará el tiempo de respuesta del bot a diferentes tipos de comandos, especialmente aquellos que involucran la llamada al LLM y al sistema RAG. El objetivo es asegurar que la latencia sea aceptable para una partida fluida. Se verificará el correcto funcionamiento de asyncio.to\_thread para evitar bloqueos.

### Entorno de Pruebas

* Las pruebas se ejecutarán en un servidor de Discord dedicado para desarrollo y pruebas, donde el bot DM IA esté activo.
* El dataset "Fireball" y la base de datos SQLite estarán configurados en el entorno local donde se ejecute el script del bot.
* El acceso al LLM Mistral se realizará según la configuración definida (API o local).

### Criterios de Aceptación

El sistema se considerará aceptable si:

* Al menos el 95% de los casos de prueba funcionales críticos son superados.
* La integración entre componentes funciona sin errores bloqueantes.
* Las respuestas del DM IA demuestran un nivel aceptable de coherencia y relevancia contextual en los escenarios de prueba.
* La usabilidad del bot es considerada buena o aceptable por los probadores (o en autoevaluación).
* No existen errores críticos que impidan el flujo normal de una partida de prueba.

## Indicadores del Desempeño y/o Utilidad

Para evaluar la efectividad y calidad del DM IA desarrollado, se proponen los siguientes indicadores:

* **Indicadores de Desempeño Técnico:**
  + **Tiempo de Respuesta Medio:**
    - Para comandos directos del bot (ej. !sheet): medido en milisegundos/segundos.
    - Para interacciones con el DM IA (que involucran RAG y LLM): medido en segundos. Se registrará este tiempo para un conjunto de interacciones de prueba.
  + **Tasa de Éxito de Procesamiento de Comandos:** Porcentaje de comandos de usuario que son interpretados y procesados correctamente por el bot sin generar errores internos.
  + **Relevancia de Recuperación del RAG (Cualitativa):** Para un conjunto de N escenarios de prueba, porcentaje de veces que la información recuperada por el RAG se considera "altamente relevante" o "relevante" para el contexto de la consulta (evaluación manual).
  + **Estabilidad del Bot:** Número de reinicios inesperados o caídas durante un período de prueba prolongado (ej. una sesión de juego de 1-2 horas).
* **Indicadores de Utilidad y Calidad del DM IA (principalmente cualitativos):**
  + **Coherencia Narrativa:** Evaluación mediante una escala (ej. de 1 a 5) o descriptiva de la capacidad del DM IA para mantener una trama lógica y sin contradicciones evidentes durante una sesión de prueba.
  + **Calidad de la Inmersión y *Engagement*:** Feedback subjetivo (ej. a través de una breve encuesta o discusión con jugadores de prueba) sobre cuán interesante, creativa y envolvente resulta la experiencia de juego dirigida por el DM IA.
  + **Adaptabilidad Contextual:** Capacidad del DM IA para utilizar adecuadamente la información del historial y del RAG para generar respuestas pertinentes al estado actual de la partida.
  + **Facilidad de Interacción (Usabilidad):** Medida a través de la observación de cómo los jugadores de prueba interactúan con los comandos, y su feedback sobre la claridad e intuitivita.
  + **(Si aplica) Precisión en la Aplicación de Reglas:** Para mecánicas como el combate, porcentaje de acciones resueltas correctamente según las reglas definidas.

**Método de Medición:**

* Los tiempos de respuesta y tasas de éxito de comandos pueden obtenerse mediante *logging* en el código del bot.
* Los indicadores cualitativos (coherencia, inmersión, usabilidad, relevancia del RAG) se evaluarán mediante la ejecución de escenarios de prueba específicos, autoevaluación utilizando rúbricas predefinidas, y, si es posible, feedback directo de sesiones de prueba con otros usuarios.

## Escalabilidad / Extensión / Mantenimiento / Soporte

### Escalabilidad

* **Bot de Discord:** La librería discord.py está diseñada para manejar la conexión a múltiples servidores y un gran número de usuarios. Sin embargo, la carga real en el bot de este TFG dependerá del procesamiento del LLM y RAG. Para un uso individual o en pequeños grupos, no se anticipan problemas de escalabilidad del bot en sí.
* **LLM (Mistral):** Si se utiliza una API, la escalabilidad la gestiona el proveedor. Si se usa un modelo local, la escalabilidad está limitada por los recursos computacionales (CPU/GPU, RAM) de la máquina anfitriona. Procesar múltiples solicitudes simultáneas a un LLM local grande puede ser un cuello de botella.
* **Sistema RAG:** FAISS es eficiente para la búsqueda en índices vectoriales. La escalabilidad dependerá del tamaño del dataset "Fireball" y la frecuencia de acceso. Para los volúmenes de datos típicos de un TFG, no debería ser un problema.
* **Base de Datos (SQLite):** Adecuada para el alcance del TFG (gestión de hojas de personaje para un número limitado de partidas/usuarios). Para un sistema con alta concurrencia o un gran número de personajes, se requeriría una solución de base de datos más robusta (ej. PostgreSQL, MySQL server).

### Extensión (Líneas de Trabajo Futuro)

El sistema actual sienta las bases para numerosas mejoras y extensiones futuras:

* **Ampliación de Contenido RAG:** Incorporar múltiples módulos de aventura o libros de reglas al sistema RAG, permitiendo al DM IA dirigir diferentes tipos de partidas.
* **Soporte para Múltiples Sistemas de Rol:** Adaptar la lógica y los *prompts* para soportar diferentes conjuntos de reglas de juegos de rol.
* **Mecánicas de Juego Avanzadas:** Implementar sistemas más detallados de magia, habilidades, progresión de personajes, economía, etc.
* **Personalización del DM:** Permitir a los usuarios influir en el estilo narrativo o la "personalidad" del DM IA.
* **Gestión de Estado de Partida Persistente:** Guardar y cargar el estado completo de una partida para continuarla en sesiones posteriores.
* **Integración Visual Mejorada:** Uso de imágenes generadas por IA para escenas o personajes, o integración con herramientas de mapas virtuales si la API de Discord lo permite de forma sencilla.
* **Multijugador Mejorado:** Optimizar la gestión de turnos e interacciones para múltiples jugadores simultáneos de forma más fluida.

### Mantenimiento

* **Código Fuente:** La estructura modular (si se usan Cogs en discord.py y funciones bien definidas) facilitará el mantenimiento. Será necesario mantener actualizadas las librerías principales (discord.py, transformers, etc.) y adaptar el código a posibles cambios en sus APIs.
* **Dataset RAG:** Si el contenido del dataset "Fireball" cambia o se expande, será necesario reprocesarlo y regenerar el índice FAISS.
* **Modelo LLM:** Si se utiliza una API, se depende de la disponibilidad y política de versiones del proveedor. Si es un modelo local, se podrían explorar actualizaciones o afinamientos del modelo.
* **Base de Datos:** SQLite requiere poco mantenimiento para el volumen de datos de este TFG.

### Soporte

En el contexto de un TFG, el "soporte" se refiere principalmente a la documentación que permita a otros (o al propio autor en el futuro) entender, utilizar y potencialmente modificar el sistema.

* **Documentación Principal:** El README.md del repositorio Git será la fuente principal de información para la instalación, configuración y uso básico.
* **Comentarios en el Código:** Código bien comentado es esencial para la comprensión y el mantenimiento.
* **Memoria del TFG:** Este propio documento sirve como una descripción detallada del sistema.

## Plan de Formación de Usuarios (Guía de Usuario)

Dado que el DM IA es una aplicación interactiva, se proporcionará una guía para los usuarios (jugadores).

### Público Objetivo

El público objetivo principal para el DM IA desarrollado en este proyecto se compone de:

* **Jugadores de rol (TTRPGs)**: Individuos con interés o experiencia previa en juegos de rol de mesa, especialmente aquellos familiarizados con Dungeons & Dragons (D&D), dado que el sistema RAG se nutre inicialmente del dataset "Fireball" y el LLM está orientado a este tipo de narrativa.
* **Usuarios de la plataforma Discord**: Personas que utilizan activamente Discord para actividades sociales y de juego, y que, por lo tanto, están familiarizadas con la interfaz de la plataforma, el uso de bots y la interacción mediante comandos de texto.
* **Jugadores con dificultad para encontrar un Dungeon Master humano**: El proyecto surge en parte para abordar la escasez de DMs, por lo que se dirige a grupos o jugadores individuales que deseen una experiencia de rol sin la necesidad de una persona que dirija la partida.
* **Personas interesadas en la IA en juegos**: Jugadores y entusiastas de la tecnología curiosos por explorar nuevas formas de interacción lúdica y el potencial de la IA en el ámbito de los juegos de rol y la narrativa interactiva.
* **Jugadores individuales**: Aunque los juegos de rol suelen ser grupales, este DM IA puede ofrecer una experiencia satisfactoria a jugadores individuales que deseen explorar una aventura a su propio ritmo. El alcance inicial del proyecto se centra en una experiencia textual para un solo jugador.
* **Nuevos jugadores de rol**: Si bien la familiaridad con los juegos de rol es una ventaja, el DM IA podría servir como una introducción accesible al mundo del rol para nuevos jugadores, siempre que se sientan cómodos con la interacción a través de Discord y la naturaleza textual del juego.

### Contenido de la Guía de Usuario

La guía se estructurará para cubrir los siguientes aspectos:

1. **Introducción:** Breve descripción de qué es el DM IA y cómo funciona a grandes rasgos.
2. **Cómo Empezar:**
   * (Si aplica) Cómo invitar el bot a un servidor de Discord.
   * Comandos iniciales para verificar que el bot está activo (ej. !ping o /ping).
3. **Interacción Principal con el DM:**
   * Cómo enviar acciones o preguntas al DM (ej. !dm <tu acción> o /dm <tu acción>).
   * Consejos para formular las entradas de manera clara para obtener mejores respuestas del DM IA.
   * Cómo interpretar las respuestas del DM (incluyendo el uso de *Embeds*).
4. **Gestión del Historial:**
   * Comando para limpiar el historial (ej. !clearhistory o /clearhistory).
5. **Gestión de Hojas de Personaje:**
   * Cómo registrar un nuevo personaje (ej. !register <nombre> <clase> ... o /register ...).
   * Cómo ver la hoja de personaje (ej. !sheet o /sheet).
   * Cómo modificar atributos (ej. !setHP <valor> o /sethp ...).
6. **(Opcional) Mecánicas de Combate:**
   * Si se implementa, explicación de los comandos de combate (!startCombat, !initiative, !nextTurn, etc.) y el flujo básico del combate.
7. **Lista Completa de Comandos:** Un resumen de todos los comandos disponibles, su sintaxis y una breve descripción. Esto podría ser accesible también mediante un comando !help o /help en el bot.
8. **FAQ / Resolución de Problemas Comunes:**
   * "El bot no responde, ¿qué hago?"
   * "¿Cómo puedo reiniciar una aventura?" (si esta funcionalidad existe).

### Formato y Distribución

* **Principalmente en README.md:** El archivo README.md del repositorio Git contendrá la guía de usuario más completa.
* **Comando de Ayuda en el Bot:** Un comando !help (o /help) integrado en el bot que liste los comandos disponibles y su uso básico.
* **(Opcional) Sección en la Memoria:** Un resumen de esta guía podría incluirse como un anexo en la memoria del TFG.

# CONCLUSIONES

El presente capítulo marca la culminación de este TFG , recogiendo las principales conclusiones derivadas del análisis, diseño, implementación y pruebas del sistema DM IA para Discord. Se evaluará el grado de consecución de los objetivos planteados inicialmente, se reflexionará sobre los resultados técnicos y el aprendizaje personal obtenido, y finalmente, se delinearán posibles vías futuras de trabajo que podrían expandir y mejorar la solución desarrollada.

## Objetivos Alcanzados

A lo largo de este proyecto, se establecieron una serie de objetivos para guiar el desarrollo del DM IA. A continuación, se evalúa el grado de consecución de cada uno de ellos, basándose en los objetivos generales y específicos planteados en la sección 1.3 de este documento:

* **Objetivo General:** Desarrollar un sistema de IA funcional, accesible a través de un bot de Discord, que actúe como Asistente de DM para partidas de Dungeons & Dragons (D&D). Este sistema deberá ser capaz de generar narrativas, interpretar acciones de los jugadores, aplicar reglas del juego mediante técnicas de RAG con el LLM Mistral, y gestionar interacciones básicas con personajes no jugadores (PNJs), con el fin de enriquecer y facilitar la experiencia de los juegos de rol.
  + **Logro:** Este objetivo general se ha **alcanzado satisfactoriamente**. Se ha desarrollado un prototipo funcional del DM IA integrado en Discord, capaz de generar narrativas contextualizadas, interpretar las entradas de los jugadores, utilizar el modelo Mistral junto con un sistema RAG basado en el dataset "Fireball" para la contextualización y el acceso a información relevante del juego, y gestionar interacciones básicas con PNJs. El sistema está diseñado para actuar como DM en partidas de Dungeons & Dragons.

**Objetivos Específicos:**

**Objetivo específico 1:** Implementar una interfaz de comunicación a través de un bot de Discord, permitiendo a los jugadores introducir sus decisiones y descripciones de sus acciones mediante comandos de texto.

**Logro:** **Alcanzado**. El sistema se ha implementado como un bot de Discord utilizando la librería discord.py, permitiendo la interacción con los jugadores mediante comandos de texto, tal como se describe en el diseño y la implementación del bot (ver secciones 4.2, 6.2.2, 6.3.2).

**Objetivo específico 2:** Implementar la capacidad en la IA para generar respuestas narrativas dinámicas y coherentes, incluyendo descripciones de escenarios, eventos y diálogos de personajes no jugadores (PNJs), utilizando el modelo de lenguaje Mistral.

**Logro:** **Alcanzado**. El sistema integra el modelo de lenguaje Mistral, el cual es responsable de generar las narrativas, descripciones de escenarios y eventos, así como los diálogos de los PNJs en respuesta a las acciones de los jugadores. El diseño de *prompts* ha sido un elemento clave para guiar esta generación (ver secciones 4.3.1, 6.2.3, 6.3.2).

**Objetivo específico 3:** Aplicar las reglas del juego mediante datasets como "Fireball" como base fundamental del mundo virtual gestionado por la IA, asegurando que las mecánicas del juego y la información contextual sean accesibles y utilizadas por el sistema mediante técnicas RAG.

**Logro:** **Alcanzado en gran medida**. Se ha implementado un sistema de Generación Aumentada por Recuperación (RAG) que utiliza el dataset "Fireball" como fuente de conocimiento. Este sistema permite al LLM acceder a información contextual relevante para enriquecer las narrativas y asegurar la coherencia con el mundo de juego. Si bien la aplicación estricta de reglas complejas es un desafío continuo para los LLMs, el sistema RAG asegura que la información del dataset, que puede incluir elementos relacionados con reglas o mecánicas, sea accesible y utilizada por la IA para contextualizar sus respuestas y acciones (ver secciones 4.4, 6.3.3, 6.3.4).

**Objetivo específico 4:** Establecer un control del flujo de juego que se base en una secuencia de turnos o interacciones lógicas, donde la IA gestione la progresión de la partida en función de las acciones de los jugadores y los eventos narrativos.

**Logro:** **Alcanzado**. El sistema gestiona la progresión de la partida basándose en las acciones de los jugadores y los eventos que la IA genera. Se mantiene un historial de la conversación para dar coherencia al flujo de la partida (ver sección 6.3.4). Aunque una gestión detallada de turnos de combate se consideró opcional, el control del flujo narrativo y la secuencia lógica de interacciones están implementados.

**Objetivo específico 5:** Capacitar a la IA para el manejo básico de Personajes No Jugadores (PNJs), permitiendo que estos interactúen de forma creíble con los jugadores y contribuyan al desarrollo de la narrativa.

**Logro:** **Alcanzado**. El LLM Mistral se encarga de generar los diálogos y las acciones de los PNJs, permitiendo interacciones básicas con los jugadores que contribuyen al desarrollo de la historia. El objetivo especificaba un manejo "básico", lo cual es coherente con las capacidades implementadas y las vías futuras propuestas para un modelado más avanzado de PNJs.

## Conclusiones del Trabajo y Personales

### Conclusiones del Trabajo

Este proyecto ha permitido explorar la aplicación práctica de Grandes Modelos de Lenguaje y técnicas de Recuperación Aumentada por Generación en la creación de un DM IA para juegos de rol en Discord. Las principales conclusiones técnicas y académicas son:

* **Viabilidad de LLMs para Rol:** Se ha demostrado que LLMs como Mistral poseen la capacidad generativa necesaria para asumir el rol de DM, creando narrativas y respondiendo a las acciones de los jugadores de forma plausible y, en muchos casos, creativa.
* **Impacto Positivo del RAG:** La implementación del sistema RAG ha resultado fundamental para anclar las respuestas del LLM a un contexto específico (el dataset "Fireball"), mejorando significativamente la coherencia y relevancia de la narrativa en comparación con un LLM operando sin contexto adicional.
* **Importancia Crítica del Diseño de *Prompts*:** La calidad de la interacción con el DM IA depende enormemente de la ingeniería de los *prompts*. Un diseño cuidadoso, que incorpore el contexto RAG, el historial de conversación y directrices claras para el LLM, es esencial para obtener respuestas satisfactorias. Este proceso es inherentemente iterativo.
* **Desafíos de Integración:** La integración de múltiples componentes (bot de Discord, API del LLM o modelo local, sistema RAG, base de datos) presenta desafíos técnicos que requieren una cuidadosa planificación de la arquitectura y el flujo de datos.
* **Limitaciones Inherentes:** A pesar de los avances, los LLMs aún pueden presentar limitaciones en cuanto a la coherencia a muy largo plazo, la comprensión profunda de reglas de juego complejas si no están explícitamente guiados, y la generación ocasional de "alucinaciones". La evaluación de la "calidad" de una experiencia de rol es, además, inherentemente subjetiva.
* **Contribución del Proyecto:** Este TFG ofrece una implementación funcional y documentada de un DM IA, sirviendo como un ejemplo práctico y una base para futuros desarrollos en el creciente campo de la IA aplicada al entretenimiento interactivo y los juegos de rol.

### Conclusiones Personales

La realización de este TFG ha supuesto una experiencia de aprendizaje sumamente enriquecedora y desafiante a nivel personal y profesional:

* **Adquisición y Consolidación de Conocimientos:** Este proyecto me ha permitido adquirir conocimientos prácticos y profundizar en tecnologías punteras como los LLMs (Mistral), la técnica RAG, el desarrollo de bots con discord.py, la gestión de *embeddings* con sentence-transformers y FAISS, y el manejo de bases de datos como SQLite. Además, ha reforzado mis habilidades en Python y en la gestión de proyectos de software individuales.
* **Desarrollo de Habilidades de Resolución de Problemas:** Me he enfrentado a diversos desafíos técnicos, desde la configuración de APIs y la optimización de *prompts*, hasta la depuración de errores en un sistema con múltiples componentes interdependientes. Superar estos obstáculos ha fortalecido mi capacidad de análisis y de resolución de problemas.
* **Gestión del Tiempo y Autonomía:** La planificación y ejecución de un proyecto de esta envergadura de manera individual ha requerido un alto grado de autonomía, disciplina y gestión eficiente del tiempo, especialmente siguiendo un plan de trabajo intensivo.
* **Satisfacción con el Resultado:** A pesar de los desafíos, considero que el resultado obtenido es satisfactorio, logrando un prototipo funcional que cumple con la mayoría de los objetivos propuestos y que demuestra el potencial de las tecnologías empleadas.
* **Relevancia para mi Futuro Profesional:** Las habilidades y conocimientos adquiridos en áreas de alta demanda como la IA, el PLN y el desarrollo de software son directamente aplicables y muy valiosos para mi futura carrera profesional en el campo de la ingeniería informática.

## Vías Futuras

El sistema DM IA desarrollado en este TFG, si bien funcional, presenta numerosas oportunidades para futuras mejoras, extensiones e investigaciones. Algunas de las líneas de trabajo más prometedoras incluyen:

* **Mejoras en la Inteligencia Artificial y Personalización del DM:**
  + **Afinamiento (Fine-tuning) del LLM:** Entrenar o afinar un modelo como Mistral con corpus específicos de literatura de fantasía, transcripciones de partidas de rol o manuales de juego podría mejorar significativamente su estilo narrativo, conocimiento del dominio y coherencia como DM.
  + **Sistema RAG Avanzado:** Implementar técnicas más sofisticadas de RAG, como el re-ranking de los fragmentos recuperados, la generación de consultas múltiples para el RAG, o la capacidad de manejar y sintetizar información de múltiples documentos recuperados de forma más eficiente.
  + **Memoria a Largo Plazo:** Desarrollar mecanismos más robustos para que el DM IA recuerde eventos y decisiones clave de sesiones de juego anteriores o de arcos narrativos extensos.
  + **Modelado de Personalidades de PNJs:** Permitir que el DM IA module la personalidad y el estilo de habla de los Personajes No Jugadores (PNJs) que interpreta, basándose en perfiles predefinidos.
  + **Exploración de Nuevos Modelos:** Evaluar la integración de otros LLMs que puedan ofrecer ventajas en términos de coste, rendimiento o capacidades específicas para el rol.
* **Expansión de Funcionalidades y Mecánicas de Juego:**
  + **Sistema de Combate Detallado:** Desarrollar un módulo de combate completo, con manejo de estadísticas, tiradas de dados, aplicación de efectos, IA para enemigos, etc.
  + **Reglas de Juego Completas:** Integrar sistemas de reglas de juegos de rol populares (ej. D&D 5e, Pathfinder) de forma más explícita, permitiendo al DM aplicar modificadores, resolver pruebas de habilidad, etc.
  + **Progresión de Personajes:** Implementar un sistema de ganancia de experiencia, subida de niveles y adquisición de nuevas habilidades para los personajes jugadores.
  + **Herramientas de Creación de Contenido:** Desarrollar una interfaz o sistema que permita a los usuarios (o al propio estudiante) añadir fácilmente nuevos datasets de aventura, reglas o lore al sistema RAG.
  + **Soporte para Múltiples Aventuras:** Permitir seleccionar diferentes módulos o campañas para que el DM IA los dirija.
* **Mejoras Técnicas, de Arquitectura y UX:**
  + **Despliegue en Servidor Dedicado:** Migrar el bot a un servidor en la nube para asegurar su disponibilidad 24/7.
  + **Optimización del Rendimiento:** Investigar técnicas para reducir la latencia de las respuestas del LLM, especialmente si se utilizan modelos locales grandes.
  + **Interfaz Gráfica de Usuario (GUI):** Considerar el desarrollo de una interfaz web complementaria para la gestión avanzada de personajes, campañas o para visualizar el estado del juego.
  + **Internacionalización:** Adaptar el sistema para que pueda operar y generar narrativa en diferentes idiomas.
  + **Evaluación de Usuario Formal:** Realizar estudios con un mayor número de jugadores para recoger datos cuantitativos y cualitativos sobre la experiencia de juego, la usabilidad y la calidad del DM IA, utilizando encuestas y métricas estandarizadas.
* **Investigación Adicional:**
  + **Ética en DMs IA:** Analizar y abordar posibles sesgos en el LLM o en los datos de entrenamiento, y estudiar el impacto de la IA en la creatividad y la agencia del jugador.
  + **Generación de Contenido Multimodal:** Explorar la integración con herramientas de IA para generar imágenes de escenas, personajes o mapas que acompañen la narrativa textual.

# BIBLIOGRAFÍA

Formato APA 7ª ed

A continuación, se presenta una lista de las fuentes bibliográficas y recursos clave que han sido consultados y utilizados durante la investigación, diseño y desarrollo del presente TFG. Las referencias incluyen documentación de software, librerías, artículos de investigación relevantes para las tecnologías de IA empleadas y guías de estilo.

*(Recuerda que esta lista es una base. Deberás añadir cualquier otro libro, artículo, página web o recurso específico que hayas utilizado para tu estado del arte, metodología, etc., y verificar/completar los detalles de las siguientes entradas).*

American Psychological Association. (2020). *Publication manual of the American Psychological Association* (7th ed.). <https://doi.org/10.1037/0000165-000>

Chacon, S., & Straub, B. (2014). *Pro Git* (2nd ed.). Apress. *(Alternativamente, si solo te refieres al software y su web principal):* Git SCM. (s.f.). *Git* [Software]. Recuperado el 9 de mayo de 2025, de <https://git-scm.com/>

Discord Inc. (s.f.). *Discord Developer Portal*. Recuperado el 9 de mayo de 2025, de <https://discord.com/developers/docs/intro>

Facebook AI Research (FAIR). (s.f.). *FAISS (Facebook AI Similarity Search)* (Versión X.Y) [Software]. Recuperado el 9 de mayo de 2025, de <https://github.com/facebookresearch/faiss> *(Nota: Cita el artículo si es más relevante para tu contexto: Johnson, J., Douze, M., & Jégou, H. (2019). Billion-scale similarity search with GPUs. IEEE Transactions on Big Data, 7(3), 535-547.* [*https://doi.org/10.1109/TBDATA.2019.2921729*](https://www.google.com/search?q=https://doi.org/10.1109/TBDATA.2019.2921729) *o el preprint de arXiv si lo usaste: Johnson, J., Douze, M., & Jégou, H. (2017). Billion-scale similarity search with GPUs. arXiv.* [*https://arxiv.org/abs/1702.08734*](https://arxiv.org/abs/1702.08734)*)*

Hugging Face. (s.f.-a). *Datasets* (Versión X.Y) [Software]. Recuperado el 9 de mayo de 2025, de <https://github.com/huggingface/datasets>

Hugging Face. (s.f.-b). *Transformers* (Versión X.Y) [Software]. Recuperado el 9 de mayo de 2025, de <https://github.com/huggingface/transformers> *(Nota: También puedes citar el artículo de la librería si es más apropiado: Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T. L., Gugger, S., … Brew, J. (2020). Transformers: State-of-the-Art Natural Language Processing. En Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations (pp. 38-45). Association for Computational Linguistics.* [*https://doi.org/10.18653/v1/2020.emnlp-demos.6*](https://doi.org/1%3C3%3E0.18653/v1/2020.emnlp-demos.6)*)*

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. En *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)* (pp. 9459-9474). Curran Associates, Inc. <https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html>

Mistral AI. (s.f.). *[Nombre del modelo específico de Mistral utilizado, ej., Mistral 7B / Mixtral 8x7B]* [Modelo de lenguaje grande]. Recuperado el 9 de mayo de 2025, de [se quitó una URL no válida] *(Nota: Si te basaste principalmente en un paper específico del modelo, cítalo. Ej.: Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, S., de las Casas, D., ... & Lample, G. (2023). Mistral 7B. arXiv.* [*https://arxiv.org/abs/2310.06825*](https://arxiv.org/abs/2310.06825)*)*

Python Software Foundation. (s.f.-a). *asyncio — Asynchronous I/O*. Documentación de Python X.Y.Z. Recuperado el 9 de mayo de 2025, de <https://docs.python.org/>[Version]/library/asyncio.html *(Reemplaza X.Y.Z y [Version] con la versión de Python que usaste, ej. 3.11)*

Python Software Foundation. (s.f.-b). *Python* (Versión X.Y.Z) [Software]. Recuperado de <https://www.python.org> *(Reemplaza X.Y.Z con la versión de Python que usaste)*

Python Software Foundation. (s.f.-c). *venv — Creation of virtual environments*. Documentación de Python X.Y.Z. Recuperado el 9 de mayo de 2025, de <https://docs.python.org/>[Version]/library/venv.html *(Reemplaza X.Y.Z y [Version] con la versión de Python que usaste)*

Rapptz, D., & ComputerByte. (s.f.). *discord.py* (Versión X.Y) [Software]. Recuperado el 9 de mayo de 2025, de <https://github.com/Rapptz/discord.py> *(Busca la versión específica que instalaste en tu proyecto).*

Reimers, N., & Gurevych, I. (2019). *Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks*. Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), 3982-3992. <https://doi.org/10.18653/v1/D19-1410> *(o el enlace de arXiv:* [*https://arxiv.org/abs/1908.10084*](https://arxiv.org/abs/1908.10084)*)* *(Nota: Esta es la cita del paper. Si te refieres a la librería en sí: UKPLab. (s.f.). sentence-transformers (Versión X.Y) [Software]. Recuperado el 9 de mayo de 2025, de* [*https://github.com/UKPLab/sentence-transformers*](https://github.com/UKPLab/sentence-transformers)*)*

Serrano Nieto, F. (2025). *Dataset "Fireball" para DM IA en Discord* (Versión 1.0) [Conjunto de datos]. Repositorio del Proyecto TFG [o especificar si es una fuente externa]. [Si lo subes a GitHub, el URL del repositorio aquí, de lo contrario, describe su origen en la metodología] *(Nota: Este es un ejemplo si creaste el dataset o si no tiene una fuente publicable formal. Si "Fireball" es un dataset publicado por otros, debes citarlo según su fuente original. Si es un recurso muy específico o interno no publicado, describe su origen y contenido en la sección de Metodología o Diseño del Sistema RAG en lugar de aquí, a menos que tu tutor indique lo contrario).*

SQLite Development Team. (s.f.). *SQLite* (Versión X.Y.Z) [Software]. Recuperado el 9 de mayo de 2025, de <https://www.sqlite.org/index.html> *(Indica la versión que utilizaste).*

The Pytorch Team. (s.f.). *PyTorch* (Version X.Y) [Software]. Recuperado de <https://pytorch.org/> *(Si utilizaste PyTorch. Reemplaza X.Y con la versión).*

The TensorFlow Authors. (s.f.). *TensorFlow* (Version X.Y) [Software]. Recuperado de <https://www.tensorflow.org/> *(Si utilizaste TensorFlow. Reemplaza X.Y con la versión).*

# **ANEXOS**

## Manuales de instalación

### Guía de Implementación del Bot DM en un Servidor de Discord

Introducción

Esta guía está dirigida a los administradores de servidores de Discord. Su objetivo es explicar cómo añadir el bot de DM a tu servidor para que tus miembros puedan empezar a interactuar con él.

Este proceso no requiere conocimientos de programación. Solo necesitas tener los permisos adecuados en tu servidor.

Requisitos Previos

Ser Administrador del Servidor: Debes tener una cuenta de Discord con el rol de "Administrador" o, como mínimo, con el permiso de "Gestionar Servidor" en el servidor al que deseas invitar al bot.

Paso 1: Obtener el Enlace de Invitación del Bot

Para invitar a un bot, necesitas su enlace de invitación único (también conocido como enlace OAuth2).

Este enlace te lo debe proporcionar el desarrollador o la persona que aloja el bot. Tendrá un formato similar a este: https://discord.com/api/oauth2/authorize?client\_id=ID\_DEL\_BOT&permissions=8&scope=bot

Paso 2: Añadir el Bot al Servidor

Haz clic en el enlace de invitación que te han proporcionado.

Se abrirá una ventana en tu navegador de internet, pidiéndote que autorices la entrada del bot.

En esta ventana:

Asegúrate de haber iniciado sesión con la cuenta de Discord correcta.

En el menú desplegable "Añadir al servidor", selecciona el servidor de Discord al que quieres invitar al bot.

Haz clic en "Continuar".

Paso 3: Conceder Permisos

A continuación, Discord te mostrará una lista de los permisos que el bot solicita para funcionar correctamente. Estos pueden incluir:

Enviar mensajes

Leer el historial de mensajes

Insertar enlaces

Adjuntar archivos

Es importante que aceptes todos los permisos solicitados, ya que son necesarios para que el bot ejecute sus comandos (responder a los usuarios, mostrar fichas de personaje, etc.).

Haz clic en el botón "Autorizar".

Es posible que debas completar un CAPTCHA para demostrar que no eres un robot.

Paso 4: Verificación Final

¡Y eso es todo! El bot ha sido añadido a tu servidor.

Verás un mensaje en tu canal de bienvenida (si lo tienes configurado) que indica que el bot se ha unido.

El bot aparecerá en la lista de miembros de tu servidor, normalmente en la columna de la derecha.

Para confirmar que todo funciona, ve a cualquier canal de texto y escribe el comando de ayuda:

!ayuda

El bot debería responder con una lista de todos los comandos disponibles. Si lo hace, la implementación ha sido un éxito.

Solución de Problemas Comunes

El bot aparece como "Desconectado" (Offline): Esto significa que el programa del bot no se está ejecutando en el servidor que lo aloja. Debes contactar a la persona que te proporcionó el enlace de invitación para que lo inicie.

El bot está "En línea" pero no responde a los comandos: Revisa los permisos del canal de texto donde estás escribiendo. Asegúrate de que el rol del bot tenga permiso para "Ver el canal" y "Enviar mensajes" en ese canal específico.

### Manual de Usuario para el Bot DM de Discord

#### Introducción

¡Bienvenido al Manual de Usuario del Asistente de DM para Discord! Este bot ha sido diseñado para enriquecer tus partidas de rol, permitiéndote explorar mundos fantásticos, consultar reglas y gestionar tus personajes directamente desde tu servidor de Discord.

#### Cómo Empezar

Para comenzar a usar el bot, simplemente asegúrate de que un administrador lo haya invitado a tu servidor. Una vez que el bot esté en línea, puedes usar los siguientes comandos en cualquier canal de texto.

#### Comandos Principales

A continuación se describen los comandos esenciales para interactuar con el bot.

##### Comando de Ayuda General

* !help o !ayuda

**Descripción**: Muestra una lista completa de todos los comandos disponibles y una breve explicación de lo que hace cada uno. Es tu punto de partida si olvidas cómo usar una función.

#### Interactuar con el Dungeon Master

* !dm [tu acción o pregunta]
  + **Descripción**: Este es el comando principal para jugar. Describe lo que tu personaje hace o dice, y el DM IA te responderá continuando la narración.
  + **Ejemplo**: !dm Exploro la cueva con cuidado, buscando trampas.
* !limpiar\_historial\_dm o !resetdm
  + **Descripción**: Borra el historial de la conversación actual de la partida en el canal. Útil para empezar una nueva escena o si la conversación se ha desviado.
* !historia
  + **Descripción**: Muestra un resumen de los últimos eventos importantes que han ocurrido en la partida en ese canal.

#### Gestión de Personajes

* !crear\_personaje [nombre]
  + **Descripción**: Inicia el proceso interactivo para crear un nuevo personaje. El bot te guiará con una serie de preguntas.
  + **Ejemplo**: !crear\_personaje Kaelen el Valiente
* !ficha o !sheet
  + **Descripción**: Muestra la hoja de tu personaje activo, con sus atributos, nivel, puntos de golpe, etc.
* !mis\_personajes o !listpj
  + **Descripción**: Lista todos los personajes que has creado asociados a tu cuenta de Discord.
* !activar\_personaje [nombre]
  + **Descripción**: Establece cuál de tus personajes será el activo para los comandos !ficha y !dm.
  + **Ejemplo**: !activar\_personaje Lyra
* !set\_hp [valor]
  + **Descripción**: Actualiza los Puntos de Golpe (HP) actuales de tu personaje activo.
  + **Ejemplo**: !set\_hp 15
* !escribir\_diario [texto]
  + **Descripción**: Añade o sobrescribe una entrada en el diario de tu personaje, útil para llevar un registro de la aventura desde su perspectiva.
* !leer\_diario
  + **Descripción**: Muestra el contenido actual del diario de tu personaje.

#### Consulta de Reglas y Hechizos

* !regla [término]
  + **Descripción**: Consulta la definición o explicación de una regla específica de D&D.
  + **Ejemplo**: !regla Cobertura
* !hechizo [nombre del hechizo]
  + **Descripción**: Obtiene información detallada sobre un hechizo, incluyendo su nivel, descripción, componentes y efectos.
  + **Ejemplo**: !hechizo Bola de Fuego

#### Otros Comandos Útiles

* !tirar [expresión de dados] o !roll [expresión]
  + **Descripción**: Lanza los dados según la expresión indicada.
  + **Ejemplo**: !tirar 2d6+3
* !iniciar\_combate [enemigo1] [enemigo2] ...
  + **Descripción**: Inicia un encuentro de combate, calculando la iniciativa para los jugadores presentes y los enemigos listados.
* !turno y !siguiente
  + **Descripción**: Muestra de quién es el turno actual en combate y pasa al siguiente en el orden de iniciativa, respectivamente.

#### Preguntas Frecuentes (FAQ)

* **¿Qué hago si el bot no responde?**
  + Asegúrate de que estás usando el prefijo correcto (!). Si el bot sigue sin responder, es posible que esté desconectado o en mantenimiento. Contacta con el administrador del servidor.
* **¿Cómo puedo obtener las mejores respuestas del DM?**
  + Sé claro y descriptivo en tus acciones. En lugar de decir !dm ataco, prueba con !dm Blando mi espada y lanzo un tajo al goblin que tengo enfrente.