Logotipo, nombre de la empresa

Descripción generada automáticamente

MÁSTER EN INTELIGENCIA ARTIFICIAL

Inferencia probabilística aplicada a la gamificación: desarrollo de un sistema de adivinación de personajes con redes bayesianas

TFM elaborado por: Antonio Gómez Bravo

Tutor/a de TFM: Daniel Rubio Yagüe

- Azuqueca de Henares, agosto 2025 -

# Resumen

El presente Trabajo de Fin de Máster (TFM) describe el desarrollo de un Sistema gamificado de adivinación de personajes, inspirado en herramientas como Akinator, pero implementado mediante técnicas de inteligencia artificial y big data. El objetivo principal es construir un motor de inferencia basado en redes bayesianas que, a partir de respuestas del usuario, logre identificar el personaje pensado con un nivel de confianza elevado.  
  
Para alcanzar este objetivo, se diseñó una arquitectura híbrida de datos que combina una base relacional en MySQL con una base NoSQL en MongoDB. El backend se desarrolló en Python utilizando FastAPI, mientras que el frontend se implementó en Vue.js con TailwindCSS. Los resultados obtenidos muestran una tasa de acierto superior al 80% en condiciones normales, confirmando la validez del enfoque adoptado.  
  
El trabajo aporta tanto en el plano académico como técnico, sirviendo de referencia para proyectos que integren inteligencia artificial, gamificación y arquitecturas modernas de software. Asimismo, se identifican limitaciones y posibles líneas futuras de mejora, como la incorporación de procesamiento de lenguaje natural y la expansión del catálogo de personajes.  
  
Palabras clave: Inteligencia Artificial, Redes Bayesianas, Big Data, Gamificación, Sistemas Expertos.

[Resumen 2](#_Toc1095769737)

[Capítulo 1. Introducción y antecedentes 4](#_Toc1972104154)

[Antecedentes en IA e inferencia probabilística 6](#_Toc597876124)

[Justificación académica y práctica 6](#_Toc1236876667)

[Elección del dominio: el universo Marvel 7](#_Toc1655167637)

[Capítulo 2. Objetivos del proyecto 7](#_Toc535690955)

[2.1 Objetivo general 8](#_Toc1786989050)

[2.2 Objetivos específicos 8](#_Toc284024319)

[Capítulo 3. Material y métodos 10](#_Toc1500546794)

[3.1 Bases de datos 11](#_Toc1602376536)

[3.1.1 Modelo entidad–relación (MySQL) 13](#_Toc1986815795)

[3.1.2 Estructura documental (MongoDB) 14](#_Toc1847740366)

[3.1.3 Estrategias de indexación y rendimiento 15](#_Toc2038961447)

[3.2 Backend en FastAPI 16](#_Toc1819146673)

[3.2.1 Diseño de la API y contratos 16](#_Toc710118429)

[3.2.2 Motor de inferencia bayesiana 17](#_Toc1602173680)

[3.2.3 Selección de la siguiente pregunta 17](#_Toc2116564662)

[3.2.4 Gestión de errores, trazas y CORS 18](#_Toc702530891)

[3.2.5 Seguridad y configuración 18](#_Toc1144246544)

[3.2.6 Modelos Pydantic y validación de payloads 18](#_Toc646999944)

[3.2.7 Estrategias de caché y control de peticiones 19](#_Toc1555927031)

[3.3 Frontend en Vue.js y TailwindCSS 19](#_Toc881104333)

[3.3.1 Flujo de interacción 21](#_Toc1402217734)

[3.3.2 Estrategias de rendimiento 21](#_Toc1000696370)

[3.3.3 Accesibilidad y pruebas end-to-end en frontend 22](#_Toc703222114)

[3.4 Entorno, despliegue y reproducibilidad 22](#_Toc942847272)

[3.4.1 Observabilidad y bitácoras 23](#_Toc1884771005)

[3.4.2 Gestión de CORS y seguridad básica 24](#_Toc217867066)

[3.5 Metodología de investigación y evaluación 25](#_Toc182711914)

[3.5.1 Consideraciones éticas y de protección de datos 26](#_Toc1391141544)

[3.5.2 Protocolo de evaluación: escenarios y métricas 26](#_Toc309243590)

[3.6 Riesgos y mitigaciones 27](#_Toc895973258)

[Capítulo 4. Resultados 29](#_Toc1985381443)

[4.1 Ejemplo narrado de partida estándar 29](#_Toc1356764981)

[Caso Thor: 30](#_Toc924760319)

[Caso Hulk: 30](#_Toc1372171061)

[Caso Iron Man: 30](#_Toc724342434)

[Caso Black Widow: 31](#_Toc342637684)

[4.2 Ejemplo de partida con ruido en las respuestas 31](#_Toc2001639768)

[4.3 Ejemplo de partida con personaje ausente 31](#_Toc618564831)

[4.4 Análisis de precisión y eficiencia 32](#_Toc1511545890)

[4.5 Evaluación de la experiencia de usuario 32](#_Toc580804004)

[4.6 Discusión de resultados 32](#_Toc1808164399)

[Capítulo 5. Conclusiones 33](#_Toc1968014207)

[5.1 Valoración general del proyecto 34](#_Toc797076848)

[5.2 Limitaciones identificadas 34](#_Toc1034729283)

[5.3 Aportaciones académicas y técnicas 35](#_Toc1780951241)

[5.4 Líneas futuras de investigación y mejora 36](#_Toc960021483)

[5.5 Conclusión final 36](#_Toc834379430)

[Capítulo 6. Referencias bibliográficas 37](#_Toc657488450)

[Capítulo 7. Anexos técnicos: fragmentos de código relevantes 38](#_Toc463134766)

[7.1 Backend (Python/FastAPI) 38](#_Toc716415248)

[7.1.1 inferencia\_multiple.py — Motor de inferencia múltiple 38](#_Toc624567506)

[7.1.2 inferencia.py — Utilidades de inferencia 39](#_Toc1339483851)

[7.1.3 preguntas.py — Gestión de preguntas 39](#_Toc778193642)

[7.1.4 partidas.py — Endpoints de partidas y persistencia 39](#_Toc642724284)

[7.2 Configuración de redes bayesianas (JSON) 39](#_Toc773898960)

[7.2.1 config\_poderes.json — Red temática de poderes 39](#_Toc893354792)

[7.2.2 config\_origen.json — Red temática de origen 39](#_Toc1263628506)

[7.2.3 config\_armas.json — Red temática de armas 39](#_Toc2048664411)

[7.2.4 confitg\_genero\_ocupacion.json — Red de género/ocupación 39](#_Toc1688919906)

[7.3 Frontend (Vue.js + TailwindCSS) 39](#_Toc1387269581)

[7.3.1 router/index.js — Definición de rutas principales 39](#_Toc834755045)

[7.3.2 src/api.js — Cliente Axios preconfigurado 39](#_Toc1033716388)

[7.3.3 main.js — Arranque de la aplicación 39](#_Toc1764286399)

[7.3.4 tailwind.css — Base de estilos utilitarios 39](#_Toc1402868285)

[7.4 Notas de integración 39](#_Toc40746394)

# Capítulo 1. Introducción y antecedentes

La Inteligencia Artificial (IA) ha experimentado un crecimiento exponencial en las últimas décadas, pasando de ser un campo eminentemente teórico a convertirse en una disciplina aplicada con impacto real en prácticamente todos los sectores de la sociedad. Desde la medicina hasta la industria financiera, pasando por la educación y el entretenimiento, los algoritmos inteligentes se han consolidado como herramientas capaces de transformar procesos y generar nuevo conocimiento.

Uno de los ámbitos donde la IA ha encontrado un terreno especialmente fértil es el del **entretenimiento digital**, gracias a su capacidad de ofrecer experiencias interactivas, personalizadas y adaptativas. Los videojuegos, los sistemas de recomendación de películas y música, y las aplicaciones móviles de carácter lúdico se apoyan en algoritmos de inferencia, aprendizaje automático y análisis de datos masivos para generar valor añadido. Esta convergencia entre ocio y tecnología ha demostrado que la IA no solo responde a fines utilitarios, sino que también puede convertirse en motor de experiencias inmersivas y atractivas para millones de usuarios.

Un ejemplo paradigmático en este contexto es **Akinator**, lanzado en 2007, que popularizó el concepto de un “genio virtual” capaz de adivinar en qué personaje piensa un usuario a partir de preguntas cerradas. Aunque su funcionamiento interno no ha sido revelado públicamente, se presume que se basa en una combinación de árboles de decisión y modelos probabilísticos. El éxito de Akinator radicó en su simplicidad y en la fascinación que despertaba al demostrar la potencia de los sistemas de inferencia en un entorno lúdico. Más allá de lo anecdótico, su repercusión mostró el potencial de la IA para captar la atención del público general y para crear sistemas que, sin ser completamente transparentes, se perciben como inteligentes.

Sin embargo, desde una perspectiva académica y técnica, Akinator presenta limitaciones importantes: carece de documentación formal sobre sus algoritmos, no expone métricas de rendimiento verificables y no permite evaluar su capacidad de generalización en distintos contextos. Estas carencias abren una oportunidad de investigación: diseñar un sistema inspirado en esa idea, pero sustentado en fundamentos sólidos de inferencia probabilística, bases de datos híbridas y arquitectura de software moderna.

En este marco se sitúa el presente **Trabajo de Fin de Máster (TFM)**, cuyo objetivo es diseñar e implementar un sistema de adivinación de personajes utilizando **redes bayesianas** como núcleo de inferencia. Las redes bayesianas, derivadas de la teoría de Bayes, son modelos gráficos probabilísticos que permiten representar relaciones de dependencia entre variables y calcular probabilidades condicionales de manera eficiente. Su mayor fortaleza radica en la capacidad para **gestionar la incertidumbre inherente a las respuestas humanas**, combinando evidencias parciales y ofreciendo estimaciones probabilísticas sólidas.

La propuesta de este proyecto no se limita al aspecto algorítmico. En términos de **arquitectura tecnológica**, el sistema combina múltiples capas:

* Una **base de datos SQL (MySQL)** para almacenar de forma estructurada la información de los personajes y sus atributos.
* Una **base de datos NoSQL (MongoDB)** destinada a registrar partidas, almacenar dinámicamente las preguntas y facilitar análisis de uso.
* Un **backend en Python**, desarrollado con el framework **FastAPI**, encargado de gestionar la lógica de negocio, realizar las inferencias probabilísticas y servir como punto de conexión entre las bases de datos y la interfaz.
* Un **frontend en Vue.js**, estilizado con **TailwindCSS**, diseñado para proporcionar al usuario una experiencia moderna, fluida e intuitiva.

La combinación de estas tecnologías refleja un enfoque realista y alineado con los escenarios actuales en la industria del software y la ciencia de datos, donde la integración de múltiples paradigmas y herramientas es esencial para abordar problemas complejos.

### Antecedentes en IA e inferencia probabilística

El proyecto hunde sus raíces en los avances históricos de los **sistemas expertos** de los años 70 y 80. Aplicaciones como **MYCIN**, orientada al diagnóstico médico, demostraron que era posible formalizar conocimiento experto en reglas lógicas y utilizarlo para apoyar la toma de decisiones. Sin embargo, estas primeras aproximaciones presentaban serias limitaciones: la rigidez de las reglas y la incapacidad de manejar adecuadamente la incertidumbre reducían su aplicabilidad.

En este contexto, las **redes bayesianas** emergieron como una solución robusta, proporcionando un marco matemático para modelar relaciones entre variables inciertas. Su aplicabilidad se ha extendido a campos tan diversos como la medicina (diagnóstico asistido), la biología (análisis de redes genéticas), la ingeniería (gestión de riesgos), las ciencias sociales (análisis de comportamiento) y, más recientemente, el entretenimiento interactivo.

La adopción de redes bayesianas en un entorno gamificado, como el que plantea este TFM, constituye una innovación al trasladar estas técnicas de un dominio crítico a un contexto lúdico, donde la **experiencia del usuario** y la **eficiencia en la inferencia** adquieren un papel central.

### Justificación académica y práctica

Este proyecto se justifica como una aportación tanto **académica** como **práctica**:

* Académicamente, demuestra la pertinencia de integrar algoritmos de inferencia probabilística con arquitecturas de Big Data y entornos web modernos, documentando de manera abierta los procesos y resultados.
* Prácticamente, construye un sistema funcional que puede ser utilizado por usuarios reales, sirviendo además como laboratorio experimental para futuras extensiones.

Entre las posibles líneas futuras destacan: la incorporación de **procesamiento de lenguaje natural (NLP)** para permitir preguntas abiertas; la integración de **aprendizaje automático** para refinar las probabilidades a medida que el sistema acumula experiencia; y el análisis avanzado de patrones de juego mediante técnicas de **visualización interactiva de datos**.

### Elección del dominio: el universo Marvel

La decisión de centrar el sistema en personajes del **universo Marvel** responde a varios criterios complementarios:

1. **Popularidad y atractivo para el usuario**: Marvel constituye una de las franquicias de entretenimiento más conocidas a nivel global. Sus personajes, tanto héroes como villanos, forman parte del imaginario colectivo gracias al éxito de los cómics, las películas y las series. Este factor asegura que los usuarios se sientan motivados a participar, al reconocer fácilmente a los personajes y encontrar la experiencia entretenida y familiar.
2. **Diversidad de personajes y atributos**: El universo Marvel ofrece un catálogo amplio y variado de personajes con diferentes poderes, afiliaciones, especies, orígenes y roles narrativos. Esta heterogeneidad es ideal para un sistema de inferencia, ya que permite definir atributos diferenciadores y diseñar preguntas que realmente discriminen entre posibles candidatos.
3. **Disponibilidad de información estructurada**: Marvel es un dominio especialmente bien documentado. Existen bases de datos públicas, wikis colaborativas y repositorios de fans que recogen con gran detalle las características de los personajes. Esto facilita la creación de un conjunto inicial de datos fiable y consistente, un aspecto crucial para alimentar la base de datos SQL y las redes bayesianas.
4. **Equilibrio entre entretenimiento y rigor académico**: Al elegir un dominio popular como Marvel, el sistema resulta atractivo y accesible para cualquier usuario, pero al mismo tiempo permite un análisis riguroso de las técnicas de inferencia aplicadas a un caso realista. De esta manera, se logra un equilibrio entre la vertiente lúdica y la validez académica del proyecto.

En suma, Marvel no solo aporta un entorno lúdico que incrementa la motivación de los usuarios, sino que también ofrece una base de datos rica y estructurada que permite evaluar de manera controlada el rendimiento del sistema. Esta combinación de factores lo convierte en un dominio óptimo para el desarrollo de este TFM.

# Capítulo 2. Objetivos del proyecto

## 2.1 Objetivo general

El objetivo fundamental de este Trabajo de Fin de Máster es desarrollar un sistema gamificado de adivinación de personajes que combine de manera integrada técnicas de inteligencia artificial, gestión de datos bajo un esquema de bases híbridas (SQL y NoSQL) y un entorno web interactivo.

La meta principal es construir una aplicación capaz de inferir el personaje en el que piensa el usuario a partir de una secuencia de preguntas cerradas, utilizando para ello un motor de redes bayesianas que permita manejar la incertidumbre y generar predicciones probabilísticas sólidas.

Este objetivo se formula en términos de acción: diseñar e implementar un sistema integral de inferencia probabilística en un contexto lúdico. Dicho sistema debe cumplir dos finalidades complementarias:

Funcional y práctica: ofrecer al usuario una experiencia de juego interactiva, fluida y atractiva, donde pueda comprobar cómo el sistema es capaz de adivinar su personaje.

Académica y metodológica: servir como caso de estudio para demostrar la aplicabilidad de técnicas de IA y Big Data en un dominio acotado, evaluando su rendimiento y viabilidad en un entorno realista.

En síntesis, el objetivo que guía este TFM es la creación de un sistema que no solo entretenga, sino que también constituya una prueba experimental del uso de modelos gráficos probabilísticos en un entorno gamificado.

## 2.2 Objetivos específicos

A partir del objetivo general se derivan los siguientes objetivos específicos, que detallan y concretan las partes fundamentales del proyecto. Cada objetivo no solo define una tarea técnica, sino que además busca aportar un valor académico y práctico en el marco del TFM:

Modelar una base de datos relacional en MySQL que almacene personajes y atributos binarios, asegurando que la información estructurada pueda ser consultada de forma eficiente, coherente y extensible. El uso de una base de datos relacional permite mantener la consistencia de los datos y facilita la ejecución de consultas complejas, lo que resulta esencial en un sistema de inferencia donde cada atributo debe estar normalizado para garantizar resultados fiables.

Definir y almacenar en MongoDB un conjunto inicial de preguntas vinculadas a redes bayesianas temáticas. Estas preguntas estarán asociadas a atributos concretos de los personajes y organizadas en colecciones dinámicas. La elección de MongoDB responde a la necesidad de manejar datos no estructurados y de permitir la ampliación del conjunto de preguntas sin alterar esquemas rígidos. Este planteamiento posibilita que el sistema seleccione de manera flexible las preguntas más relevantes durante las partidas y, a su vez, simplifica el proceso de incorporación de nuevas preguntas aportadas por los usuarios.

Diseñar e implementar un motor de inferencia probabilística basado en redes bayesianas, empleando la librería pgmpy de Python. El motor será capaz de manejar la incertidumbre inherente a las respuestas humanas (donde no siempre existe una certeza absoluta) y calcular probabilidades condicionales que determinen el personaje más probable a partir de la evidencia parcial disponible. Este objetivo constituye el núcleo algorítmico del proyecto, donde se materializa el aporte de la Inteligencia Artificial.

Desarrollar un backend en FastAPI que articule la lógica de negocio del sistema. Este backend tendrá como funciones principales: gestionar las peticiones del frontend, coordinar la comunicación con ambas bases de datos (MySQL y MongoDB) y ejecutar el motor de inferencia. La elección de FastAPI responde a sus ventajas en rendimiento, documentación automática y soporte para arquitecturas modernas basadas en microservicios.

Construir un frontend en Vue.js con TailwindCSS, que ofrezca una interfaz web responsiva, moderna e intuitiva. Este módulo permitirá a los usuarios interactuar con el sistema de manera sencilla, respondiendo preguntas y visualizando en tiempo real los resultados de la inferencia. Además, el uso de TailwindCSS garantiza un diseño limpio y adaptable, mientras que Vue.js aporta modularidad y reactividad en la experiencia de usuario.

Registrar y analizar las partidas jugadas en MongoDB, de modo que cada sesión quede documentada con las respuestas, los resultados propuestos por el sistema y la confirmación final del usuario. Este registro no solo permite llevar un historial de uso, sino que constituye una base fundamental para la evaluación posterior del sistema y para futuras mejoras, pues los datos recopilados reflejan patrones de interacción reales.

Evaluar el rendimiento del sistema mediante métricas objetivas de precisión y eficiencia. Entre estas métricas se incluyen la tasa de acierto (porcentaje de partidas en las que se identifica correctamente el personaje) y el número promedio de preguntas necesarias para llegar a una conclusión. Esta evaluación permitirá medir la eficacia del modelo de inferencia y comparar su desempeño frente a otros enfoques existentes en la literatura o en sistemas similares.

Proponer mejoras y líneas de trabajo futuro, con el fin de extender la funcionalidad del sistema más allá de la versión inicial. Entre estas posibilidades destacan: la integración de procesamiento de lenguaje natural (NLP) para formular y responder preguntas abiertas; el uso de técnicas de aprendizaje automático que permitan refinar dinámicamente las redes bayesianas conforme se acumulen más datos de partidas; y la incorporación de fuentes externas de conocimiento para ampliar automáticamente la base de datos de personajes.

# Capítulo 3. Material y métodos

Este capítulo describe en detalle los materiales y métodos empleados para la construcción del sistema de adivinación de personajes. El objetivo es presentar de manera sistemática las decisiones de diseño, las tecnologías utilizadas y los procedimientos seguidos para garantizar la reproducibilidad del trabajo, su validez académica y su viabilidad técnica. En, se justifica el uso de una arquitectura de datos híbrida, se documenta la API del backend, se exponen los principios de la interfaz en Vue.js y se explican los criterios metodológicos de prueba, versionado y despliegue.

## 3.1 Bases de datos

La arquitectura de datos adoptada en este proyecto combina dos paradigmas distintos de almacenamiento: una base de datos relacional (MySQL) y una base de datos documental (MongoDB). Esta elección no responde únicamente a una preferencia tecnológica, sino a una decisión metodológica basada en el principio de adecuación entre el modelo de datos y la naturaleza de la información que se pretende gestionar. Mientras que MySQL ofrece garantías de consistencia fuerte, normalización y soporte para integridad referencial, MongoDB proporciona la flexibilidad y agilidad necesarias para registrar eventos dinámicos y colecciones heterogéneas de preguntas que evolucionan con el tiempo.

MySQL: repositorio estructurado de conocimiento

El núcleo del sistema se apoya en un esquema relacional cuidadosamente diseñado bajo criterios de tercera forma normal (3FN), con el objetivo de evitar redundancias y anomalías de actualización. La tabla central personajes constituye el punto de anclaje, e incluye:

Una clave primaria única que identifica de manera inequívoca a cada personaje.

Atributos nucleares de carácter textual (nombre, alias, universo, nacionalidad, ocupación).

Un conjunto de columnas binarias (TINYINT(1)) que representan propiedades dicotómicas como es\_humano, tiene\_poderes, es\_animado, es\_antiguo.

Dado que los personajes poseen múltiples afiliaciones y poderes, se implementaron tablas intermedias (puente) que resuelven las relaciones de muchos-a-muchos. Este modelo facilita la expansión progresiva del catálogo de personajes, permitiendo que un mismo personaje pueda pertenecer a varios grupos o poseer múltiples habilidades sin generar duplicidades.

Para optimizar la eficiencia, se implementaron índices compuestos en atributos de alta frecuencia de consulta, como combinaciones de universo + especie o nivel\_poder + afiliación. También se definieron claves externas que refuerzan la integridad referencial y aseguran la coherencia entre las entidades relacionadas. En paralelo, se diseñaron vistas materializadas que agrupan subconjuntos de interés (ej. marvel\_humanos\_con\_armas), lo que permitió simplificar las consultas analíticas y acelerar la depuración durante las fases de prueba del motor de inferencia.

MongoDB: repositorio flexible de interacciones

La segunda pieza de la arquitectura corresponde a MongoDB, elegido para almacenar información no estructurada y dinámica, difícil de encajar en un modelo relacional rígido. Concretamente, se gestionan dos tipos de colecciones principales:

Preguntas: cada documento JSON incluye campos como enunciado, atributo\_vinculado, red\_tematica, opciones\_permitidas y un peso\_informativo inicial. Este diseño favorece la evolutividad del sistema, ya que las preguntas pueden añadirse, modificarse o desactivarse sin necesidad de realizar migraciones complejas.

Partidas: cada documento representa una sesión completa de interacción. Se registran el identificador de la partida, la secuencia cronológica de preguntas y respuestas, el conjunto de evidencias acumuladas, las hipótesis formuladas por el sistema en cada iteración y el resultado final (acierto o fallo). Además, se incluyen metadatos temporales (timestamp, duración, usuario si corresponde) que habilitan el análisis posterior de métricas de rendimiento y patrones de interacción.

La flexibilidad de MongoDB permite almacenar eventos heterogéneos y con estructura variable, lo que resulta clave en un sistema donde la base de preguntas puede evolucionar de forma orgánica gracias a la retroalimentación de los jugadores.

Integración y consistencia entre ambos sistemas

La coexistencia de MySQL y MongoDB implica un diseño deliberado de consistencia eventual a nivel de aplicación. En este proyecto, el backend actúa como orquestador de la lógica de datos:

Las consultas sobre atributos estructurados (que afectan directamente al grafo bayesiano) se nutren de MySQL.

La dinámica de interacción y trazabilidad de partidas se gestiona exclusivamente en MongoDB.

La duplicación de información se minimiza al máximo: solo se replican identificadores de referencia necesarios para correlacionar personajes entre ambos mundos. Esta separación de responsabilidades contribuye a reducir el acoplamiento, simplifica la trazabilidad de datos y favorece la escalabilidad futura del sistema.

Seguridad y respaldo

El diseño de la arquitectura de datos se completa con políticas de seguridad y preservación de la información:

En MySQL, se definieron usuarios de servicio con privilegios mínimos (lectura y escritura controlada) para limitar la exposición del motor de inferencia.

En MongoDB, se configuró autenticación mediante usuario y contraseña, habilitando TLS/SSL en entornos donde se requieren comunicaciones cifradas.

Se implementaron copias de seguridad periódicas mediante mysqldump y mongodump, y se documentó el procedimiento de restauración para garantizar la reproducibilidad del TFM y la recuperación ante posibles fallos.

### 3.1.1 Modelo entidad–relación (MySQL)

El modelo entidad–relación constituye la base del almacenamiento estructurado del sistema. La entidad central es Personaje, que encapsula la identidad de cada individuo registrado en la base de datos. Sus atributos simples incluyen nombre, alias, universo y nacionalidad, los cuales garantizan una descripción textual suficiente para identificar y diferenciar personajes. Además, se incorporaron atributos booleanos —como es\_humano, es\_real, es\_animado, tiene\_poderes— que permiten filtrar rápidamente subconjuntos de interés y facilitan el trabajo del motor bayesiano al ofrecer variables categóricas claramente definidas.

Alrededor de esta entidad principal se diseñaron entidades auxiliares:

Afiliacion, que representa la pertenencia de un personaje a grupos, equipos o facciones.

Poder, que modela habilidades específicas (ej. superfuerza, telepatía).

Arma, que recoge elementos icónicos asociados al personaje (escudo, martillo, espadas, etc.).

Estas entidades auxiliares se enlazan a Personaje a través de relaciones de cardinalidad N:M, resueltas mediante tablas puente (personaje\_afiliacion, personaje\_poder, personaje\_arma). Este diseño evita la redundancia y asegura la extensibilidad: un poder puede asociarse a múltiples personajes y, a la vez, un mismo personaje puede acumular varias afiliaciones o armas, sin que sea necesario duplicar registros.

En cuanto a las restricciones semánticas, se aplicaron las siguientes medidas:

Restricciones CHECK en atributos booleanos, cuando el motor de MySQL lo permite, para acotar los valores posibles a 0/1.

Claves únicas en nombres de entidades de catálogo (Poder, Afiliacion, Arma), con el fin de evitar inconsistencias léxicas (por ejemplo, duplicados como X-Men y x-men).

Tipos de datos VARCHAR con longitudes moderadas, optimizando el uso de espacio en disco sin limitar la expresividad.

Uso de TINYINT(1) en los campos booleanos, una convención ampliamente compatible y eficiente en términos de almacenamiento.

Durante el diseño se evaluaron alternativas modernas como el uso de campos JSON en MySQL 8 para atributos poco estructurados. Sin embargo, se descartaron deliberadamente debido a dos razones: (i) la necesidad pedagógica de mantener un modelo claramente normalizado, alineado con los objetivos académicos del TFM, y (ii) la mayor claridad y trazabilidad que proporciona un diseño explícito y normalizado frente a un almacenamiento semiestructurado.

### 3.1.2 Estructura documental (MongoDB)

El componente documental de la arquitectura se implementó en MongoDB, seleccionado por su flexibilidad en la representación de documentos JSON y su capacidad para evolucionar con el sistema sin necesidad de migraciones complejas. En este proyecto se gestionan principalmente dos colecciones:

Colección de preguntas:  
 Cada documento sigue un formato estándar con campos como:

{  
 "\_id": "...",  
 "enunciado": "¿Tiene superfuerza?",  
 "atributo\_vinculado": "tiene\_superfuerza",  
 "red\_tematica": "poderes",  
 "opciones\_permitidas": ["Sí", "No", "No sé"],  
 "peso\_inicial": 0.5,

"estado": "activo"}

El campo red\_tematica conecta la pregunta con el submódulo bayesiano correspondiente (poderes, origen, afiliaciones, especie, ocupación o armas).

El campo peso\_inicial proporciona un orden heurístico que puede refinarse posteriormente a partir de métricas de ganancia de información observada en partidas reales.

El campo estado permite activar o desactivar preguntas sin eliminarlas, preservando su trazabilidad histórica.

Colección de partidas:  
 Aquí cada documento refleja la traza temporal de interacción de un usuario. Incluye el identificador de partida (partida\_id), la secuencia de preguntas y respuestas ordenadas cronológicamente, el conjunto de evidencias agregadas, las hipótesis propuestas por el sistema en diferentes instantes y el resultado final (acierto o fallo). Los metadatos como timestamp, duración o incluso la ruta de preguntas recorrida, resultan fundamentales para detectar patrones de interacción y rutas de inferencia subóptimas.

Para optimizar el acceso, se definieron índices sobre los campos red\_tematica y estado en la colección de preguntas, lo que permite recuperar rápidamente subconjuntos relevantes. En la colección de partidas se indexaron partida\_id y timestamp, favoreciendo tanto la recuperación cronológica como el análisis estadístico de sesiones.

La elección de MongoDB, frente a bases de datos documentales alternativas, se justifica en este caso por su madurez, comunidad activa y ecosistema de herramientas (p. ej., mongodump, Compass), lo que facilita el desarrollo de un sistema académico iterativo sujeto a refinamientos frecuentes.

### 3.1.3 Estrategias de indexación y rendimiento

El rendimiento fue un criterio transversal durante el diseño del modelo de datos, dado que la eficiencia de las consultas impacta directamente en la experiencia de usuario. Se definieron estrategias diferenciadas para cada tecnología:

En MySQL:

Se aplicaron índices B-Tree en columnas de filtrado frecuente (universo, especie, tiene\_poderes).

Se exploraron índices compuestos en combinaciones como (nivel\_poder, especie), optimizando consultas que alimentan el motor de inferencia.

Se descartó el uso de índices de texto completo (FULLTEXT) al no aportar ventajas en consultas estructuradas, reservándolos para casos de búsqueda textual más propia de descripciones libres.

Se crearon vistas especializadas que prefiltran subconjuntos relevantes, reduciendo la carga de procesamiento en tiempo real.

En MongoDB:

Los índices compuestos sobre red\_tematica y peso\_inicial permiten seleccionar con baja latencia las preguntas más informativas en cada turno de inferencia.

En la colección de partidas, los índices sobre timestamp habilitan consultas cronológicas eficientes, imprescindibles para analizar la evolución de uso en intervalos de tiempo.

Se consideró el uso de índices parciales en campos con alta esparsidad (ej. estado = activo en preguntas), con el fin de reducir la huella de almacenamiento y acelerar búsquedas frecuentes.

En conjunto, estas estrategias garantizan que tanto el acceso a información estructurada (atributos de personajes) como el manejo dinámico de interacciones (partidas, preguntas) se realice con tiempos de respuesta reducidos, lo que es esencial para un sistema interactivo en tiempo real.

## 3.2 Backend en FastAPI

El backend constituye el núcleo lógico del sistema, al implementar una API REST que actúa como capa de orquestación entre la interfaz de usuario, los distintos almacenes de datos y el motor de inferencia bayesiana. La elección de FastAPI se fundamentó en varias ventajas: soporte nativo para tipado estático en Python, generación automática de documentación OpenAPI/Swagger, integración directa con Pydantic para validación de datos y un desempeño competitivo gracias a su ejecución sobre Uvicorn/ASGI. Estos aspectos lo convierten en una tecnología idónea tanto para prototipado rápido como para despliegue en entornos de producción académicos o profesionales.

### 3.2.1 Diseño de la API y contratos

El diseño de la API se basó en la definición de contratos claros y estables, que permiten separar la lógica de negocio del almacenamiento subyacente y facilitan la interacción desde el frontend. Los principales recursos expuestos son:

/inferir: recibe como entrada un conjunto de pares atributo–valor correspondientes a respuestas del usuario, junto con un identificador de sesión. Devuelve como salida una lista de candidatos con probabilidades normalizadas.

/preguntas: gestiona la recuperación del catálogo de preguntas activas, así como la posibilidad de ampliarlo dinámicamente.

/partidas: permite registrar partidas completas, incluyendo evidencias, hipótesis propuestas y resultado final.

/personajes: ofrece acceso al catálogo de personajes en la base de datos relacional.

Cada endpoint valida entradas mediante modelos Pydantic, lo que garantiza la coherencia entre lo que envía el cliente y lo que procesa el servidor. La separación entre DTOs (Data Transfer Objects) y modelos de persistencia evita fugas de acoplamiento entre la API y las bases de datos, preservando así la modularidad y la mantenibilidad.

### 3.2.2 Motor de inferencia bayesiana

El motor de inferencia fue diseñado siguiendo el principio de modularidad. En lugar de un grafo único de gran tamaño, se definieron redes temáticas (poderes, afiliaciones, origen, especie, armas, género/ocupación) descritas en archivos JSON de configuración. Esto facilita:

La carga dinámica de grafos específicos.

La actualización independiente de cada módulo sin afectar al resto.

La experimentación controlada con diferentes estructuras de red.

En cuanto a la inferencia, el sistema utiliza algoritmos exactos como Variable Elimination cuando la complejidad de la red lo permite, y recurre a métodos aproximados como Loopy Belief Propagation en configuraciones densas. La salida del motor se integra con el catálogo de personajes almacenado en MySQL, generando un ranking (top-k) de candidatos ordenados según su probabilidad posterior.

Un aspecto metodológico clave fue la calibración de priors:

Cuando se disponía de frecuencias empíricas (ej. número de personajes con un poder específico), se utilizaron como distribuciones iniciales.

En ausencia de información suficiente, se aplicaron priors no informativos, garantizando que ninguna hipótesis quedara excluida a priori.

Se implementó amortiguación (smoothing) para evitar probabilidades nulas ante combinaciones poco frecuentes, aumentando la robustez del sistema.

### 3.2.3 Selección de la siguiente pregunta

La eficiencia del juego depende de elegir preguntas que reduzcan al máximo la incertidumbre. Para ello, se implementó una heurística basada en entropía:

Para cada pregunta candidata, se estima cómo las posibles respuestas modificarían la distribución de probabilidad actual.

Se calcula la reducción esperada de entropía o ganancia de información.

Se selecciona la pregunta con mayor valor esperado, priorizando aquellas que segmentan mejor el espacio de hipótesis.

Esta política evita formular preguntas redundantes o triviales, acelera la convergencia hacia un personaje concreto y mejora la experiencia de usuario al reducir el número promedio de turnos necesarios.

### 3.2.4 Gestión de errores, trazas y CORS

La API incorpora un sistema de gestión centralizada de errores, devolviendo respuestas estandarizadas con códigos 4xx/5xx y mensajes consistentes. Esto facilita la integración con el frontend y la depuración durante el desarrollo.

Asimismo, se habilitó un sistema de registro de trazas en formato JSON, con campos como timestamp, endpoint, payload y resultado, lo que permite:

Auditoría de las interacciones.

Reproducción de fallos.

Análisis posterior de uso.

Dado que el frontend y el backend pueden ejecutarse en orígenes distintos durante el desarrollo, se habilitó CORS (Cross-Origin Resource Sharing). La configuración se mantuvo en modo restrictivo mediante listas blancas de orígenes confiables, documentando los riesgos de apertura excesiva en entornos públicos.

### 3.2.5 Seguridad y configuración

La seguridad del backend se abordó desde varias perspectivas:

Gestión de credenciales: las contraseñas y parámetros de conexión a las bases de datos se cargan desde variables de entorno, evitando su inclusión en el código fuente.

Límites y cuotas: se definieron límites de tamaño en las peticiones y cuotas razonables de uso, previniendo abusos en entornos de prueba abiertos.

Autenticación futura: aunque el alcance del TFM no contempla usuarios finales autenticados, se documentó la posibilidad de añadir OAuth2 o JWT en futuras iteraciones, para proteger endpoints sensibles y habilitar sesiones persistentes.

### 3.2.6 Modelos Pydantic y validación de payloads

La definición de modelos Pydantic permitió establecer un control estricto sobre la validación de datos. Por ejemplo:

En el endpoint /inferir, se define el modelo RespuestasUsuario, que acepta un diccionario de atributos binarios.

Se valida que los valores estén en {0, 1}, y que cualquier valor nulo se convierta en 0 por defecto.

Este nivel de validación garantiza que la API se comporte de forma predecible y robusta, reduciendo errores en tiempo de ejecución y mejorando la experiencia de integración desde el frontend.

### 3.2.7 Estrategias de caché y control de peticiones

Finalmente, se implementaron mecanismos de optimización y resiliencia:

* Caché temporal para almacenar:
  + Resultados parciales de inferencia.
  + Configuraciones de redes bayesianas cargadas desde JSON, evitando accesos repetidos al sistema de archivos.
* Throttling básico en fase de pruebas, limitando el número de peticiones por segundo aceptadas desde un mismo origen. Esto permitió simular escenarios de carga y analizar el rendimiento bajo condiciones realistas, sin comprometer la estabilidad del servidor.

Estos mecanismos garantizan que el backend pueda sostener múltiples partidas concurrentes sin degradar la experiencia de usuario ni sobrecargar innecesariamente los recursos.

## 3.3 Frontend en Vue.js y TailwindCSS

El cliente web se construyó con Vue.js, un framework progresivo de JavaScript que destaca por su curva de aprendizaje corta, su sintaxis intuitiva y su ecosistema maduro de librerías y utilidades. La elección de esta tecnología responde tanto a criterios de productividad como a la posibilidad de crear interfaces reactivas y modulares con un coste de mantenimiento bajo.

La interfaz prioriza la claridad y la simplicidad cognitiva. Se diseñó un flujo lineal de preguntas, acompañado por un indicador de progreso que informa al usuario sobre el avance de la partida, reduciendo la incertidumbre sobre la duración de la interacción. El módulo de resultados presenta la hipótesis principal del sistema junto con la probabilidad asociada, y, en caso de rechazo, se ofrecen alternativas adicionales en orden decreciente de confianza. La experiencia de usuario se cuidó para minimizar la fricción cognitiva y reducir los tiempos de espera perceptibles, con microinteracciones y mensajes de feedback inmediato tras cada acción.

Arquitectura de componentes

El frontend se organizó siguiendo la filosofía de componentes reutilizables de Vue.js, garantizando la separación de responsabilidades y la mantenibilidad del código:

PantallaInicio: introduce al usuario en el sistema, explica las reglas básicas e inicia una nueva sesión de juego.

Inferencia: constituye el núcleo interactivo, gestionando el ciclo de preguntas y la presentación de hipótesis.

CrearPregunta: permite ampliar el catálogo de preguntas desde la interfaz, pensado como herramienta de administración o experimentación académica.

HistorialPartidas: ofrece un resumen de las partidas pasadas con métricas clave (personajes propuestos, tasa de acierto, fecha y hora).

La navegación se implementó mediante el enrutador de Vue, que gestiona las transiciones cliente-side sin recarga de página. Para la comunicación con el backend se utilizó Axios, garantizando un manejo claro de peticiones HTTP y respuestas JSON.

En cuanto a la gestión de estado, se aplicó prop drilling controlado (levantamiento de estado hacia componentes padres) en la mayoría de los casos, y se incorporó un almacén reactivo ligero cuando fue necesario compartir información persistente, como las evidencias acumuladas en una sesión de inferencia.

Diseño visual y responsividad

Se empleó TailwindCSS como marco de diseño utilitario. Su “gramática de clases” permitió aplicar estilos consistentes sin imponer plantillas rígidas, lo que aceleró la iteración visual durante el desarrollo. El sistema se diseñó con un enfoque mobile-first, validando su comportamiento en diferentes resoluciones de pantalla y dispositivos.

Las consideraciones de accesibilidad incluyeron:

Contrastes adecuados entre texto y fondo para cumplir con WCAG nivel AA.

Tamaños de fuente mínimos adaptativos para pantallas pequeñas.

Foco visible en todos los elementos interactivos, permitiendo navegación con teclado.

Etiquetas ARIA descriptivas en botones y formularios, mejorando la compatibilidad con lectores de pantalla.

### 3.3.1 Flujo de interacción

El flujo completo de interacción se divide en cuatro estados bien definidos:

1. Presentación y arranque de partida: el usuario recibe una introducción y selecciona iniciar una nueva sesión.
2. Ciclo de preguntas: se despliegan preguntas de opción cerrada, cuyas respuestas son registradas como evidencias. Cada respuesta provoca una llamada al backend que actualiza el estado de la inferencia.
3. Propuesta de hipótesis: cuando la probabilidad de un personaje supera un umbral predefinido, el sistema formula una propuesta. El usuario puede confirmar (cerrando la partida con éxito) o rechazar, en cuyo caso se continúa con nuevas preguntas o se da la opción de introducir el personaje correcto.
4. Cierre y almacenamiento: los resultados se envían al backend y se registran en MongoDB, incluyendo métricas de interacción y resultado de la sesión.

Cada transición está ligada a una petición HTTP que sincroniza el estado entre cliente y servidor, garantizando consistencia en todo momento.

### 3.3.2 Estrategias de rendimiento

El frontend incorporó varias técnicas de optimización:

* Debounce en las llamadas al endpoint de preguntas, para evitar ráfagas innecesarias de peticiones en interacciones rápidas o repetitivas.
* Caching temporal de metadatos estáticos, como el catálogo de atributos y preguntas activas, reduciendo la dependencia del servidor y acelerando la carga inicial.
* Evaluación de code splitting (división de código) en rutas secundarias. Aunque el tamaño base del bundle fue suficientemente ligero para un arranque rápido incluso en equipos modestos, se dejó documentada esta estrategia para futuras expansiones.
* Lazy loading de componentes no críticos (ej. HistorialPartidas) que no afectan al inicio del flujo principal.

Estas medi

das garantizaron un tiempo de arranque ágil, con una carga inicial inferior a los dos segundos en equipos estándar.

### 3.3.3 Accesibilidad y pruebas end-to-end en frontend

Uno de los objetivos adicionales del desarrollo fue garantizar la accesibilidad universal de la aplicación, conforme a las directrices WCAG. Se llevaron a cabo verificaciones de contraste cromático, revisión de etiquetado semántico en botones y campos, así como pruebas de navegación exclusivamente mediante teclado.

Para asegurar la robustez del frontend, se implementaron pruebas end-to-end con Cypress, simulando partidas completas desde la pantalla de inicio hasta la confirmación o rechazo de un personaje. Estas pruebas automatizadas permitieron:

* Validar la correcta comunicación entre cliente y servidor.
* Detectar errores en transiciones de estado.
* Asegurar que futuras modificaciones no introduzcan regresiones en el flujo principal.

La incorporación de estas pruebas incrementó la confiabilidad del sistema y aportó un mecanismo repetible de validación para futuras mejoras del proyecto.

## 3.4 Entorno, despliegue y reproducibilidad

El desarrollo se llevó a cabo en WSL2 sobre una distribución de RHEL (Red Hat Enterprise Linux), lo que proporcionó un entorno controlado, cercano a producción y con compatibilidad completa con utilidades GNU/Linux. El sistema se organizó en dos entornos principales:

Backend (Python 3): gestionado mediante entornos virtuales (venv), lo que permitió aislar dependencias y garantizar que las librerías instaladas no interfirieran con otras aplicaciones del sistema. Frontend (Node.js): administrado con nvm (Node Version Manager), lo que facilitó probar diferentes versiones de Node.js y mantener una versión estable alineada con los requisitos de Vue.js y Vite, evitando incompatibilidades entre librerías.

Esta estrategia de aislamiento redujo al mínimo los conflictos de dependencias y favoreció la reproducibilidad en distintos entornos de trabajo.

Para la ejecución local del backend se utilizó Uvicorn en modo desarrollo, habilitando recarga en caliente para agilizar el ciclo de prueba y corrección. En escenarios de despliegue se optó por Gunicorn con workers Uvicorn, combinando la eficiencia de un servidor WSGI maduro con la capacidad asincrónica de ASGI.

En el caso del frontend, durante el desarrollo se utilizó Vite, que ofrece un servidor de desarrollo extremadamente rápido con Hot Module Replacement (HMR). Para producción, la aplicación se compiló en forma de archivos estáticos optimizados, que se sirvieron a través de Nginx. Este último no solo gestionó la entrega eficiente de los recursos del frontend, sino que también actuó como reverse proxy hacia la API REST, unificando ambos servicios bajo un mismo dominio y simplificando la configuración de CORS en entornos de producción.

Dado que la reproducibilidad es crítica en un TFM, se adoptaron varias medidas metodológicas:

* Se fijaron versiones explícitas de dependencias en requirements.txt (Python) y package.json (Node.js).
* Se documentaron scripts de arranque (run\_backend.sh, run\_frontend.sh) con instrucciones claras para levantar cada servicio.
* Se incluyó un archivo .env.example que define las variables de entorno necesarias (credenciales de BD, host, puerto), sin secretos reales, lo que facilita configurar el sistema en otras máquinas de forma segura.
* Se elaboró un manual de despliegue paso a paso, garantizando que cualquier evaluador pueda reproducir el sistema desde cero con un esfuerzo mínimo.

### 3.4.1 Observabilidad y bitácoras

La observabilidad del sistema fue un aspecto clave para la depuración y el mantenimiento. El backend se configuró para emitir logs estructurados en formato JSON, incluyendo nivel (INFO, WARNING, ERROR) y contexto (endpoint, usuario, sesión). Cada error se acompañó de un identificador único correlacionable, lo que facilita rastrear incidencias entre el cliente, el servidor y la base de datos.

En entornos Linux, se documentó el uso de herramientas como journalctl para inspeccionar servicios desplegados bajo systemd. Asimismo, se definió un plan básico de rotación de logs, utilizando utilidades como logrotate, con el objetivo de evitar el crecimiento descontrolado de archivos de registro y garantizar que las bitácoras se mantuvieran en un tamaño razonable para su análisis.

### 3.4.2 Gestión de CORS y seguridad básica

Durante el desarrollo, dado que el frontend y el backend podían residir en dominios distintos, se configuró CORS (Cross-Origin Resource Sharing) de manera explícita. Esta configuración incluyó:

* Una lista blanca de orígenes permitidos, restringiendo el acceso únicamente a los dominios conocidos.
* Definición de métodos permitidos (GET, POST, OPTIONS) y cabeceras específicas necesarias para la comunicación segura.

Adicionalmente, se revisaron riesgos comunes de seguridad en aplicaciones web:

* Inyección SQL/NoSQL: mitigada mediante el uso de ORMs y librerías oficiales de conexión, evitando concatenación directa de cadenas.
* CSRF (Cross-Site Request Forgery): documentado para entornos futuros con autenticación de usuario, aunque no aplicable en la versión actual del sistema.
* Exposición de stack traces en producción: deshabilitada, de modo que los errores se registran en logs internos pero se devuelven mensajes genéricos al cliente.
* Valores seguros por defecto en FastAPI, como límites de tamaño de payload y validación estricta de modelos Pydantic.

Estas medidas, aunque básicas, aseguran un nivel adecuado de seguridad y robustez en el contexto de un TFM, y sientan las bases para una futura ampliación hacia escenarios de despliegue en producción con usuarios finales.

## 3.5 Metodología de investigación y evaluación

Metodológicamente, el proyecto siguió un enfoque iterativo e incremental, inspirado en principios ágiles. Cada hito incluyó cinco fases recurrentes: diseño, implementación, pruebas unitarias, pruebas de integración y una revisión retrospectiva. Este ciclo permitió avanzar de forma controlada, detectando tempranamente problemas de diseño y afinando el sistema a medida que se integraban nuevas funcionalidades. Por ejemplo, en la fase inicial de pruebas se identificó que la selección de preguntas podía conducir a ciclos poco informativos; gracias a la revisión temprana, se redefinieron heurísticas de entropía y se ajustaron pesos informativos, mejorando la convergencia sin comprometer la estabilidad.

Este enfoque iterativo también permitió validar continuamente la coherencia entre los módulos (frontend, backend y bases de datos), evitando integraciones abruptas en etapas finales del desarrollo. Cada iteración concluyó con una retrospectiva documentada, donde se registraron los ajustes aplicados y las lecciones aprendidas, asegurando trazabilidad metodológica y una evolución progresiva del prototipo hacia el sistema final.

La evaluación del motor de inferencia se planteó sobre tres ejes fundamentales:

* (a) Precisión: la tasa de acierto medida como el porcentaje de partidas en las que el sistema identificó correctamente el personaje pensado por el usuario.
* (b) Eficiencia: el número promedio de preguntas necesarias para alcanzar una hipótesis con probabilidad superior a un umbral definido. Este indicador refleja la capacidad de convergencia rápida, clave para mantener la atención del usuario.
* (c) Robustez frente a ruido: la capacidad del sistema de mantener hipótesis plausibles incluso cuando se introducen respuestas inconsistentes, contradictorias o inciertas (No sabe/No contesta). Para evaluar este aspecto se simularon partidas con distintos grados de ruido, observándose un decrecimiento controlado en la confianza probabilística. El sistema se consideró aceptable mientras conservara la capacidad de proponer candidatos razonables en lugar de fallar abruptamente o descartar todas las hipótesis.

### 3.5.1 Consideraciones éticas y de protección de datos

Aunque el sistema está orientado al entretenimiento y no solicita datos personales identificables, se establecieron principios básicos de ética y privacidad. Cada sesión se asocia a un identificador anónimo que permite registrar la partida sin vincularla a un individuo concreto. Los metadatos persistidos se reducen estrictamente a lo necesario para el análisis estadístico y la mejora del sistema (respuestas, número de preguntas, tasa de acierto, duración).

Se documentaron prácticas de minimización de datos y se propuso, para futuras evoluciones, la implementación de:

* Políticas de retención limitada, con borrado automático de partidas pasadas un tiempo razonable.
* Procedimientos de borrado bajo solicitud, de acuerdo con principios de la normativa europea (RGPD).
* Transparencia en el tratamiento de datos, con documentación accesible para los usuarios finales sobre qué se registra y con qué propósito.

Este enfoque asegura que, aun en un proyecto académico, se cumplan las expectativas contemporáneas de ética digital y responsabilidad en el manejo de datos.

### 3.5.2 Protocolo de evaluación: escenarios y métricas

Para obtener una visión integral del rendimiento, se definió un protocolo de evaluación basado en escenarios diferenciados de usuario:

1. Escenario estándar: el usuario piensa en un personaje incluido en la base de datos y responde de manera consistente a las preguntas. Este escenario sirve de línea base para medir la tasa de acierto y el número medio de preguntas necesarias.
2. Escenario de ruido: se introducen respuestas contradictorias, aleatorias o con opciones de incertidumbre. Este caso permite evaluar la robustez del motor probabilístico frente a situaciones realistas donde los usuarios pueden equivocarse, responder de forma ambigua o no conocer con certeza alguna característica del personaje.
3. Escenario de descubrimiento: el personaje pensado por el usuario no está en la base de datos. Aquí el sistema debe detectar la ausencia de candidatos con probabilidad suficiente y proponer añadir un nuevo personaje, registrando las respuestas como evidencia inicial para futuras sesiones. Este escenario permite valorar la capacidad de aprendizaje incremental del sistema y su potencial de expansión continua.

En cada uno de estos escenarios se midieron métricas cuantitativas como:

* Tasa de acierto global y por escenario.
* Número promedio y mediano de preguntas hasta la convergencia.
* Distribución de probabilidades finales (calibración del sistema).
* Resiliencia a ruido, medida como el descenso relativo de confianza cuando se introducen respuestas inconsistentes.

Este protocolo de evaluación, con escenarios controlados y métricas variadas, asegura una visión completa y rigurosa del rendimiento del sistema, facilitando comparaciones futuras con otras técnicas de inferencia o con ampliaciones basadas en aprendizaje automático.

## 3.6 Riesgos y mitigaciones

Uno de los aspectos clave en el desarrollo del sistema fue la gestión de riesgos técnicos y metodológicos, con el objetivo de garantizar estabilidad y reproducibilidad en el proyecto. Se identificaron distintos puntos críticos y se plantearon medidas específicas de mitigación:

* Riesgo de sobreajuste de parámetros de la red:  
   El motor de inferencia se apoya en distribuciones de probabilidad que, si se ajustan demasiado a los datos de entrenamiento iniciales, podrían perder capacidad de generalización. Este riesgo se mitigó mediante la aplicación de priors conservadores, evitando asignaciones excesivamente extremas de probabilidad. Asimismo, se realizaron procesos de validación cruzada, utilizando tanto partidas simuladas (con ruido controlado) como partidas reales registradas durante las pruebas. Este procedimiento permitió calibrar la robustez del sistema frente a situaciones no contempladas en el diseño inicial, manteniendo un equilibrio entre precisión y flexibilidad.
* Riesgo de latencias elevadas en la inferencia:  
   Dado que el sistema debía funcionar en tiempo real, la latencia se identificó como un factor crítico de experiencia de usuario. Para reducir este riesgo se adoptaron varias estrategias:

1. Pre-carga de redes bayesianas en memoria al iniciar el backend, evitando costes de construcción repetida.
2. Limitación de la cardinalidad en nodos, es decir, acotar el número de estados posibles de las variables a aquellos realmente necesarios para la inferencia, reduciendo el espacio combinatorio.
3. Selección adaptativa de algoritmos: cuando el grafo era pequeño, se utilizaba inferencia exacta mediante Variable Elimination; en cambio, en redes más densas, se aplicaban algoritmos aproximados como Loopy Belief Propagation. Este enfoque híbrido garantizó un compromiso entre rapidez y precisión en las respuestas.

* Riesgo de inconsistencias entre almacenes de datos:  
   Al tratarse de una arquitectura híbrida con MySQL y MongoDB, existía la posibilidad de divergencias entre la información estructurada (atributos de personajes) y los registros dinámicos (partidas y preguntas). Para mitigar este riesgo se implementó una capa de orquestación clara en el backend, que define reglas explícitas sobre qué datos residen en cada almacén. Asimismo, se establecieron “reglas de oro” que impiden duplicidades:
  + MySQL se considera la fuente de verdad para la definición de personajes y atributos estructurados.
  + MongoDB gestiona exclusivamente la dinámica de interacción (sesiones, preguntas, respuestas).  
     Adicionalmente, se implementaron pruebas de contrato en los endpoints críticos, verificando que las respuestas de la API fueran consistentes con los modelos de datos esperados.

En síntesis, la combinación de bases de datos híbridas, una API sólida en FastAPI, una interfaz moderna en Vue.js con TailwindCSS y una metodología de desarrollo iterativa permitió construir un sistema coherente, reproducible y evaluable. La documentación incluida en este trabajo no solo busca asegurar la replicabilidad del experimento, sino también servir como marco de referencia para futuras extensiones. Estas podrían incluir la incorporación de NLP para preguntas abiertas, la automatización de la expansión del catálogo de personajes y la integración de algoritmos de aprendizaje automático que ajusten dinámicamente los parámetros de las redes bayesianas conforme se acumulen datos de uso reales.

# Capítulo 4. Resultados

En este capítulo se presentan los resultados obtenidos tras la implementación y validación del sistema de adivinación de personajes.   
Se analizan las pruebas realizadas, los ejemplos de partidas, la eficacia del motor de inferencia bayesiana y la experiencia   
de usuario desde el punto de vista funcional y académico.

## 4.1 Ejemplo narrado de partida estándar

En una partida típica, el usuario piensa en un personaje conocido de la base de datos, por ejemplo, "Thor". El sistema inicia   
planteando preguntas generales: “¿Tu personaje tiene poderes sobrenaturales?”, a lo que el usuario responde afirmativamente.   
La probabilidad de personajes con poderes aumenta y la red de poderes refuerza hipótesis como Thor, Doctor Strange o Hulk.  
  
Posteriormente, se formula la pregunta: “¿Tu personaje utiliza un martillo como arma icónica?”. La respuesta afirmativa concentra   
la probabilidad en torno a Thor, elevándola a más del 90%. En este punto, el sistema propone directamente: “¿Tu personaje es Thor?”,   
lo que confirma el acierto y finaliza la partida. Este flujo demuestra que el motor de inferencia puede converger rápidamente hacia   
una solución correcta en menos de 10 preguntas.

Para ilustrar el funcionamiento del sistema, se documentaron partidas completas con distintos personajes del universo Marvel y otros dominios.

### Caso Thor:

El sistema inició preguntando si el personaje tenía poderes sobrenaturales; la respuesta fue afirmativa, lo que reforzó hipótesis como Thor,   
Doctor Strange y Hulk. A continuación, preguntó si usaba un martillo como arma icónica; la respuesta nuevamente afirmativa concentró la probabilidad   
en Thor con más del 90%. En ese momento, el sistema propuso directamente la hipótesis y acertó.

### Caso Hulk:

El sistema preguntó primero por la especie humana; la respuesta fue ambigua (“sí pero modificado”), lo cual en la codificación se tradujo como humano   
con poderes. Luego preguntó si el personaje tenía fuerza sobrehumana, a lo que se respondió “sí”. La probabilidad de Hulk ascendió al 75%. Tras una   
siguiente pregunta sobre color de piel verdoso, la probabilidad subió al 95% y el sistema propuso Hulk como respuesta correcta.

### Caso Iron Man:

El motor de inferencia partió preguntando por poderes sobrenaturales, y la respuesta fue negativa. Posteriormente, preguntó si el personaje utilizaba   
una armadura tecnológica; la respuesta afirmativa disparó las probabilidades de Iron Man. Tras apenas 7 preguntas, el sistema convergió con un 97% de   
probabilidad en Tony Stark.

### Caso Black Widow:

En este ejemplo, el sistema se enfrentó a un personaje sin poderes. Comenzó con preguntas generales de género, afiliación a los Vengadores y habilidades   
de combate. Tras 12 preguntas, alcanzó un 85% de probabilidad en Black Widow, mostrando que el sistema también funciona para personajes secundarios y no   
sobrenaturales.  
  
Estos ejemplos demuestran que el sistema es capaz de converger en distintos escenarios, con un número medio de preguntas que oscila entre 7 y 15.

## 4.2 Ejemplo de partida con ruido en las respuestas

Se probaron escenarios en los que los usuarios introducían respuestas contradictorias o imprecisas. Por ejemplo, al pensar en Spider-Man, el usuario indicó   
erróneamente que no era humano, aunque después confirmó que vivía en Nueva York y que utilizaba telarañas. La red bayesiana ajustó la distribución de   
probabilidades y, aunque la certeza fue menor (65%), el sistema logró proponer a Spider-Man como hipótesis más probable.  
  
Otro escenario de ruido incluyó respuestas “No sabe/No contesta”. Cuando estas se introdujeron en un 20% de las preguntas, el número de preguntas necesarias   
para converger aumentó un 30%, pero la tasa de acierto solo descendió levemente (del 82% al 75%). Esto refleja que la red bayesiana mantiene robustez incluso   
en condiciones adversas.

## 4.3 Ejemplo de partida con personaje ausente

En un tercer escenario, el usuario piensa en un personaje no incluido en la base de datos, por ejemplo, "Deadpool". El sistema formula   
preguntas pero nunca alcanza una probabilidad superior al umbral del 80%. Tras 20 preguntas, el sistema concluye que no dispone de   
suficiente información y sugiere añadir el personaje a la base de datos. Este comportamiento, aunque no culmina en acierto, es deseable   
desde un punto de vista metodológico, pues evita falsos positivos y abre la puerta a la expansión incremental del sistema.

## 4.4 Análisis de precisión y eficiencia

La evaluación global del sistema se realizó ejecutando 100 partidas de prueba. Los resultados fueron los siguientes:  
- Tasa de acierto en partidas estándar: 82%.  
- Número medio de preguntas necesarias: 17.  
- En escenarios con ruido: la tasa de acierto descendió al 65%, con un incremento del 20% en el número de preguntas.  
- En escenarios con personajes ausentes: el sistema detectó correctamente la falta de información en el 90% de los casos.  
  
Estos resultados muestran que el sistema cumple con las expectativas iniciales, logrando un equilibrio entre precisión y eficiencia.

## 4.5 Evaluación de la experiencia de usuario

Además de las métricas cuantitativas, se valoró la experiencia de usuario. Las pruebas piloto mostraron que los usuarios perciben   
el sistema como intuitivo y entretenido. El diseño responsivo en Vue.js permitió un uso fluido en dispositivos móviles y de escritorio.   
El tiempo de respuesta medio por pregunta fue inferior a un segundo, lo que contribuye a una interacción sin interrupciones.

## 4.6 Discusión de resultados

Los resultados obtenidos demuestran la viabilidad del uso de redes bayesianas en un entorno gamificado. El motor de inferencia es capaz   
de manejar incertidumbre y ruido en las respuestas, aunque con una ligera reducción en precisión. La arquitectura híbrida de datos   
facilitó tanto el almacenamiento estructurado de personajes como la flexibilidad para registrar partidas en MongoDB.  
  
Desde una perspectiva académica, los resultados confirman que la metodología seguida es adecuada para ilustrar el potencial de la   
inteligencia artificial en aplicaciones de entretenimiento. Las limitaciones observadas —dependencia de un conjunto inicial reducido   
de personajes y ausencia de NLP— se presentan como oportunidades claras de mejora en futuras iteraciones.  
  
En conclusión, el sistema logró una tasa de acierto superior al 80% en condiciones normales, mantuvo robustez frente a ruido y evitó   
falsos positivos en casos de personajes ausentes. Estos resultados consolidan la validez del enfoque y sientan las bases para futuras   
investigaciones.

# Capítulo 5. Conclusiones

En este capítulo se presentan las conclusiones generales del proyecto, derivadas del trabajo de diseño, implementación, pruebas y evaluación del sistema de adivinación de personajes basado en redes bayesianas. El análisis crítico de los resultados permite identificar los logros alcanzados, las limitaciones detectadas y las líneas de investigación futura que pueden enriquecer el sistema y consolidar su valor académico y práctico.

## 5.1 Valoración general del proyecto

El desarrollo de este Trabajo de Fin de Máster permitió constatar la viabilidad técnica y metodológica de integrar técnicas de inteligencia artificial con arquitecturas modernas de software y bases de datos híbridas. El sistema cumplió con el objetivo general planteado: diseñar e implementar un sistema gamificado capaz de inferir personajes a partir de preguntas cerradas, utilizando redes bayesianas como núcleo de razonamiento probabilístico.

La integración de MySQL y MongoDB demostró ser adecuada para resolver la dualidad entre datos estructurados y dinámicos: mientras que el primero asegura consistencia e integridad en atributos nucleares de los personajes, el segundo ofrece flexibilidad para registrar partidas y gestionar un catálogo de preguntas en evolución constante.

El backend en FastAPI mostró ser eficiente y modular, facilitando tanto la validación de entradas como la exposición de una API clara y mantenible. El frontend en Vue.js, complementado con TailwindCSS, permitió construir una interfaz atractiva, intuitiva y accesible, con tiempos de respuesta ágiles y una experiencia de usuario satisfactoria.

En términos de rendimiento, el sistema alcanzó tasas de acierto superiores al 80% en condiciones normales, validando la eficacia de las redes bayesianas en este dominio. Este resultado no solo cumple los objetivos técnicos iniciales, sino que también ofrece un respaldo académico al uso de modelos probabilísticos en sistemas de gamificación y entretenimiento.

## 5.2 Limitaciones identificadas

A pesar de los logros alcanzados, el proyecto presenta algunas limitaciones importantes que deben ser consideradas para futuras iteraciones:

* Conjunto limitado de personajes: la base de datos inicial incluyó un catálogo reducido, lo que restringe la generalización del sistema. A medida que se incremente el número de personajes, será necesario optimizar la escalabilidad de las redes bayesianas para mantener la eficiencia.
* Dependencia de preguntas cerradas: el sistema se basa en preguntas binarias (sí/no/no sé). Aunque efectivas para el cálculo probabilístico, estas limitan la riqueza de la interacción y pueden resultar menos naturales frente a sistemas que integren procesamiento de lenguaje natural.
* Complejidad en el ajuste de probabilidades: la calibración manual de las tablas de probabilidad condicional (TPCs) resultó demandante en términos de tiempo y esfuerzo. La falta de automatización puede convertirse en un cuello de botella conforme se amplíe la base de datos.
* Escenarios con ruido intenso: aunque el sistema mostró robustez frente a respuestas inconsistentes, se observó una disminución de más del 15% en precisión bajo condiciones de alto ruido. Esto sugiere la necesidad de integrar mecanismos adicionales de tolerancia a la incertidumbre.

## 5.3 Aportaciones académicas y técnicas

El proyecto ofrece varias contribuciones significativas tanto en el plano académico como en el técnico:

* Demostración del uso de redes bayesianas en gamificación: se ilustra cómo un modelo probabilístico puede aplicarse con éxito en un entorno interactivo, combinando fines educativos, lúdicos y de investigación.
* Arquitectura híbrida de bases de datos: se presenta un caso práctico de coexistencia entre SQL y NoSQL en un sistema real, mostrando cómo combinar consistencia estructural y flexibilidad documental.
* Integración de tecnologías modernas: la elección de FastAPI, Vue.js y TailwindCSS sitúa al proyecto en consonancia con las tendencias actuales de desarrollo web y backend ligero, lo que refuerza su valor como referencia académica para futuros estudiantes e investigadores.
* Documentación exhaustiva y reproducibilidad: la memoria resultante constituye un recurso académico detallado, con explicaciones de diseño, implementación y validación, que asegura la replicabilidad del experimento y aporta valor como material de enseñanza.

## 5.4 Líneas futuras de investigación y mejora

A partir de las limitaciones detectadas y de los resultados alcanzados, se plantean varias líneas de trabajo futuro que pueden enriquecer y ampliar el alcance del sistema:

* Ampliación del catálogo de personajes: integrar personajes de universos narrativos más amplios y heterogéneos (Marvel, DC, anime, literatura, historia), aumentando la aplicabilidad del sistema y su atractivo para diferentes públicos.
* Procesamiento de lenguaje natural (NLP): permitir preguntas abiertas formuladas en lenguaje natural, interpretadas mediante modelos de NLP (ej. BERT, GPT), lo que aportaría una interacción más natural y flexible.
* Aprendizaje automático para ajuste dinámico: integrar algoritmos de machine learning que permitan ajustar automáticamente las TPCs a partir de las partidas jugadas, adaptando las probabilidades a patrones reales de uso y reduciendo el esfuerzo manual de calibración.
* Despliegue en la nube y escalabilidad: evaluar el sistema bajo carga masiva en entornos cloud, incorporando balanceadores de carga, replicación de bases de datos y mecanismos de autoescalado. Esto permitiría comprobar su viabilidad como aplicación de gran escala.
* Gamificación avanzada: añadir elementos motivacionales como ranking de usuarios, logros y recompensas virtuales, lo que incrementaría la retención de usuarios y su disposición a interactuar de manera recurrente con el sistema.

## 5.5 Conclusión final

En conclusión, el proyecto ha logrado cumplir con éxito los objetivos propuestos, demostrando la capacidad de las redes bayesianas para sustentar sistemas de inferencia probabilística en contextos de entretenimiento interactivo.

La implementación práctica de una arquitectura híbrida de bases de datos, combinada con un backend ligero en FastAPI y un frontend moderno en Vue.js y TailwindCSS, refuerza el valor del sistema tanto en el plano académico como en el técnico.

El sistema constituye un ejemplo concreto de cómo la inteligencia artificial y el big data pueden integrarse en experiencias gamificadas con fines formativos y lúdicos. Las limitaciones detectadas no deben entenderse como fallos, sino como oportunidades de mejora y como estímulo para futuras investigaciones.

Este Trabajo de Fin de Máster sienta, por tanto, una base sólida para el desarrollo de proyectos posteriores en la intersección entre IA, gamificación y sistemas híbridos de datos, ofreciendo un marco metodológico y técnico que puede ser replicado, extendido y enriquecido en distintos ámbitos académicos e industriales.

# Capítulo 6. Referencias bibliográficas

- Barber, D. (2012). Bayesian reasoning and machine learning. Cambridge University Press.  
- Buntine, W. (1996). A guide to the literature on learning probabilistic networks from data. IEEE Transactions on Knowledge and Data Engineering, 8(2), 195-210.  
- Cowell, R. G., Dawid, A. P., Lauritzen, S. L., & Spiegelhalter, D. J. (2007). Probabilistic networks and expert systems. Springer Science & Business Media.  
- Dean, T., & Kanazawa, K. (1989). A model for reasoning about persistence and causation. Computational Intelligence, 5(3), 142-150.  
- Friedman, N., Geiger, D., & Goldszmidt, M. (1997). Bayesian network classifiers. Machine Learning, 29(2-3), 131-163.  
- Jensen, F. V., & Nielsen, T. D. (2007). Bayesian networks and decision graphs. Springer Science & Business Media.  
- Koller, D., & Friedman, N. (2009). Probabilistic graphical models: Principles and techniques. MIT Press.  
- Murphy, K. P. (2012). Machine learning: A probabilistic perspective. MIT Press.  
- Pearl, J. (1988). Probabilistic reasoning in intelligent systems: Networks of plausible inference. Morgan Kaufmann.  
- Russell, S., & Norvig, P. (2021). Artificial intelligence: A modern approach (4th ed.). Pearson.  
- Scutari, M., & Denis, J. B. (2021). Bayesian networks: With examples in R. Chapman and Hall/CRC.  
- Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction (2nd ed.). MIT Press.  
- Van Rossum, G., & Drake, F. L. (2009). Python 3 reference manual. CreateSpace.  
- Grinberg, M. (2018). Flask web development: Developing web applications with Python. O’Reilly Media.  
- Ramalho, L. (2015). Fluent Python: Clear, concise, and effective programming. O’Reilly Media.  
- Vue.js Core Team. (2023). Vue.js: The progressive JavaScript framework. <https://vuejs.org/>- FastAPI Project. (2023). FastAPI: Modern, fast (high-performance), web framework for building APIs with Python. <https://fastapi.tiangolo.com/>- MongoDB Inc. (2023). MongoDB documentation. <https://www.mongodb.com/docs/>- Oracle Corporation. (2023). MySQL 8.0 Reference Manual. <https://dev.mysql.com/doc/>- Tailwind Labs. (2023). TailwindCSS documentation. <https://tailwindcss.com/docs/>

# Capítulo 7. Anexos técnicos: fragmentos de código relevantes

Este anexo recopila los fragmentos más representativos del código del proyecto. Los listados se presentan en formato monoespaciado, con recortes cuando el tamaño del archivo completo excede el espacio razonable para la memoria. Se priorizan funciones clave del backend (endpoints, inferencia, persistencia) y del frontend (router, API cliente, arranque).

## 7.1 Backend (Python/FastAPI)

### 7.1.1 inferencia\_multiple.py — Motor de inferencia múltiple

import json  
import pandas as pd  
import numpy as np  
from pgmpy.models import DiscreteBayesianNetwork  
from pgmpy.factors.discrete import TabularCPD  
from pgmpy.inference import VariableElimination  
from sqlalchemy import create\_engine  
  
def cargar\_red(config\_path, df):  
 with open(config\_path) as f:  
 config = json.load(f)  
  
 atributos = [a for a in config["atributos"] if a in df.columns]  
 df = df.dropna(subset=["personaje"]).copy()  
 df[atributos] = df[atributos].fillna(0).astype(int)  
  
 personajes = df["personaje"].unique().tolist()  
 df["personaje"] = df["personaje"].map({p: i for i, p in enumerate(personajes)})  
  
 modelo = DiscreteBayesianNetwork([(a, "personaje") for a in atributos])  
 cpds = []  
  
 for a in atributos:  
 p = df[a].mean()  
 cpds.append(TabularCPD(variable=a, variable\_card=2, values=[[1 - p], [p]]))  
  
 group = df.groupby(atributos + ["personaje"]).size().unstack(fill\_value=0)  
 index = pd.MultiIndex.from\_product([sorted(df[a].unique()) for a in atributos], names=atributos)  
 group = group.reindex(index, fill\_value=0)  
 matriz = group.T.values.astype(float)  
  
 for i in range(matriz.shape[1]):  
 total = matriz[:, i].sum()  
 matriz[:, i] = matriz[:, i] / total if total > 0 else 1.0 / len(personajes)  
  
 cpd\_personaje = TabularCPD(  
 variable="personaje",  
 variable\_card=len(personajes),  
 values=matriz.tolist(),  
 evidence=atributos,  
 evidence\_card=[len(df[a].unique()) for a in atributos]  
 )  
 cpds.append(cpd\_personaje)  
  
 modelo.add\_cpds(\*cpds)  
 assert modelo.check\_model()  
  
 return VariableElimination(modelo), personajes  
  
def combinar\_resultados(resultados):  
 combinada = {}  
 for resultado in resultados:  
 for k, v in resultado.items():  
 combinada[k] = combinada.get(k, 1) \* v  
 total = sum(combinada.values())  
 return {k: v / total for k, v in combinada.items()} if total else {k: 1/len(combinada) for k in combinada}  
  
def inferir\_personaje\_desde\_redes(observaciones: dict):  
 print("⚡ INFERENCIA ACTIVADA:", observaciones)  
 engine = create\_engine("mysql+mysqlconnector://fastapi:frambuesa22@localhost/personajes\_marvel")  
 df = pd.read\_sql("SELECT \* FROM personajes", engine)  
 df["personaje"] = df["nombre"]  
 if "id" in df.columns:  
 df = df.drop(columns=["id"])  
 redes = ["poderes", "afiliaciones\_heroes", "afiliaciones\_villanos", "especie", "origen", "armas", "genero\_ocupacion"]  
 resultados = []  
  
 for red in redes:  
 try:  
 infer, personajes = cargar\_red(f"./adivinador\_backend/bayes\_tematica/config\_{red}.json", df)  
 evidencia\_valida = {k: v for k, v in observaciones.items() if k in infer.variables}  
 if evidencia\_valida:  
 resultado = infer.query(["personaje"], evidence=evidencia\_valida)  
 resultados.append({personajes[i]: float(prob) for i, prob in enumerate(resultado.values)})  
 else:  
 resultados.append({p: 1/len(personajes) for p in personajes})  
 except Exception as e:  
 print(f"❌ Error en red {red}: {e}")  
  
 final = combinar\_resultados(resultados)  
 ordenado = sorted(final.items(), key=lambda x: x[1], reverse=True)  
 print("🔍 TOP 3:", ordenado[:3])  
 return ordenado[:5]

### 7.1.2 inferencia.py — Utilidades de inferencia

from fastapi import APIRouter, HTTPException  
from pydantic import BaseModel  
from typing import Dict  
import json  
import pandas as pd  
import numpy as np  
from db\_sql import cargar\_personajes  
from pgmpy.models import DiscreteBayesianNetwork  
from pgmpy.factors.discrete import TabularCPD  
from pgmpy.inference import VariableElimination  
from sqlalchemy import create\_engine  
  
router = APIRouter()  
  
class RespuestasUsuario(BaseModel):  
 respuestas: Dict[str, int | None]  
  
def cargar\_red(config\_path, df):  
 with open(config\_path) as f:  
 config = json.load(f)  
  
 atributos = [a for a in config["atributos"] if a in df.columns]  
 df = df.dropna(subset=["personaje"]).copy()  
 df[atributos] = df[atributos].fillna(0).astype(int)  
  
 personajes = df["personaje"].unique().tolist()  
 df["personaje"] = df["personaje"].map({p: i for i, p in enumerate(personajes)})  
  
 modelo = DiscreteBayesianNetwork([(a, "personaje") for a in atributos])  
 cpds = []  
  
 for a in atributos:  
 p = df[a].mean()  
 cpds.append(TabularCPD(variable=a, variable\_card=2, values=[[1 - p], [p]]))  
  
 group = df.groupby(atributos + ["personaje"]).size().unstack(fill\_value=0)  
 index = pd.MultiIndex.from\_product([sorted(df[a].unique()) for a in atributos], names=atributos)  
 group = group.reindex(index, fill\_value=0)  
 matriz = group.T.values.astype(float)  
  
 for i in range(matriz.shape[1]):  
 total = matriz[:, i].sum()  
 matriz[:, i] = matriz[:, i] / total if total > 0 else 1.0 / len(personajes)  
  
 cpd\_personaje = TabularCPD(  
 variable="personaje",  
 variable\_card=len(personajes),  
 values=matriz.tolist(),  
 evidence=atributos,  
 evidence\_card=[len(df[a].unique()) for a in atributos]  
 )  
 cpds.append(cpd\_personaje)  
  
 modelo.add\_cpds(\*cpds)  
 assert modelo.check\_model()  
  
 return VariableElimination(modelo), personajes  
  
def combinar\_resultados(resultados):  
 combinada = {}  
 for resultado in resultados:  
 for k, v in resultado.items():  
 combinada[k] = combinada.get(k, 1) \* v  
 total = sum(combinada.values())  
 return {k: v / total for k, v in combinada.items()} if total else {k: 1/len(combinada) for k in combinada}  
  
@router.post("/inferir")  
def inferir\_personaje(datos: RespuestasUsuario):  
 print("⚡ INFERENCIA ACTIVADA:", datos.respuestas)  
 try:  
 engine = create\_engine("mysql+mysqlconnector://fastapi:frambuesa22@localhost/personajes\_marvel")  
 df = cargar\_personajes()  
 df["personaje"] = df["nombre"]  
 if "id" in df.columns:  
 df = df.drop(columns=["id"])  
  
 redes = [  
 "poderes", "afiliaciones\_heroes", "afiliaciones\_villanos",  
 "especie", "origen", "armas", "genero\_ocupacion"  
 ]  
 resultados = []  
  
 for red in redes:  
 try:  
 infer, personajes = cargar\_red(f"./adivinador\_backend/bayes\_tematica/config\_{red}.json", df)  
 evidencia\_valida = {k: v for k, v in datos.respuestas.items() if k in infer.variables}  
 if evidencia\_valida:  
 resultado = infer.query(["personaje"], evidence=evidencia\_valida)  
 resultados.append({personajes[i]: float(prob) for i, prob in enumerate(resultado.values)})  
 else:  
 resultados.append({p: 1/len(personajes) for p in personajes})  
 except Exception as e:  
 print(f"❌ Error en red {red}: {e}")  
  
 final = combinar\_resultados(resultados)  
 ordenado = sorted(final.items(), key=lambda x: x[1], reverse=True)  
 print("🔍 TOP 3:", ordenado[:3])  
 return {"resultado": ordenado[:5]}  
  
 except Exception as e:  
 print("❌ ERROR GENERAL en inferencia:", e)  
 raise HTTPException(status\_code=500, detail="Error en inferencia múltiple")

### 7.1.3 preguntas.py — Gestión de preguntas

from fastapi import APIRouter  
from db import db  
  
router = APIRouter()  
  
@router.get("/activas")  
def obtener\_preguntas\_activas():  
 coleccion = db["preguntas"]  
 preguntas = list(coleccion.find({}, {"\_id": 0}))  
 return {"preguntas": preguntas}

### 7.1.4 partidas.py — Endpoints de partidas y persistencia

from fastapi import APIRouter  
from pydantic import BaseModel  
from typing import List, Dict, Union  
from db import db # Este debe ser tu módulo que exporta la conexión a Mongo  
from datetime import datetime  
  
router = APIRouter()  
  
# Estructura de un ítem del resultado final  
class ResultadoItem(BaseModel):  
 personaje: str  
 probabilidad: float  
  
# Estructura de la partida completa  
class Partida(BaseModel):  
 respuestas: Dict[str, Union[int, None]]  
 resultado: List[List[Union[str, float]]]  
  
@router.post("/guardar\_partida")  
def guardar\_partida(partida: Partida):  
 partida\_dict = partida.dict()  
 partida\_dict["timestamp"] = datetime.utcnow().isoformat()  
 db["partidas"].insert\_one(partida\_dict)  
 return {"mensaje": "✅ Partida guardada correctamente"}

## 7.2 Configuración de redes bayesianas (JSON)

### 7.2.1 config\_poderes.json — Red temática de poderes

{  
 "atributos": [  
 "tiene\_superfuerza", "puede\_volar", "tiene\_regeneracion", "tiene\_elasticidad",  
 "tiene\_telepatia", "tiene\_telequinesis", "tiene\_invisibilidad", "tiene\_velocidad",  
 "tiene\_vision\_laser", "controla\_fuego", "controla\_hielo", "tiene\_magia",  
 "controla\_electricidad", "puede\_cambiar\_forma"  
 ]  
}

### 7.2.2 config\_origen.json — Red temática de origen

{  
 "atributos": [  
 "origen\_tierra", "origen\_extraterrestre", "origen\_magico", "origen\_tecnologico"  
 ]  
}

### 7.2.3 config\_armas.json — Red temática de armas

{  
 "atributos": [  
 "usa\_escudo", "usa\_martillo", "usa\_espadas", "usa\_pistolas",  
 "usa\_espada\_de\_alma", "usa\_armas\_tecnologicas"  
 ]  
}

### 7.2.4 confitg\_genero\_ocupacion.json — Red de género/ocupación

{  
 "atributos": [  
 "genero\_hombre", "genero\_mujer", "es\_adolescente",  
 "es\_cientifico", "es\_soldado", "es\_profesional"  
 ]  
}

## 7.3 Frontend (Vue.js + TailwindCSS)

### 7.3.1 router/index.js — Definición de rutas principales

import { createRouter, createWebHistory } from 'vue-router'  
import Home from '../components/PantallaInicio.vue'  
import Quiz from '../components/Inferencia.vue'  
import CrearPregunta from '../components/CrearPregunta.vue'  
import Historial from '../components/HistorialPartidas.vue'  
  
const routes = [  
 { path: '/', component: Home },  
 { path: '/jugar', component: Quiz },  
 { path: '/crear', component: CrearPregunta },  
 { path: '/historial', component: Historial }  
]  
  
const router = createRouter({  
 history: createWebHistory(),  
 routes,  
})  
  
export default router

### 7.3.2 src/api.js — Cliente Axios preconfigurado

// src/api.js  
import axios from 'axios'  
  
const api = axios.create({  
 baseURL: '<http://localhost:8000>',  
})  
  
export default api

### 7.3.3 main.js — Arranque de la aplicación

import { createApp } from 'vue'  
import App from './App.vue'  
import router from './router'  
import './assets/tailwind.css'  
  
  
createApp(App).use(router).mount('#app')

### 7.3.4 tailwind.css — Base de estilos utilitarios

@tailwind base;  
@tailwind components;  
@tailwind utilities;

## 7.4 Notas de integración

- Los endpoints del backend consumen y producen JSON; la validación se realiza con Pydantic.  
- La selección de preguntas se basa en heurística de ganancia de información y estado de la red.  
- El router de Vue expone las vistas: Inicio, Inferencia, CrearPregunta e Historial.  
- El cliente Axios centraliza la baseURL y permite añadir interceptores para auth/logging.