
Generación de historias por secciones utilizando
Large Language Model
Story generation by sections using Large
Language Models



Trabajo de Fin de Grado
Curso 2024–2025

Autor

Pablo Enrique Padial Iniesta

Director

Pablo Gervás Gómez-Navarro

Gonzalo Méndez Pozo

Grado en Ingeniería de Computadores
Facultad de Informática
Universidad Complutense de Madrid

Generación de historias por secciones
utilizando Large Language Model
Story generation by sections using Large
Language Models

Trabajo de Fin de Grado en Ingeniería de Computadores

Autor

Pablo Enrique Padial Iniesta

Director

Pablo Gervás Gómez-Navarro

Gonzalo Méndez Pozo

Convocatoria: *Septiembre 2025*

Grado en Ingeniería de Computadores

Facultad de Informática

Universidad Complutense de Madrid

1 de septiembre de 2025

Dedicatoria

A mis padres y mi hermana.

Agradecimientos

A la familia y amigos por el apoyo y la ayuda.

Resumen

Generación de historias por secciones utilizando Large Language Model

La mejora de los LLMs (*Large Language Models*) han permitido su uso en aspectos creativos como la generación de historias o relatos. Este trabajo presenta una aplicación llamada **historIAs** para la creación de historias asistidas por modelos LLMs, mediante un flujo guiado por secciones: *trama e hilo simbólico*, *descripción del mundo*, *personajes principales*, *descripción de escenarios*, *estructura de la historia* y finalmente la escritura de los capítulos. Este sistema permite personalizar y editar la historia antes y durante la redacción. La información se almacena en una base de datos dando la posibilidad al usuario de continuar con la historia en cualquier otro momento.

Para evaluar la calidad de las historias, se plantea una valoración de diferentes usuarios buscando contrastar la producción narrativa del modelo con la escritura humana e identificar fortalezas y limitaciones. Finalmente se discute el encaje de la generación de historias mediante LLMs como apoyo para escritores en su proceso creativo.

Palabras clave

LLM, generación de historias, escritura asistida, IA, Llama, Transformer, token.

Abstract

Story generation by sections using Large Language Models

The improvement of *Large Language Models* (LLMs) has enabled their use in creative domains such as story and narrative generation. This work presents an application called **historIAs** for the creation of stories assisted by LLMs, through a workflow structured into sections: *plot and symbolic thread*, *world description*, *main characters*, *setting description*, *story structure* and finally, the writing of chapters. This system allows the user to personalize and edit the story both before and during the writing process. All information is stored in a database, enabling the user to resume the story at any moment.

To assess the quality of the generated stories, evaluations from different users were conducted, aiming to contrast the narrative production of the model with human writing and to identify strengths and limitations. Finally, the work discusses the role of LLM-based story generation as a creative aid for writers in their storytelling process.

Keywords

LLM, story generation, assisted writing, AI, Llama, Transformers, token.

Introduction

In recent years, the way of tackling problems, searching for information, or generating ideas has been changing rapidly. The current boom in generative artificial intelligence has fully broken into most sectors of society. The ease of access to these tools, together with the quality of the results in generating texts, summaries, images, videos, or in automating tedious tasks, has led to the widespread use of these technologies even in everyday activities. The use, often excessive, of artificial intelligence models in creative fields has generated rejection in part of society, especially in sectors such as illustration, translation, or musical composition, where there is concern about the loss of human value and employment. In the case of *story generation* something similar occurs: many writers and readers question the artistic value of a text produced by a machine, considering that it lacks the emotional depth and life experience that a human author provides. In addition, it is important to highlight the biases arising from design errors or present in the training data, which can lead to reproducing stereotypes or forms of discrimination. Despite this, creative writing generated by LLMs (*Large Language Models*) can also be understood as an opportunity, not to replace writers or novelists, but as support for creation and imagination. In this context the project is presented, which aims to serve as a guide and support for authors who want to write stories, offering a structured and editable workflow. In addition, an analysis of the generated stories is proposed in order to evaluate their narrative quality and check the extent to which they can be considered comparable to those created by a human author. This evaluation does not seek to replace the writer, but to assess the potential of artificial intelligence as a creative tool, as well as to identify its current limitations in terms of originality, coherence, and literary depth.

Motivation

There are currently multiple websites and applications that allow stories to be generated automatically using artificial intelligence (Summarizer [19], Wrizzle [24]) that can create short stories from a topic and a small configuration. However, in most cases personalization by the user is minimal, without offering real control over the creative process. This results in little or no editing where the user can only restart the entire generation. Faced with these limited options, the main motivation

of this project is to develop a tool that not only generates a complete story, but also offers a high level of editing and personalization. In this way, to help and guide users with their own stories through a high level of detail. To this end, the narrative is divided into key sections: the *plot and symbolic thread* that sets the general course of the work, the *main characters* where their attributes and description are defined, the *main settings*, and the *structure of the chapters*. The objective is to build the story through sections where the user can rewrite, request modifications, or manually edit each one so that it fits their preferences. In addition, there is a quick generation mode intended to generate the story from the data entered in a questionnaire, creating stories much more quickly, sacrificing part of the editing and personalization characteristic of the other option. This gives rise to another analysis: to evaluate whether the quality and coherence of the generated story vary depending on the creation mode (see Section 3.2.1 for more details).

Objectives

The main objective of this TFG is to develop an application that guides the creation of stories using language models, allowing the structured generation of stories and offering a high degree of editing and personalization to the user. The specific objectives proposed by the project are:

- Implement an LLM (the **Groq** server is used) that enables story generation, ensuring good *token* management so as not to exceed the model's usage limit.
- Design an interactive user interface that allows the user to generate, rewrite, modify, and manually edit each section of the story for a better fit to their preferences.
- Design the narrative flow by dividing the story into key sections (plot, characters, settings, world, chapters) that facilitate editing and personalization.
- Store the generated sections in a database (MongoDB) to manage data persistence and allow the user to resume the story at any time.
- Offer two modes of story generation: a *detailed* one and a *quick* one, to adapt to the user's needs and to compare the quality and coherence of the stories generated depending on the creation mode.
- Evaluate the narrative quality of the generated stories, analyzing their coherence and literary depth in comparison with those created by a human author.
- Identify the current limitations of LLMs in story generation, proposing lines of future improvements.

Structure of the report

The project is divided into several chapters that address different aspects of the development and evaluation of the tool:

- **Chapter 1: Introduction:** Presents the project’s context, motivation, and objectives.
- **Chapter 2: State of the question:** Previous work and the main lines of research related to automatic story generation are reviewed. The evolution of AI from Alan Turing to the present day and the technical foundations of LLMs are presented. In addition, several published novels written by artificial intelligence are discussed.
- **Chapter 3: Description of the work:** Presents the main chapter of the project where everything related to the application is developed. The chapter begins with a section devoted to presenting the main resources and tools used during the project: `Groq`, `Flet` and `MongoDB`. We continue with Section 3.2, where the two story-creation models, flowcharts, and component diagrams are presented in detail. A section is included that details storage in the database and finally reference is made to the importance of *system prompts* to guide the model, and the most relevant ones are shown. To finish this section, the screens of the application and the operation of `Flet` are detailed.
- **Chapter 4: Development and evaluation of generated stories:** In this chapter an example of creating a story in *detailed mode* is shown and its subsequent evaluation through surveys.
- **Chapter 5: Conclusions and future work:** Presents a final reflection on narrative generation by LLMs, the improvements introduced through section-based creation, and finally a set of lines for future work.

Conclusions and Future Work

This work presents **historIA**s, an application designed to generate stories in sections, leveraging LLMs to achieve a higher degree of personalization and user editing both before and during creation. The application stores information in a database so that users can resume a story at any time. There is also a *quick mode* in which, after completing a short questionnaire, the model directly generates the story using the information provided.

After conducting numerous tests, it was observed that the generated stories frequently exhibited a series of similarities and biases. When the themes were similar, the model repeated character names and produced similar descriptions. If the *prompt* was not very detailed, it would introduce magical elements at any point and almost always relied on the trope of the protagonist having to defeat the villain. Coherence was maintained in most cases, although there were instances where, for example, in the *setting description* section, it created a setting that ultimately did not appear in the story. Ultimately, it was observed that these models, despite increasingly maintaining the coherence and thread of the story, are still far inferior to the writing ability of a professional author. Linguistic richness is more limited, often repeating words, expressions, or even whole sentences. The stories are quite simple in quality; there are no twists, and they follow a linear structure, which fails to evoke strong emotion in the reader. In the study by Tian, Huang, et al. [20], Figure 5.1 shows the evolution of the characters' fortune:

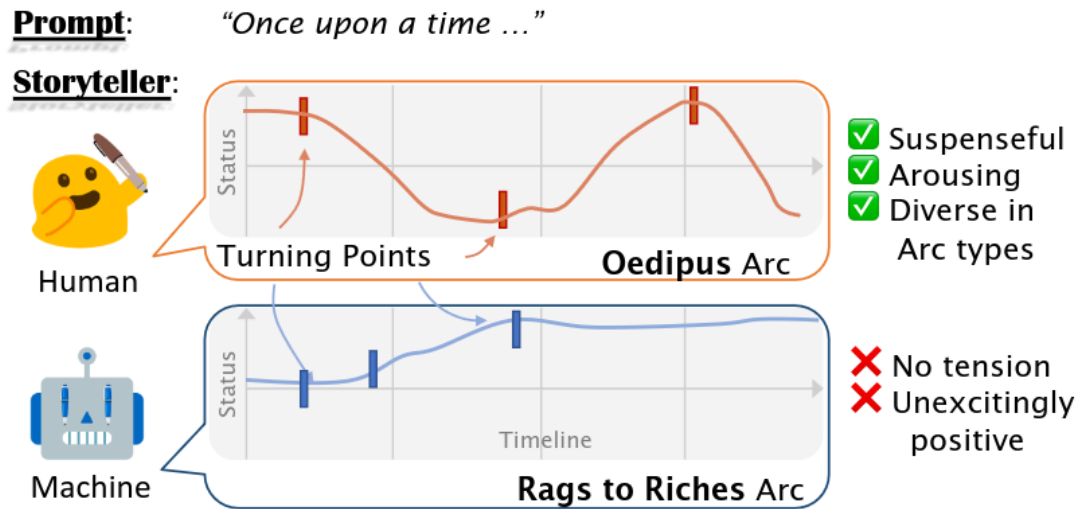


Figure 1: Comparison of narrative arcs between humans and language models (chart source [20])

This graph presents a comparison between the narrative arcs and the positions of turning points in human- and LLM-generated narratives. The vertical axis shows a character’s fortune (from bad to good), and the horizontal axis represents the story’s timeline. Compared with traditional writing, LLMs tend to employ happier and less complex arcs, introduce turning points earlier, and include less suspense and fewer setbacks in their narratives. The same trend is apparent in the stories created with this application.

Another important goal of the project was to compare story generation in *quick mode*—more similar to what is currently on the market—with the project’s proposal, the *detailed mode*. After numerous tests, coherence levels were similar, although the *detailed mode* did produce longer and more developed stories. One of the most notable features of the project has been the section-by-section creation workflow. This structure not only facilitates personalization and narrative control for the user, but has also proven to be a very useful tool for generating ideas and stimulating creativity. Although the literary quality of the model outputs still shows limitations, the step-by-step process of building a story is dynamic, enjoyable, and enriching, according to users who tested the application. In this sense, the application serves not only a generative function but also an inspirational one, acting as a starting point for writers to develop deeper and more original narratives from the ideas produced by the system.

For this reason, continuing with the section-based approach is a promising line of work. To that end, the author could be given more flexibility to choose which sections to create, rather than being limited to the five predefined ones. Another interesting direction is to provide greater flexibility to define a series of events or actions that the user wants to occur during the story, as well as to allow modifications to sections or chapters the user has already completed. The greatest challenge throughout the project was the model’s token limit imposed by **Groq**; a larger token budget would allow for more sections and chapters. Another improvement would

be to let the user introduce new characters; due to the token limit, the system is currently restricted to a maximum of four characters. It would also be valuable to include an initial configuration where aspects such as type of novel, target audience, narrative style, etc. are specified. Finally, the system prompts could be improved by applying prompting techniques.

Índice

Introduction	XIII
Conclusions and Future Work	XVII
1. Introducción	1
1.1. Motivación	2
1.2. Objetivos	2
1.3. Estructura de la memoria	3
2. Estado de la Cuestión	5
2.1. Orígenes y fundamentos generales de la Inteligencia Artificial	5
2.1.1. Fundamentos técnicos de los LLMs	7
2.2. Generación de historias	8
3. Descripción del Trabajo	11
3.1. Herramientas utilizadas	11
3.2. Diseño de la aplicación historIAs	13
3.2.1. Modos de generación de historia	13
3.2.2. Diagramas de flujo	14
3.2.3. Diagrama de componentes	17
3.2.4. Interacción con el modelo LLM	18
3.2.5. Almacenamiento en la base de datos	22
3.2.6. Gestión de prompts	24
3.3. Interfaz de usuario	25
3.3.1. Pantallas y funcionalidades	25
3.3.2. Arquitectura de la interfaz	30
4. Desarrollo y evaluación de historias generadas	33
4.1. Ejemplo de una historia en el modo detallado	33
4.2. Evaluación de la historia	38
5. Conclusiones y Trabajo Futuro	41

Bibliografía	44
A. Modelos Groq	47
B. Ejemplo de una historia y encuestas de los usuarios	49
B.1. Ejemplo de historia en modo detallado	49
B.2. Encuestas realizadas por los usuarios	54

Índice de figuras

1. Comparison of narrative arcs between humans and language models (chart source [20])	XVIII
2.1. Número de artículos científicos relacionados con los LLMs en los últimos años (fuente del gráfico [13])	6
2.2. Línea del tiempo de lanzamientos de LLM (fuente del gráfico [13])	8
3.1. Diagrama de flujo <i>Inicio de sesión</i>	15
3.2. Diagrama de flujo <i>Flujo principal</i>	16
3.3. Diagrama de componentes	17
3.4. Diagrama de secuencia	19
3.5. Precisión según la temperatura	22
3.6. Ejemplo de almacenamiento en la base de datos: <i>info_usuarios</i>	23
3.7. Ejemplo almacenamiento en la base de datos para el <i>modo detallado: sección</i>	23
3.8. Ejemplo de almacenamiento en la base de datos para el <i>modo detallado: contexto</i>	24
3.9. Ejemplo de almacenamiento en la base de datos para el <i>modo rápido: capítulos</i>	24
3.10. Pantalla de inicio de sesión	26
3.11. Pantalla de registro de usuario	27
3.12. Pantalla principal	28
3.13. Pantalla de creación de historia en modo detallado	29
3.14. Pantalla de creación de historia en modo rápido	30
3.15. Organización de las pantallas en Flet	31
4.1. Historia de ejemplo: <i>Trama e hilo simbólico</i> . Mensaje introducido: “Quiero escribir una historia realista con tonos de humor en la que suceda algo misterioso. Quiero que uno de los protagonistas tenga una mascota que participe en la acción. Además, quiero que se haga referencia a sucesos reales del presente o del pasado.”	34
4.2. Historia de ejemplo: Modificar - <i>Trama e hilo simbólico</i> . Mensaje introducido: “El evento suceso real del pasado debe ser un evento histórico de la realidad.”	35

4.3.	Historia de ejemplo: Modificación - <i>Trama e hilo simbólico</i>	35
4.4.	Historia de ejemplo: Características - <i>Personajes principales</i> . Mensaje introducido: “ <i>Los protagonistas son amigos de la infancia</i> .”	36
4.5.	Historia de ejemplo: <i>Personajes principales</i>	36
4.6.	Historia de ejemplo: <i>Capítulo 1</i>	37
4.7.	Historia de ejemplo: <i>Capítulo 4</i>	38
5.1.	Comparación de arcos narrativos entre humanos y modelos de len- guaje (fuente del gráfico [20])	42

Índice de tablas

3.1. Límites de uso de los modelos empleados.	12
3.2. Ajuste de <code>temperature</code> según el tipo de tarea	21
A.1. Tabla LLM Groq	47

Introducción

Durante los últimos años la forma de afrontar problemas, de buscar información o de generar ideas está cambiando de manera acelerada. El auge que vive actualmente la inteligencia artificial generativa ha irrumpido de lleno en la mayoría de sectores de la sociedad. La facilidad de acceso a estas herramientas, junto con la calidad de los resultados en la generación de textos, resúmenes, imágenes, vídeos o en la automatización de tareas tediosas, ha provocado un uso generalizado de estas tecnologías incluso en actividades cotidianas. El uso, en muchas ocasiones abusivo, de modelos de inteligencia artificial en ámbitos creativos ha generado rechazo en parte de la sociedad, especialmente en sectores como la ilustración, la traducción o la composición musical, donde existe preocupación por la pérdida de valor humano y de empleo. En el caso de la *generación de historias* ocurre algo similar: muchos escritores y lectores cuestionan el valor artístico de un texto producido por una máquina, al considerar que carece de la profundidad emocional y de la experiencia vital que aporta un autor humano. Además, es importante destacar los sesgos derivados de errores de diseño o presentes en los datos de entrenamiento, que pueden llevar a reproducir estereotipos o formas de discriminación. Pese a ello, la escritura creativa generada por los LLMs (*Large Language Models*) puede entenderse también como una oportunidad, no para sustituir a escritores o novelistas, si no como apoyo a la hora de creación e imaginación. En este ámbito se presenta el proyecto, que pretende servir como guía y apoyo a autores que quieran escribir historias, ofreciendo un flujo estructurado y editable. Además, se plantea un análisis de las historias generadas con el fin de evaluar su calidad narrativa y comprobar hasta qué punto pueden considerarse comparables a las creadas por un autor humano. Esta evaluación no busca sustituir al escritor, sino valorar el potencial de la inteligencia artificial como herramienta creativa, así como identificar sus limitaciones actuales en cuanto a originalidad, coherencia y profundidad literaria.

1.1. Motivación

Actualmente existen múltiples páginas web y aplicaciones que permiten generar historias de manera automática mediante inteligencia artificial (Summarizer [19], Wrizzle [24]) que pueden crear pequeños relatos a partir de un tema y una pequeña configuración. Sin embargo, en la mayoría de casos la personalización por parte del usuario es mínima, sin ofrecer un control real sobre el proceso creativo. Esto provoca una edición baja o inexistente donde el usuario únicamente puede reiniciar la generación completa. Frente a estas limitadas opciones, la motivación principal de este proyecto es desarrollar una herramienta que no solo genere una historia completa, sino que también ofrezca un gran nivel de edición y personalización. De esta forma, ayudar y guiar a los usuarios con sus propias historias a través de un gran nivel de detalle. Para ello, la narrativa se divide en secciones clave: la *trama e hilo simbólico* que marca el rumbo general de la obra, los *personajes principales* donde se define sus atributos y descripción, los *escenarios principales* y la *estructura de los capítulos*. El objetivo es construir la historia a través secciones donde el usuario pueda reescribir, pedir modificaciones o editar manualmente cada una de ellas y así se ajuste a sus preferencias. Además, se dispone de un modo de generación rápida en el que se pretende generar la historia a partir de los datos introducidos en un cuestionario, creando historias de forma mucho más rápida, sacrificando parte de la edición y personalización característica de la otra opción. Con esto surge otro análisis: evaluar si la calidad y coherencia de la historia generada varían en función del modo de creación (ver la sección 3.2.1 para más detalles).

1.2. Objetivos

El objetivo principal de este TFG consiste en desarrollar una aplicación que guíe la creación de historias mediante modelos de lenguaje, permitiendo la generación estructurada de historias y ofreciendo un alto grado de edición y personalización al usuario. Los objetivos específicos que plantea el proyecto son:

- Implementar un LLM (se utiliza el servidor **Groq**) que permita la generación de historias, asegurando una buena gestión de *tokens* para no superar el límite de uso del modelo.
- Diseñar una interfaz de usuario interactiva que permita al usuario generar, reescribir, modificar y editar manualmente cada sección de la historia para un mayor ajuste a sus preferencias.
- Diseñar el flujo narrativo dividiendo la historia en secciones clave (trama, personajes, escenarios, mundo, capítulos) que faciliten la edición y personalización.

- Almacenar las secciones generadas en una base de datos (MongoDB) para gestionar la persistencia de los datos y permitir que el usuario pueda retomar la historia en cualquier momento.
- Ofrecer dos modos de generación de historias: uno *detallado* y otro *rápido*, para adaptarse a las necesidades del usuario y comparar la calidad y coherencia de las historias generadas en función del modo de creación.
- Evaluar la calidad narrativa de las historias generadas, analizando su coherencia y profundidad literaria en comparación con las creadas por un autor humano.
- Identificar las limitaciones actuales de los LLMs en la generación de historias proponiendo líneas de mejoras futuras.

1.3. Estructura de la memoria

El proyecto está dividido en varios capítulos que abordan diferentes aspectos del desarrollo y evaluación de la herramienta:

- **Capítulo 1: Introducción:** Presenta el contexto del proyecto, la motivación y los objetivos.
- **Capítulo 2: Estado de la cuestión:** Se revisan los trabajos previos y las principales líneas de investigación relacionadas con la generación automática de historias. Se expone la evolución de la IA desde Alan Turing hasta el día de hoy y los fundamentos técnicos de los LLMs. Además, se comentan varias novelas publicadas escritas por inteligencia artificial.
- **Capítulo 3: Descripción del trabajo:** Presenta el capítulo principal del proyecto donde se desarrolla todo lo relacionado con la aplicación. El capítulo inicia con una sección dedicada a exponer los recursos y herramientas principales utilizadas durante el proyecto: **Groq**, **Flet** y **MongoDB**. Seguimos con la Sección 3.2 donde se expone de forma detallada los dos modelos de creación de historias, diagramas de flujo y de componentes. Se incluye un apartado donde se detalla el almacenamiento en la base de datos y finalmente se hace referencia a la importancia de los *prompts del sistema* para guiar al modelo, y se muestran los más relevantes. Para terminar esta sección, se detalla las pantallas de la aplicación y el funcionamiento de **Flet**.
- **Capítulo 4: Desarrollo y evaluación de historias generadas:** En este capítulo se muestra un ejemplo de la creación de una historia en el *modo detallado* y su posterior evaluación mediante unas encuestas.

- **Capítulo 5: Conclusiones y trabajo futuro:** Presenta una reflexión final acerca de la generación narrativa por parte de los LLMs, las mejoras introducidas mediante la creación por secciones y finalmente se exponen una serie de líneas para trabajos futuros.

Estado de la Cuestión

Los LLMs (Large Language Models) han transformado radicalmente el procesamiento del lenguaje y la generación de texto. A lo largo de este capítulo se presenta un análisis de los LLMs, con un enfoque en su aplicación en la generación literaria mediante Inteligencia Artificial Generativa. Se incluye un cronograma con los hitos más relevantes, desde el surgimiento de los LLMs, gracias a la arquitectura *Transformers*, hasta el modelo *Llama 3.3-70b* empleado en el proyecto y destacando obras literarias generadas por LLMs.

2.1. Orígenes y fundamentos generales de la Inteligencia Artificial

La Inteligencia Artificial es un campo de la informática que busca desarrollar sistemas capaces de realizar tareas que, tradicionalmente, requerían de inteligencia humana, como el razonamiento, el aprendizaje o la comprensión del lenguaje. Se busca diseñar sistemas capaces de tomar decisiones y aprender a partir de ejemplos y experiencias.

Las primeras ideas sobre la Inteligencia Artificial se remontan a mediados del siglo XX, con hitos como el *Test de Turing* propuesto por Alan Turing en 1950 [22], considerado uno de los padres de la computación. Se trata de una prueba para evaluar la capacidad de una máquina de exhibir un comportamiento inteligente similar o indistinguible al de un ser humano.

Unos años más tarde en 1956, durante la conferencia de Dartmouth, *Dartmouth Summer Research Project on Artificial Intelligence*, organizada por John McCarthy, fue acuñado el término “inteligencia artificial” [12]. Durante las décadas de 1950 y 1960, comenzaron a surgir los primeros programas de IA centrados en la resolución de problemas lógicos y el procesamiento simbólico, como el Logic Theorist (1956)

[14] y el General Problem Solver (1959) [15]. Sin embargo, debido a las limitaciones computacionales y la falta de datos provocaron un estancamiento en el desarrollo.

A partir de los años 80 y 90, el desarrollo de los algoritmos de *machine learning* (aprendizaje automático) permitió que los patrones “aprendieran” a partir de datos.

No fue hasta los años 2000’s cuando pudo ser implementado, en parte, gracias a la disponibilidad de grandes volúmenes de datos y la disponibilidad de hardware económico con un gran poder de cálculo como fueron las primeras *Unidades de Procesamiento Gráfico* (GPU). Gracias a estos grandes avances tecnológicos, se pudieron construir las redes neuronales con cientos de capas de neuronas, dando nacimiento al termino *Deep Learning* (redes neuronales profundas), mediante las cuales se puede generar y entrenar modelos a partir de miles de datos.

En 2017 se presenta el artículo de investigación *Attention Is All You Need* donde se introduce la arquitectura *Transformers* [21]. La combinación de grandes volúmenes de datos, potentes unidades de procesamiento gráfico (GPU) y arquitecturas avanzadas como *Transformers* ha impulsado la actual generación de LLMs, capaces de producir texto coherente, contextual y creativo a una escala sin precedentes. La evolución de los LLMs ha marcado un punto de inflexión en la inteligencia artificial, permitiendo la generación de texto con coherencia, estilo y contexto. En particular, su aplicación en narrativa literaria ha abierto nuevas posibilidades creativas, desafiando límites de la autoría y la imaginación humana. En la Figura 2.1 se observa el aumento significativo de artículos científicos relacionados con los LLMs.

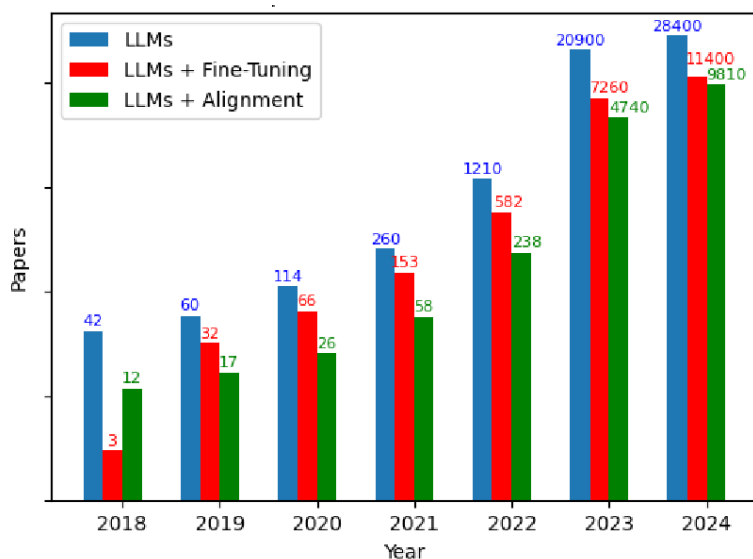


Figura 2.1: Número de artículos científicos relacionados con los LLMs en los últimos años (fuente del gráfico [13])

2.1.1. Fundamentos técnicos de los LLMs

La gran mayoría de LLMs se basan en la arquitectura *Transformers*, cuyos modelos se entrenan de forma auto-supervisada leyendo cantidades colosales de texto y aprendiendo a predecir la siguiente palabra de una frase. Desde BERT hasta GPT-5, los modelos han escalado en parámetros y capacidades. A finales de 2024, Meta presentó *Llama 3.3-70b*, un modelo de 70 mil millones de parámetros, optimizado para tareas multilingües y generación creativa.

La arquitectura del *Transformers* se basa en un *encoder* y un *decoder*. El *encoder* es la parte del *Transformers* encargada de leer los *tokens* de entrada previamente convertidos en vectores de incrustación (*embedding*) y poder representar sus características semánticas.

El *decoder* contiene un mecanismo que combina la información interna del *decoder* con la información proveniente del *encoder*. La función mapea una consulta y un conjunto clave-valor a una salida, donde la salida es la suma ponderada de los valores, con los pesos determinados por una función de compatibilidad entre la consulta y las claves correspondientes. Así, cada *token* generado integra el historial parcial de salida con la información del input. De esta forma, el resultado, es un proceso donde cada nuevo *token* se produce a partir de la información previa generada y la representación del *input* proveniente del *encoder*. Este diseño, paso a paso, es especialmente eficaz en tareas secuencia-a-secuencia como traducción o resumen.

Otro aspecto importante del proceso de los LLMs es la *tokenización*. Un *token* es la unidad mínima de texto que el modelo procesa, ya sea una palabra completa o un segmento subpalabra resultante de dividir una palabra en partes más pequeñas. El proceso de *tokenización* convierte el texto en una serie de *tokens* con un identificador único (*token ID*). El resultado es una secuencia numérica que representa el texto original en un formato manejable para el modelo. La *tokenización* es un proceso clave para el rendimiento: la eficiencia depende en gran medida del número de *tokens* en que se divide un texto. Dada una sucesión de m *tokens*, x_1, \dots, x_m , se busca generar los siguientes n *tokens* para obtener la sucesión $x_1, \dots, x_m, \dots, x_{m+n}$. Holtzman, Buys, et al. en su trabajo [7] consideran que los modelos calculan la probabilidad

$$P(\{x_i\}_{i=1}^{m+n}) = \prod_{i=1}^{m+n} P(x_i | x_1, \dots, x_{i-1}),$$

para generar la continuación *token a token* mediante una estrategia de decodificación determinada, como por ejemplo la *decodificación basada en maximización*, siendo una de las más usadas.

Para un mayor recorrido histórico acerca de la IA y los LLMs, puede verse el trabajo de D.H. Rai [17]. En la Figura 2.2 se presenta una línea del tiempo con los hitos más relevantes de los LLMs.

se realizó un experimento donde el programa *Racter* generó *The Policeman's Beard Is Half Constructed* (1984) [16], que consistía en fragmentos de prosa. El texto finalmente fue editado y supervisado por el humano. No fue hasta 2017 cuando Ross Goodwin condujo desde Nueva York hasta Nueva Orleans con una IA conectada a sensores instalados en el vehículo, cuyos datos convirtió en palabras. La novela *1 the road* (2018) [3] fue publicada un año mas tarde en 2018 sin ninguna edición por parte del humano. En estos últimos años se han publicado más novelas escritas por IA y pulidas por el autor, siendo uno de los primeros casos *Iris* (2023) [6] escrita por David Guisado con la ayuda de ChatGPT.

A pesar de la mejora de los modelos y de la consistencia y coherencia a la hora de generar texto narrativo, seguía habiendo desafíos al crear tramas extensas. Estos modelos tienden a divagar, olvidar detalles mencionados o introducir incoherencias en la historia a medida que se alarga el relato. Para poder abordar estos problemas, la investigación reciente se centra en integrar técnicas de planificación narrativa con modelos generativos. Se propusieron enfoques como el de “*Plan-and-Write*” mediante un modelo jerárquico [25]. En este método, dado un tema, primero se extrae una serie de eventos clave que surgen a lo largo de la historia, para que luego este esquema sirva de guía para producir el texto final. De esta forma se intenta que el modelo conserve el “plan” hacia donde transcurre la historia mientras la redacta y así tratar de evitar fallos de consistencia o esos “olvidos” producidos cuanto más larga es la historia. Los experimentos mostraron que incluir esta fase de planificación previa mejora notablemente la fidelidad, coherencia e interés de las historias obtenidas, en comparación con generar texto de una sola pasada sin un plan previo.

Para abordar estos problemas, este proyecto propone un sistema de generación de historias guiado por secciones. Previamente el usuario define las secciones con ayuda del modelo, pudiendo editar y personalizar cada una de ellas. Una vez definidas estas secciones, el modelo genera los capítulos de la historia haciendo uso de la información almacenada en la base de datos de cada sección. De esta forma el modelo siempre tiene acceso a la información relevante de la historia como la trama, personajes o escenarios, evitando así los problemas de coherencia y consistencia que suelen surgir en relatos largos. Este proyecto ha dado lugar a la aplicación que denomino **historIAs**.

Descripción del Trabajo

3.1. Herramientas utilizadas

En este proyecto se han empleado tres herramientas principales que han sido determinantes para el desarrollo de la aplicación: **Groq**, **Flet** y **MongoDB**

- **Groq**: Es una empresa tecnológica desarrolladora de hardware específico para la inferencia de la inteligencia artificial [5]. **Groq** cuenta con el desarrollo de los procesadores **Large Processing Units (LPUs)**, que se caracterizan por ser altamente eficientes y estar optimizados para las tareas de inferencia en tiempo real. Las **LPUs** permiten ejecutar LLMs y otros modelos a mayores velocidades y hasta diez veces más eficiente energéticamente en comparación con las GPUs. En un principio se planteaba la idea de utilizar un LLM de forma local pero enseguida comprobamos que no teníamos el hardware necesario para utilizarlo. Tras utilizar plataformas como **Google Colab** [4] y ver que no era viable para proyectos más grandes, se optó por hacer uso de un servidor para correr el modelo. Inicialmente se barajaron alternativas en **Microsoft Azure** [10] o **Amazon Web Service** [1] pero finalmente optamos por **Groq**, un servidor menos conocido pero que ofrece una mayor velocidad y facilidad para el uso. Únicamente se precisa de una cuenta de usuario y una **API Key** para poder utilizar los modelos que ofrece en la versión *Free*. A continuación, en la Tabla 3.1 se muestran los modelos que se han ensayado durante el proyecto. Nos hemos centrado en los modelos de **Llama** ya que son los más descargados y mejor valorados por la comunidad según **Hugging Face** [8], siendo el modelo más descargado **meta-llama/Llama-3.1-8B-Instruct** con 15 millones de descargas a fecha de este TFG. Al momento de seleccionar el modelo, la parte más importante era los límites de uso que ofrecía cada uno. En nuestro caso (ver datos de la Tabla 3.1) *RPM* y *RPD* no son cruciales, al ser un proyecto pequeño se hace imposible superar ese número de peticiones por minuto y por día. Lo relevante son los valores *TPM* y *TPD*, en especial el número de

tokens por minuto por la demanda del “contexto” (ver la Sección 3.2.4 para más detalles) de la historia. A medida que se avanza en la generación de la historia se tiene que añadir más información al LLM, provocando en ocasiones, que con una sola petición se supere el límite de *tokens* de **Groq**. Por este motivo, los modelos con un *TPM* menor de 6K en numerosas ocasiones sobrepasaban el límite permitido por el servidor. Finalmente se optó por el modelo **llama3-70b-versatile** que posee el doble de límite de *tokens* por minuto permitiendo una mayor flexibilidad a la hora de generar la historia (ver tabla completa de los modelos en el Apéndice A).

Tabla 3.1: Límites de uso de los modelos empleados.

Modelo	RPM	RPD	TPM	TPD
deepseek-r1-distill-llama-70b	30	1K	6K	100K
llama-3.1-8b-instant	30	14.4K	6K	500K
llama3-70b-versatile	30	1K	12K	100K
llama3-8b-8192	30	14.4K	6K	500K

RPM: peticiones/min; **RPD:** peticiones/día; **TPM:** tokens/min;
TPD: tokens/día. Valores sujetos a cambios según cuenta/plan.

- **Flet:** Es un **framework** de código abierto basado en **Flutter**, lo que permite desarrollar aplicaciones web, de escritorio y móviles usando únicamente Python [2]. A diferencia de otras bibliotecas, **Flet** abstrae la complejidad del **frontend**, simplificando las complejidades y proporcionando componentes listos para usar sin necesidad de tener conocimientos previos en **frontend**.

En el proyecto se ha utilizado **Flet** para implementar la interfaz de usuario encargada de guiar al usuario en el proceso de creación de historias. La decisión de emplear esta herramienta se debe a que no se quería combinar Python con otros lenguajes adicionales como PHP, HTML o CSS para el desarrollo de la parte visual. En su lugar, se buscó una solución que permitiera mantener toda la aplicación en un único lenguaje. Tras analizar distintas alternativas, se optó por **Flet**, un **framework** relativamente reciente que facilita la construcción de interfaces mediante componentes ya definidos, lo que agiliza notablemente el desarrollo y asegura una integración directa con la lógica implementada en Python.

- **MongoDB:** En el proyecto se ha utilizado **MongoDB** como sistema de gestión de base de datos [11]. Se trata de una base de datos NoSQL orientada a documentos, en la que la información se almacena en formato BSON (Binary JSON). Cada base de datos posee distintas colecciones donde se almacenan los documentos con los datos a guardar. A diferencia de los sistemas relacionales, **MongoDB** no requiere esquemas fijos, lo que le otorga gran flexibilidad para manejar estructuras de datos. Un aspecto clave es que el desarrollador trabaja siempre con documentos en formato JSON. Cuando se inserta un documento, **MongoDB** se encarga automáticamente de convertirlo a BSON para almacenarlo

de manera eficiente. De igual modo, al recuperar la información, MongoDB transforma el BSON de vuelta a JSON para que pueda manipularse de forma sencilla en la aplicación.

La elección de esta herramienta se debe a la necesidad de almacenar entidades heterogéneas como personajes, tramas, escenarios o capítulos, cuya estructura podía variar según la historia. Con una base de datos relacional habría sido necesario definir tablas y relaciones rígidas, lo que dificultaría la evolución del modelo. En cambio, con MongoDB es posible guardar directamente documentos con la estructura que se necesite en cada momento, simplificando la persistencia de datos y facilitando tanto la consulta como la actualización durante el proceso de generación de historias.

3.2. Diseño de la aplicación historIAs

Este capítulo presenta el proceso de diseño de la aplicación, ofreciendo una visión global de cómo se estructura y comporta el sistema, así como el flujo de información interno.

3.2.1. Modos de generación de historia

La aplicación cuenta con dos modos principales de creación de historias, diseñados para adaptarse a los distintos estilos de trabajo y posibilidades:

- *Modo detallado*: Este modo se centra en tener una mayor personalización de la historia por parte del usuario. El proceso de creación se divide en varias secciones predeterminadas, tales como *trama e hilo simbólico*, *descripción del mundo*, *personajes principales*, *desarrollo de escenarios*, *estructura de capítulos* y la propia escritura de la historia. Una vez se genera la sección, el usuario a través de botones interactúa generando, revisando y modificando cada sección antes de continuar. Este enfoque permite un alto grado de personalización y control sobre el contenido, así como la posibilidad de reescribir o editar manualmente cualquier parte de la historia.
- *Modo rápido*: Orientado a obtener resultados inmediatos, este modo presenta al usuario un breve cuestionario con distintos campos opcionales. El modelo va generando los capítulos siguiendo las indicaciones proporcionadas o, en ausencia de ellas, elaborando los elementos de forma creativa. Este modo no permite realizar modificaciones sacrificando parte de la personalización en favor de la velocidad y simplicidad.

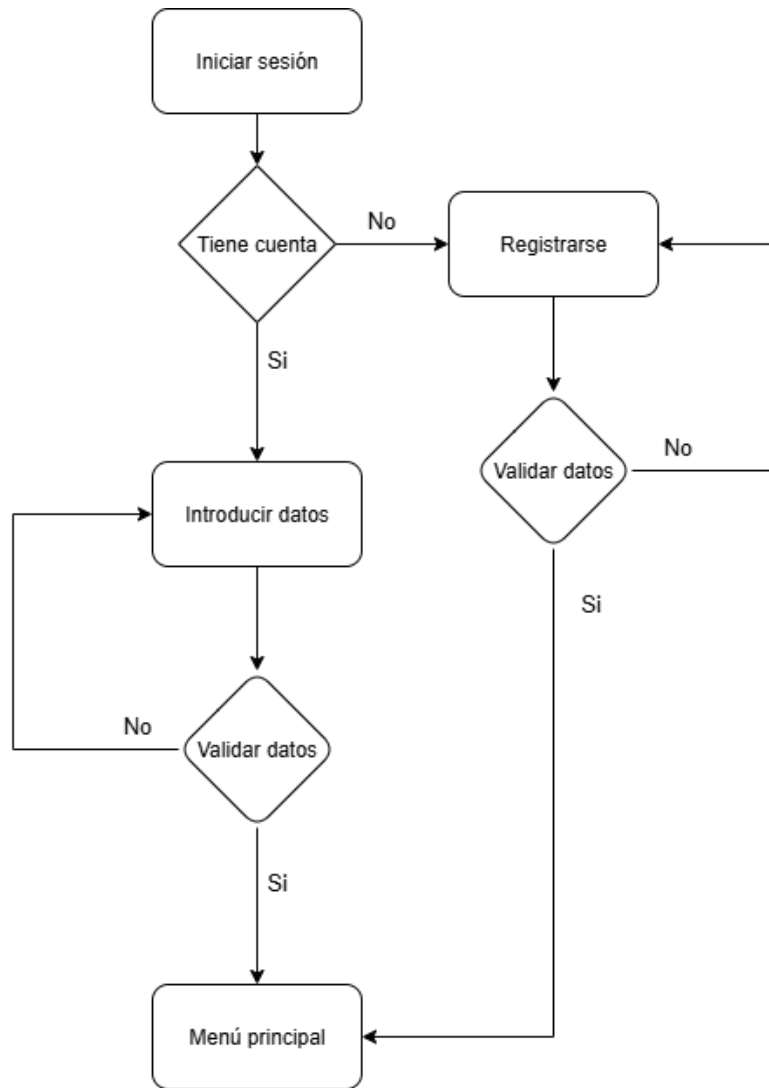
Con ambos modos se constituye un enfoque innovador respecto a otros LLMs como así se ha citado en la Sección 2. Esto permite una gran flexibilidad, aportando opcio-

nes para los usuarios que deseen un control exhaustivo del proceso creativo y para los usuarios que precisen resultados más rápidos. Internamente, cada modo emplea un conjunto de **prompts** específicos (ver Sección 3.2.6) adaptados a los requisitos del modo.

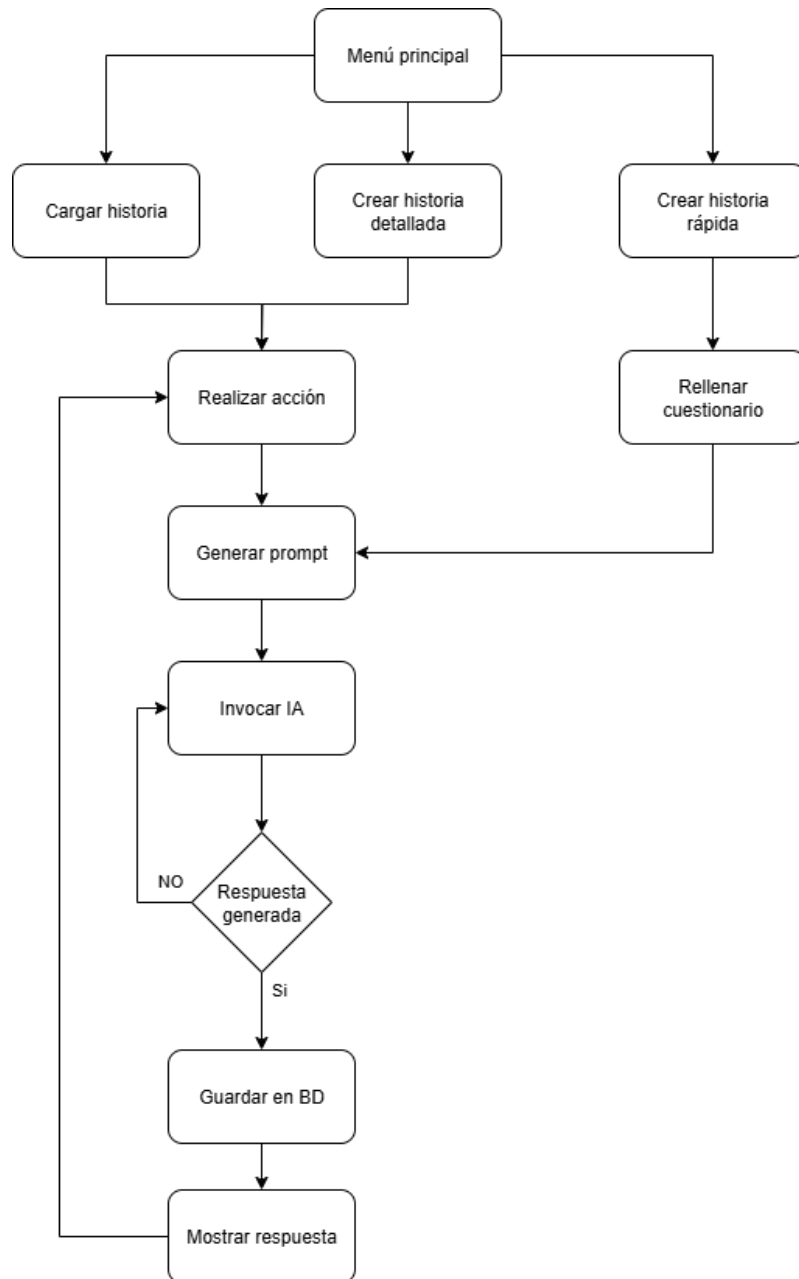
3.2.2. Diagramas de flujo

Para poder abordar con más detalle el diseño de la aplicación **historIAs**, haremos uso de distintos diagramas. Se ha dividido el diagrama de flujo para facilitar el detalle y la comprensión:

- *Inicio de sesión*: El usuario inicia sesión para acceder a la aplicación. Al iniciar la aplicación, el usuario se encuentra la pantalla de **Login** donde debe introducir su usuario y contraseña para poder acceder al menú principal. En caso de no disponer de una cuenta, posee la opción de registrarse creando una nueva. Se validan los datos introducidos, si son correctos, se accede al *Menú principal* (**Home**).

Figura 3.1: Diagrama de flujo *Inicio de sesión*

- *Flujo principal:* Una vez se accede al menú principal, el usuario puede realizar cuatro acciones principales: *eliminar historia*, *cargar historia*, *crear historia detallada* y *crear historia rápida*. En todo momento el usuario puede cargar una historia ya creada, permitiendo retomar el trabajo en cualquier momento. De igual forma, el usuario puede comenzar una nueva historia en cualquiera de ambos modos (ver Sección 3.2.1 para más detalles sobre los modos de creación de historias) manteniendo varias iniciadas al mismo tiempo.

Figura 3.2: Diagrama de flujo *Flujo principal*

3.2.3. Diagrama de componentes

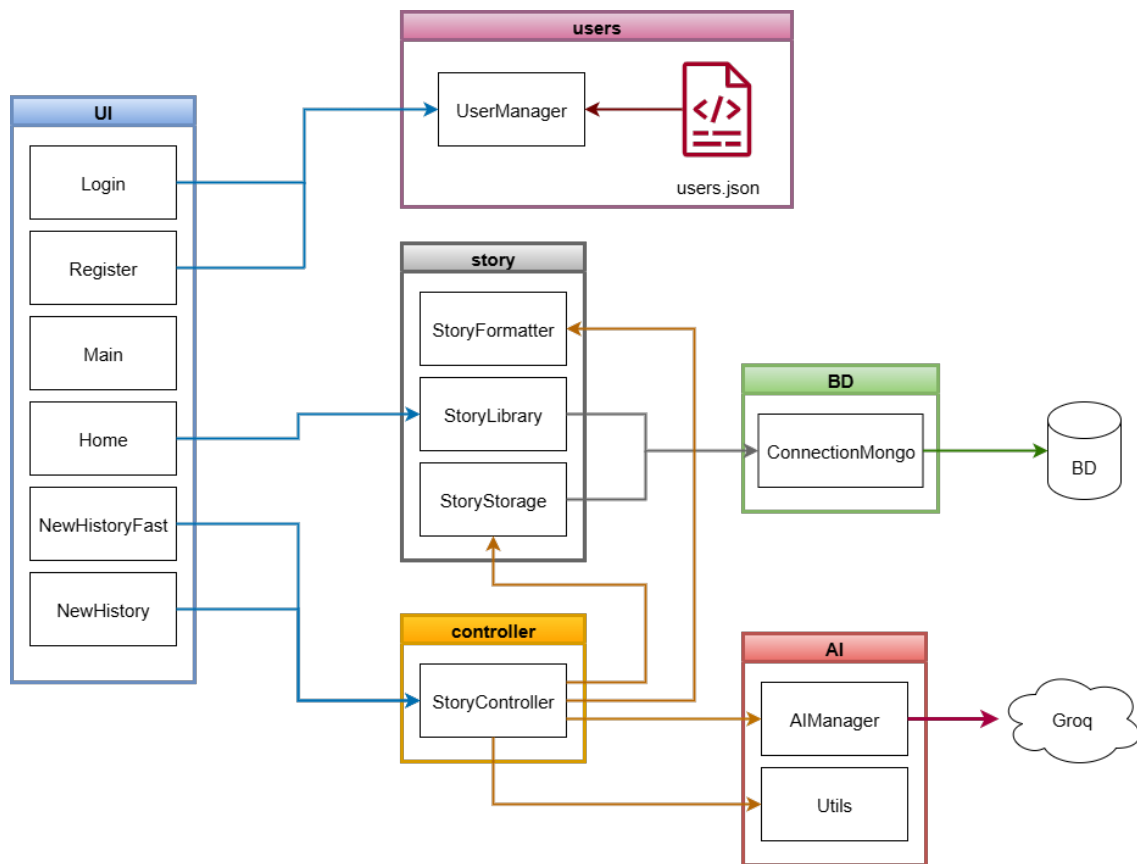


Figura 3.3: Diagrama de componentes

En este diagrama de la Figura 3.3 se observa como está estructurada la aplicación. Esta dividida en distintos paquetes:

- **UI (interfaz de usuario)**: Se encuentran las clases que implementan **Flet** para crear la interfaz gráfica. Tenemos el **Main** que actúa como punto de arranque de la aplicación en **Flet** y se encarga de definir las rutas asociadas a cada pantalla. El resto de clases corresponden a las diferentes pantallas de la aplicación *referencia a la seccion pantallas*.
- **users**: La clase **UserManager** se encarga de la gestión de usuarios a la hora de realizar el **login** y el registro. Contiene un fichero **users.json** donde se almacena el nombre, correo y contraseña de cada usuario.
- **story**: Esta sección agrupa toda la lógica que hace posible la creación, formateo y almacenamiento de las historias. **StoryFormatter** se encarga de todo lo relacionado con formatear el texto para convertirlo en un formato JSON que sea compatible con **MongoDB**. La clase **StoryLibrary** se encarga de gestionar las historias creadas por el usuario y **StoryStorage** crea todo lo relacionado

con la historia, como las secciones, resúmenes y almacenado en la base de datos.

- *controller*: Su función es separar la lógica de la interfaz y las llamadas al servidor **Groq**. Se ocupa de llevar la lógica del procesamiento del mensaje, detectar la sección en la que se encuentra el usuario, utilizar los métodos creados en **StoryStorage** y realizar las llamadas a **AIManager** que devuelven la respuesta generada por el modelo.
- *BD (base de datos)*: Contiene la clase **ConnectionMongo** encargada de realizar las consultas en la base de datos.
- *AI*: Contiene la clase **AIManager** encargada de establecer la conexión con el cliente **Groq** y realizar las llamadas al servidor. Además contiene un fichero **Utils** donde se almacenan los distintos **prompts** iniciales que se usarán para generar las respuestas.

3.2.4. Interacción con el modelo LLM

La interacción entre el LLM y el usuario es la parte principal de la funcionalidad de la aplicación. El sistema se apoya en la *API de Groq*, utilizando el modelo **llama-3.3-70b-versatile** [9], para generar el contenido narrativo. Este contenido se genera siguiendo una serie de pasos que se van a explicar en esta sección. Cada vez que el usuario realiza una acción desde **NewHistory**, esta clase crea un *thread* para separar la parte de la interfaz de la petición al servidor y así evitar bloqueos en la parte de la interfaz. Mientras este *thread* está activo, bloquea temporalmente la posibilidad al usuario de realizar nuevas acciones. Esto permite que el usuario no pueda generar múltiples mensajes simultáneamente mientras se genera una respuesta, evitando así colisiones y duplicación de peticiones al servidor. Además el *thread* se conecta con **StoryController** que se encarga de realizar el procesamiento del mensaje introducido. El modo rápido de creación de historias funciona de igual forma, lo único que cambia es la manera en la que sea crea el mensaje.

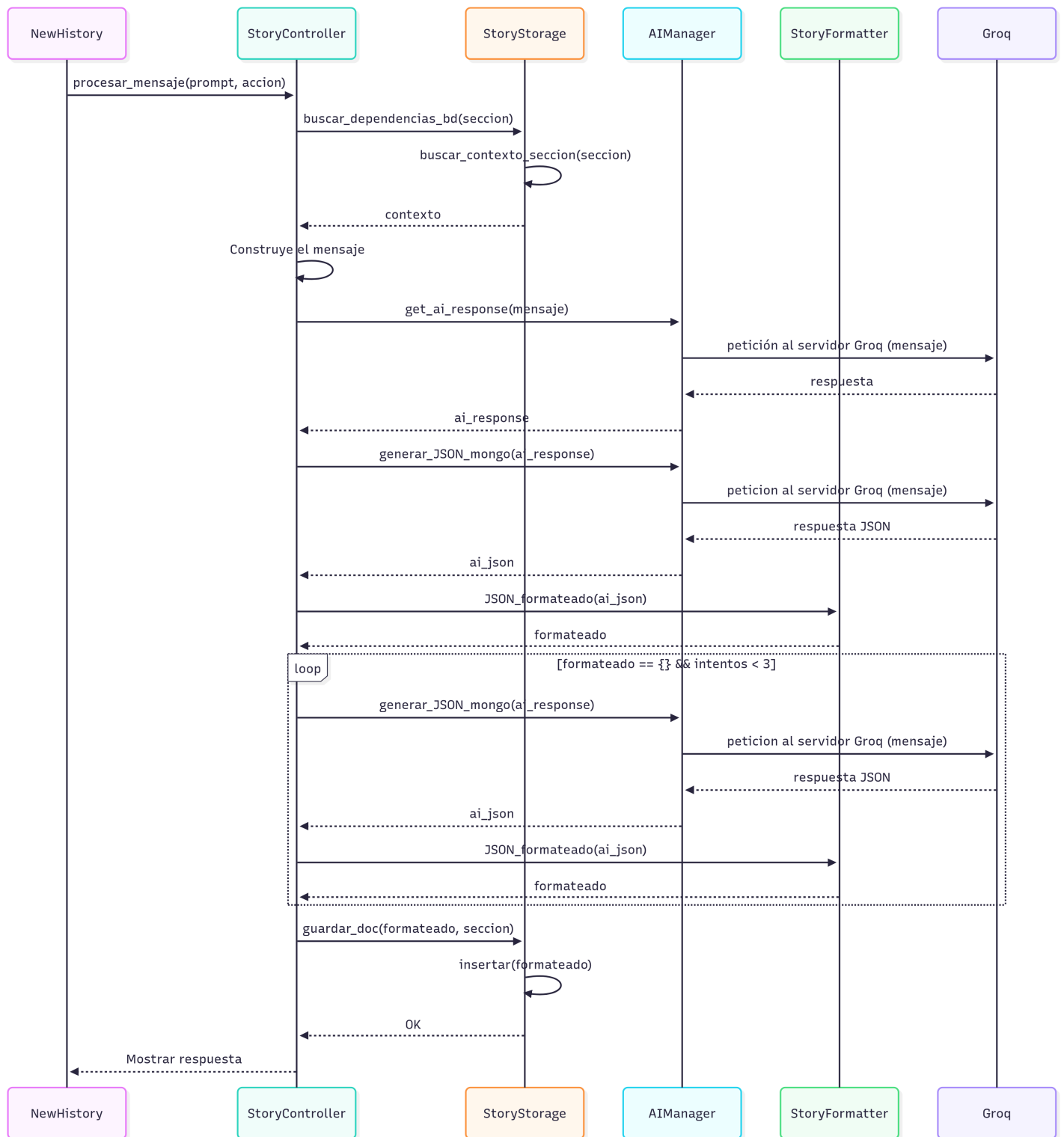


Figura 3.4: Diagrama de secuencia

Como se observa en la Figura 3.4, **StoryController** realiza la función de puente entre la parte de la interfaz y de la lógica. Esta clase se encarga de actualizar en todo

momento la sección en la que se encuentra el usuario. Con dicha sección llamamos a `buscar_dependencias_bd` de `StoryStorage` encargada de buscar, para la sección actual, qué otras secciones necesita el modelo para generar el `prompt`. Para cada sección se hace una búsqueda en la base de datos y se crea el contexto que se devuelve a `StoryController`. En el caso del modo rápido donde directamente se crean los capítulos, el contexto se forma con la encuesta y los capítulos anteriores. Ahora se construye el mensaje formado por el contexto creado anteriormente, la sección actual y el `input` del usuario. Con este mensaje se realiza una petición al servidor y genera una respuesta en formato texto. Para poder almacenar el contenido de la historia en `MongoDB`, se necesita convertir el texto a formato JSON. Durante el desarrollo nos encontramos con un problema fundamental: la respuesta generada por el LLM. Aunque en la mayoría de casos seguía las normas esperada, no siempre respetaba de forma exacta la misma estructura. Intentar depurar estas variaciones de forma manual haciendo uso, por ejemplo de `json.loads()` junto con expresiones regulares, resultó ser una tarea extremadamente compleja provocando una gran cantidad de errores ocasionados por un mínimo cambio inesperado en la respuesta. Para poder solucionar estas dificultades se opta por hacer uso nuevamente del LLM enviándole directamente la respuesta generada y pidiendo que la convierta en un objeto JSON válido. De este modo, el modelo actúa como “formateado” de su propio `output`, liberándonos de la necesidad de escribir código de `parsing` extremadamente específico. Al igual que antes, el modelo en ocasiones comete errores y no siempre genera un JSON válido, lo que provoca un fallo al guardar en la base de datos. Para solucionar estos errores, se implementa un bucle que permite al modelo generar un nuevo JSON en cada iteración. Además, se incorpora un contador para evitar que el proceso se prolongue indefinidamente. Tras varias pruebas, comprobamos que al limitar el bucle a tres iteraciones se consigue una tasa de acierto prácticamente perfecta. Con este nuevo mensaje, como se muestra en el Código 3.1, se llama a `get_ai_response` de `AI/AIManager.py` donde se realiza la llamada al cliente de `Groq`.

Código 3.1: Llamada al cliente de `Groq`

```
def get_ai_response(messages: str) -> str:
    """Obtiene la respuesta de Groq AI y la limpia antes de
        mostrarla."""
    # Agregar mensaje del sistema
    messages.insert(0, {
        "role": "system",
        "content": PROMPT_INICIAL
    })

    completion = client.chat.completions.create(
        model="deepseek-r1-distill-llama-70b",
        messages=messages,
        temperature=0.8,
        max_tokens=4096,
    )

    # Obtener la respuesta sin <think>...</think>
    raw_response = completion.choices[0].message.content
```

```
return clean_ai_response(raw_response)
```

Se añade al mensaje el *prompt inicial* que contiene la estructura y las reglas que debe seguir el modelo para generar la historia. Seguidamente se hace la llamada al modelo configurando varios parámetros que influyen directamente en el comportamiento de la generación:

- **temperature:** Este parámetro controla el nivel de creatividad en la generación del texto por parte del modelo de lenguaje (ver Código 3.1). Funciona ajustando la distribución de probabilidad con la que el modelo selecciona la siguiente palabra (*token*) en cada paso de la generación. Los valores varían entre 0 y 2 siendo 0 menos creatividad y más conservador y 2 más aleatoriedad. Normalmente se utilizan valores comprendidos entre 0 y 1, ya que, según se observa en el trabajo de Renze y Guven (2024) [18], la precisión de la respuesta se mantiene estable hasta el valor 1 donde empieza a decaer (ver Figura 3.5). En nuestro caso se ha optado por seleccionar 0,8 para conseguir una gran creatividad pero sin perder la precisión a la hora de generar la respuesta.

Caso de uso	Rango	Descripción y ejemplo
Generación de código	0,0 – 0,3	Alta precisión y coherencia. Ideal para scripts, funciones matemáticas y automatización.
Chatbots y asistentes	0,4 – 0,7	Equilibrio entre naturalidad y control. Útil para asistentes virtuales conversacionales.
Narrativa creativa	0,8 – 1,0	Estilo más libre y expresivo. Adecuado para cuentos, diálogos y contenido literario.
Exploración artística	>1,0	Generación altamente impredecible y experimental. Usado en arte generativo o literatura abstracta.

Tabla 3.2: Ajuste de **temperature** según el tipo de tarea

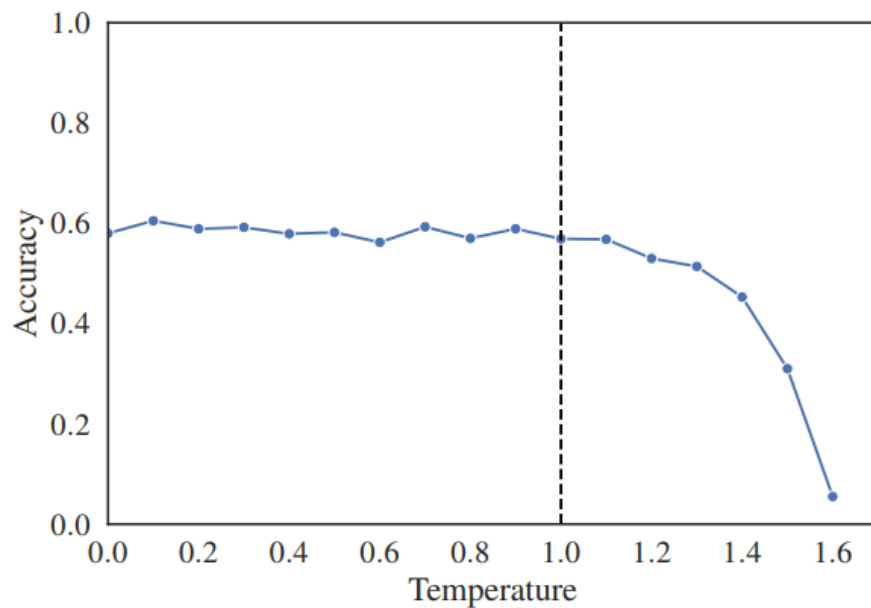


Figura 3.5: Precisión según la temperatura (de 0,0 a 1,6) para GPT-3.5 usando el prompt CoT en el examen de 100 preguntas (fuente del gráfico [18]).

- **max_tokens**: Define el número máximo de *tokens* que puede devolver el modelo de lenguaje en una respuesta. Se ha establecido este tope en 4.096 *tokens* lo cual permite una salida completa del modelo evitando cortes a mitad de la generación. Esta cantidad debe convivir con el *prompt del sistema* y el mensaje generado, por eso se debe ajustar cuidadosamente el máximo de *tokens* por respuesta para no desbordar el tope.

Ya con el mensaje generado, se guarda en la base de datos y se muestra la respuesta al usuario.

3.2.5. Almacenamiento en la base de datos

Para este proyecto únicamente se necesita guardar el texto generado en cada sección, por eso se ha decidido usar una base de datos no relacionada, facilitando su uso frente a una base de datos SQL. Cada sección de la historia se guarda en su propia colección, lo que facilita las consultas. Se crea una base de datos **info_usuarios** (ver Figura 3.6) donde se almacenan documentos con la información más importante de cada historia: usuario, nombre de la historia, modo y estado de las secciones. Esto ayuda a la hora de mostrar al usuario las historias que ha creado y pueda retomarla desde donde la dejó.

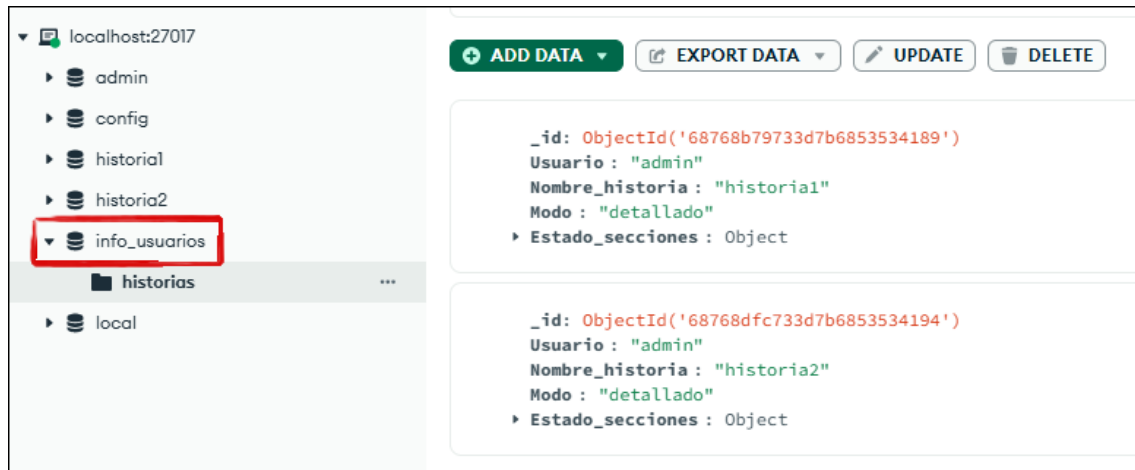


Figura 3.6: Ejemplo de almacenamiento en la base de datos: *info_usuarios*

Dentro de la base de datos de cada historia se guardan las colecciones de cada una de las secciones (ver Figura 3.7) y además, dos colecciones especiales como son: **info** y **contexto**. La colección **info** almacena la información de usuario, modo y estado de las secciones de la historia. Mientras que **contexto** (ver Figura 3.8) guarda el contenido de cada una de las secciones en formato de texto plano para facilitar su uso en la construcción del mensaje que se le pasa al modelo.



Figura 3.7: Ejemplo almacenamiento en la base de datos para el *modo detallado: sección*

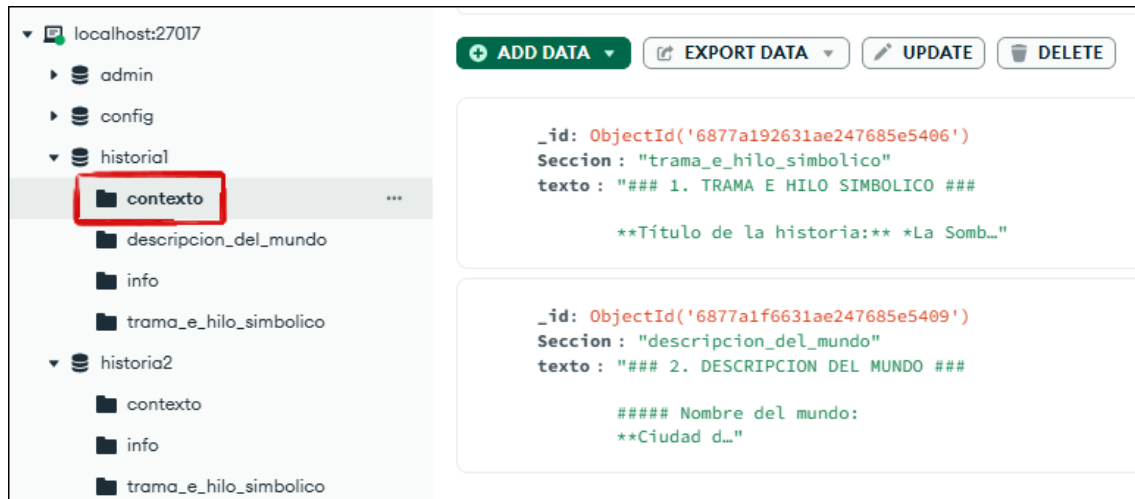


Figura 3.8: Ejemplo de almacenamiento en la base de datos para el *modo detallado*: *contexto*

La base de datos del *modo rápido* es más sencilla al tener que guardar únicamente los capítulos en una colección llamada **capítulos** (ver Figura 3.9). Mientras que el *modo detallado* requiere un número mayor de colecciones haciéndolo más complejo.



Figura 3.9: Ejemplo de almacenamiento en la base de datos para el *modo rápido*: *capítulos*

3.2.6. Gestión de prompts

Uno de los aspectos clave en el proyecto es la creación de los *prompt de sistema* para que el modelo pueda seguir una serie de pautas y actúe de determinada manera. En estos *prompts* se explicitan las reglas e instrucciones que el modelo debe seguir a la hora de generar una respuesta. Es importante comentar que aunque se concreten unas reglas y formato deseados, en ocasiones no se ajusta a las reglas preestablecidas. A continuación se explican los *prompts* más importantes:

- **PROMPT_INICIAL**: Se trata del **prompt** para el modo detallado donde se especifican las reglas que tiene que seguir. A medida que se realizaban pruebas y

se añadían nuevas funcionalidades este **prompt** también se iba actualizando. La parte más importante es la especificación de las secciones: *trama e hilo simbólico*, *descripción del mundo*, *personajes principales*, *descripción de escenarios* y *estructura de capítulos*. Cada una de las secciones contiene unas reglas determinadas y se proporciona un ejemplo de salida para lograr una mayor precisión en la respuesta. Tanto las secciones de personajes como escenarios, se aporta una plantilla con distintos campos para así facilitar la creación y guardado en la base de datos.

- **PROMPT_JSON**: Se invoca tras la generación del contenido narrativo del LLM para transformar el texto en formato JSON válido, listo para insertarlo en la base de datos de MongoDB. Se describe una serie de reglas para poder conseguir el formato JSON deseado donde el nombre de la sección se guarda en un campo *Sección* o secciones como personajes y escenarios se construyen a partir de listas. Con este **prompt** evitamos errores y permite controlar los pequeños cambios de formato que pueda producir el modelo al generar la respuesta.
- **PROMPT_ESCRITURA_RAPIDA**: Este es el **prompt** utilizado para el modo rápido. Al no tener las secciones y directamente generar los capítulos, se necesitan otras normas e instrucciones a la hora de construir la respuesta. Se especifica la importancia de utilizar los datos introducidos en la encuesta inicial para crear la historia.
- **PROMPT_CAPITULOS**: Tras desarrollar todas las secciones y pasar al momento de la escritura, se ha decidido crear un nuevo *prompt* más detallado para escribir los capítulos. De esta forma lo separamos del **prompt** donde se especifican las secciones logrando así, una mayor precisión. De igual forma, sirve para gestionar mejor la limitación de *tokens* al separar el **prompt** y no usar uno de mayor longitud.

3.3. Interfaz de usuario

Esta sección describe la aplicación desde el punto de vista del usuario final. El objetivo es que cualquier usuario, tenga o no base técnica, entienda cómo se crea una historia de principio a fin usando la herramienta, y por qué la interfaz está organizada del modo en que lo está.

3.3.1. Pantallas y funcionalidades

En este apartado se describen las vistas principales de la aplicación, explicando las funcionalidades que puede realizar el usuario en cada una de ellas así como mostrando capturas de pantalla de la aplicación (Figuras de *login* 3.10, de *registro* 3.11, de *Mis historIAS* 3.12, del *modo detallado* 3.13 y del *modo rápido* 3.14).

3.3.1.1. Inicio de sesión

Pantalla de inicio donde se solicita al usuario iniciar sesión a través de su *usuario* y *contraseña* para poder acceder a la aplicación. En caso de no estar registrado se puede crear una nueva cuenta mediante el botón **Register**.



Figura 3.10: Pantalla de inicio de sesión

3.3.1.2. Registro de usuario

En esta vista se solicitan los datos necesarios para crear una nueva cuenta: Username, E-mail, Password y Repeat Password. Una vez validados los datos, el sistema registra el usuario y lo dirige automáticamente al menú principal.

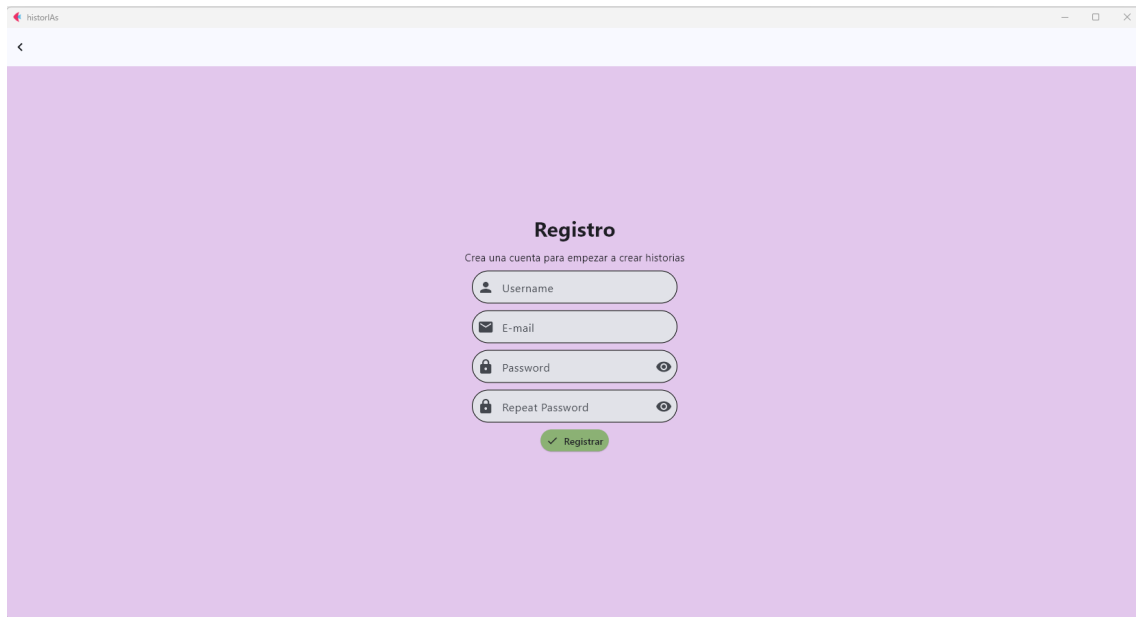


Figura 3.11: Pantalla de registro de usuario

3.3.1.3. Página principal

Esta pantalla muestra el menú principal estructurado en dos bloques principales:

- *Cargar y eliminar:* Cada usuario cuenta con el área *Mis historIAS* donde se visualiza su historial de historias junto con el progreso alcanzado. Para cada una de las novelas, el usuario puede retomarla desde el punto en que la dejó o eliminarla si no desea conservarla.
- *Empezar nueva historia:* Existe la opción de iniciar una nueva historia ya sea en *modo detallado* o *modo rápido*. En ambos casos se solicita al usuario introducir un nombre para guardar la historia en el historial.

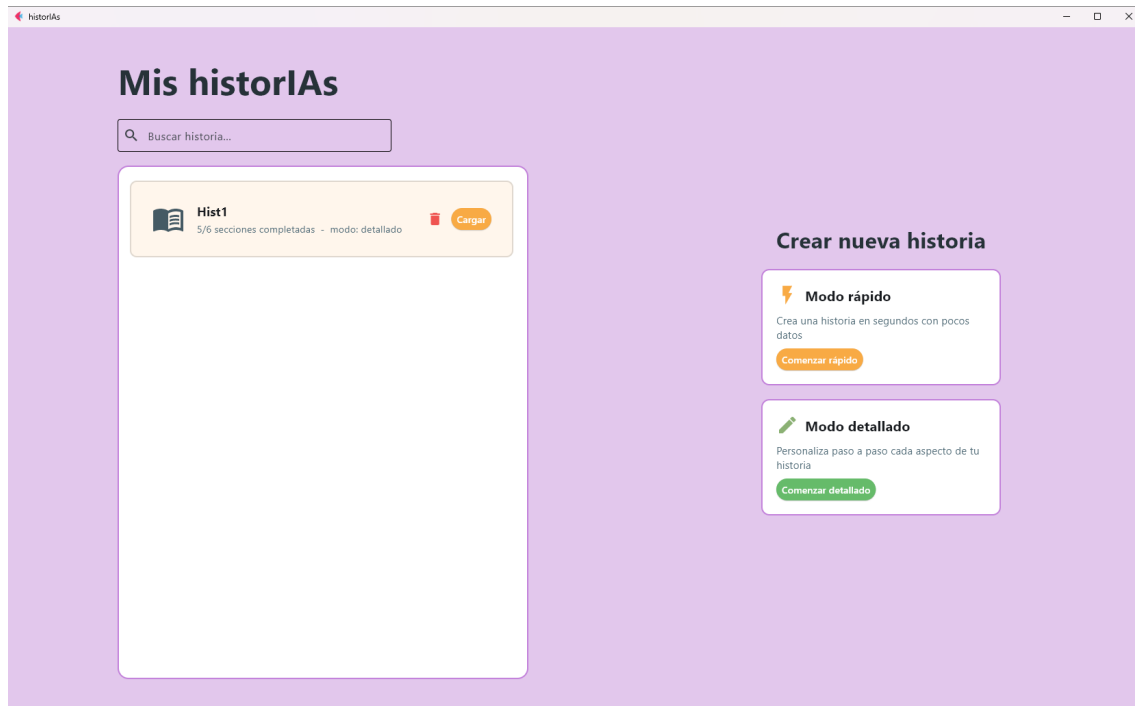


Figura 3.12: Pantalla principal

3.3.1.4. Comenzar historia detallada

Se trata de la pantalla principal del proyecto, dividida en tres áreas:

- *Lienzo narrativo*: Área principal donde se visualiza el contenido de la historia generado por el modelo. Al acceder por primera vez, se muestra un bloque de escritura en el que el usuario puede introducir una idea inicial o la temática de la historia. Desde este espacio también es posible emplear la opción *Reescribir* para volver a generar una sección completa, o *Modificar* para realizar cambios específicos en fragmentos concretos del texto.
- *Panel de secciones*: Zona de interacción del usuario dividida en dos partes: (1) *Ver*, permite en cualquier momento visualizar la sección deseada y, mediante la acción *Editar*, realizar modificaciones de forma manual. (2) *Generar*, posibilita seleccionar la sección que se desea crear, con la opción de especificar alguna características concretas para orientar la generación.
- *Instrucciones*: Espacio donde se presenta una breve guía para facilitar el uso de la aplicación

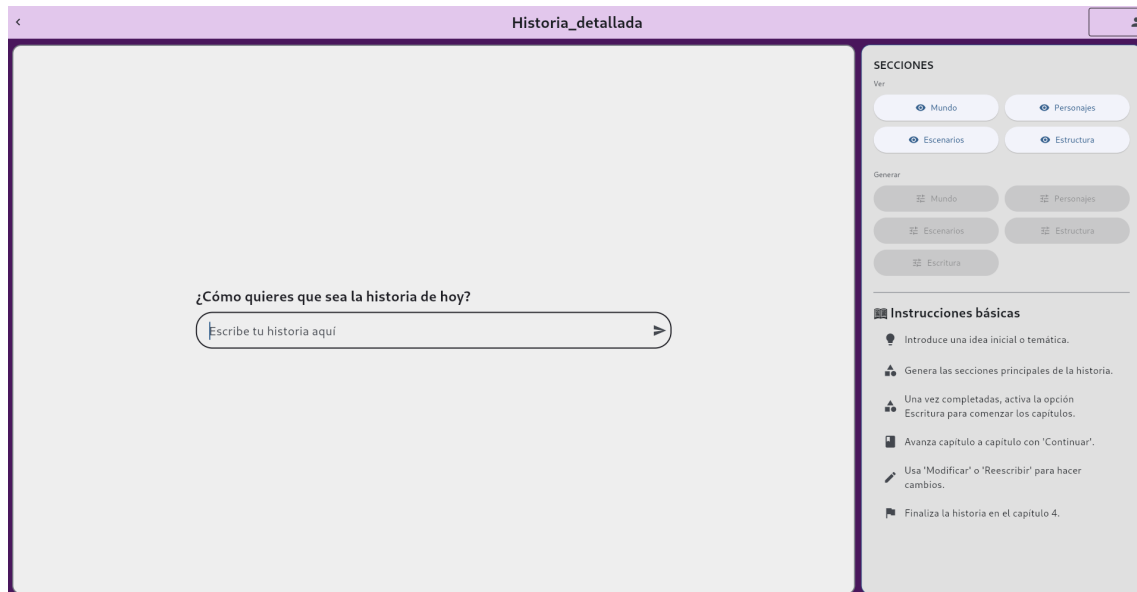


Figura 3.13: Pantalla de creación de historia en modo detallado

3.3.1.5. Empezar historia rápida

En este modo, la creación de la historia se desarrolla en dos fases principales:

- *Cuestionario inicial:* Al acceder a esta pantalla, el usuario se encuentra con un breve formulario donde puede introducir datos opcionales como la temática, personajes, ambientación o estilo narrativo. Una vez enviado el cuestionario, el sistema crea el primer capítulo.
- *Visualización y edición de capítulos:* Tras la generación, el capítulo se muestra en el área principal. Para cada capítulo el usuario cuenta con las acciones de *Reescribir*, *Modificar* y *Continuar* con el siguiente capítulo.

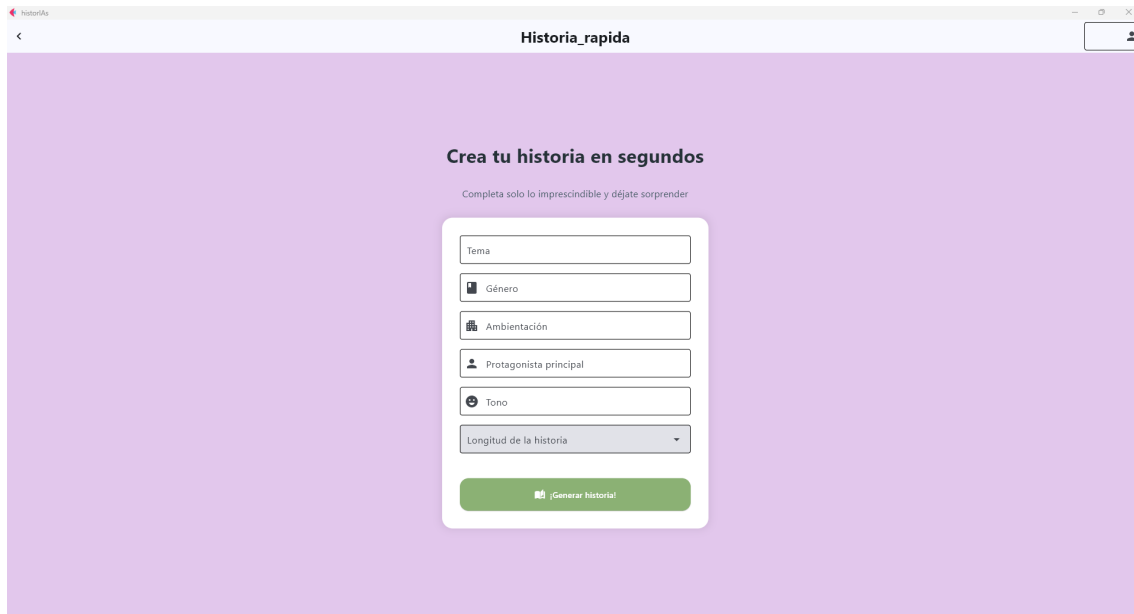


Figura 3.14: Pantalla de creación de historia en modo rápido

3.3.2. Arquitectura de la interfaz

Durante este apartado se explica el funcionamiento de **Flet** a la hora de crear y navegar por las distintas pantallas de la aplicación.

3.3.2.1. Modelo de UI en Flet

Las interfaces de usuario en **Flet** se construyen a partir de **controles** (también llamados **widgets**), que pueden organizarse en forma de árbol, donde el nodo raíz es **Page**. Cada pantalla de la aplicación se modela como una **View** asociada a una **ruta**, y la navegación consiste en cambiar la ruta activa para que **Flet** renderice la vista correspondiente. En este proyecto se utiliza la librería **flet_route** para gestionar el enrutamiento y la navegación.

3.3.2.2. Navegacion con **flet_route**

La ruta de la página se encuentra a continuación del símbolo **#** en la URL, por ejemplo:

```
https://localhost/#/nueva_hisotria
```

Todas las rutas comienzan con **/**, por ejemplo **/home**, **/nueva_historia**. Por defecto **Flet** comienza en la ruta: **http://localhost:PORT/#/** y cada vez que el usuario

navega entre páginas, Flet llama a `page.on_route_change`. En nuestro caso, al usar `flet_route`, se simplifica y únicamente usamos `page.go(nueva_historia)`.

Al construir varias vistas, `Page` deja de ser una única página y se convierte en un contenedor de vistas en capas una encima de otra como una pila (ver Figura 3.15).

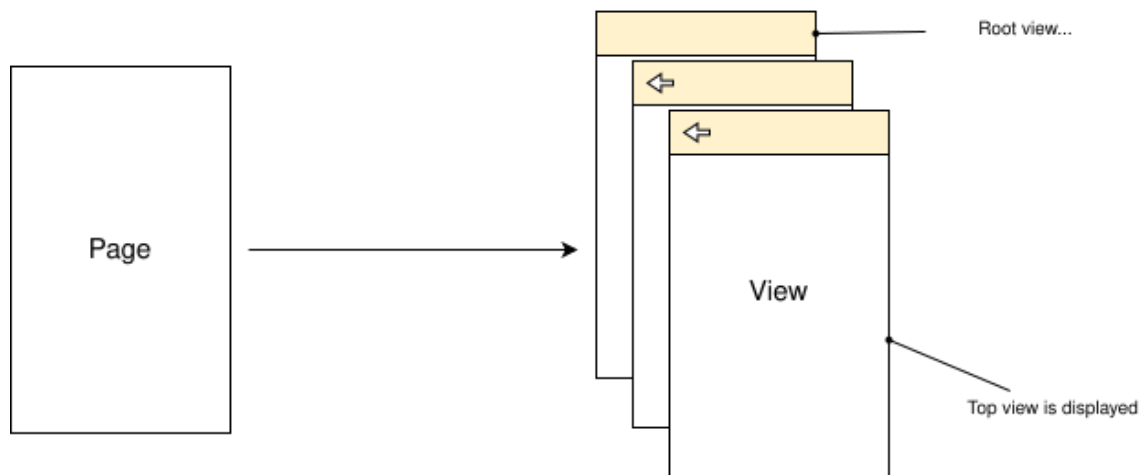


Figura 3.15: Organización de las pantallas en Flet

Se organizan como una lista de vistas donde la última de la lista es la que se muestra actualmente. En el siguiente fragmento de código se muestra la lista y organización de las vistas en el proyecto:

Código 3.2: Llamada al cliente de Groq

```
import flet as ft
from flet_route import Routing, path

#Importamos las pantallas
from UI.Home import Home
from UI.Login import Login
from UI.Register import Register
from UI.NewHistory import NuevaHistoria
from UI.NewHistoryFast import NuevaHistoriaRapida
from UI.LoadHistory import CargarHistoria

def main(page:ft.Page):
    page.title="Creador de historias"
    app_routes = [
        path(url="/", clear=True, view=Login().view),
        path(url="/register", clear=True, view=Register().
            view),
        path(url="/home", clear=True, view=Home().view),
        path(url="/nueva_historia", clear=True, view=
            NuevaHistoria().view),
        path(url="/nueva_historia_rapida", clear=True, view=
            NuevaHistoriaRapida().view),
```

```
        path(url="/cargar_historia", clear=True, view=
            CargarHistoria().view)
    ]

    Routing(page=page, app_routes=app_routes)
    page.go(page.route)

ft.app(target=main)
```

En resumen, la arquitectura de la interfaz combina la flexibilidad de `Flet` con la simplicidad de `flet_route`, lo que permite organizar la aplicación en vistas independientes y navegables de forma intuitiva. De esta manera, cada pantalla queda claramente separada y el usuario puede recorrer el flujo completo de creación de historias, desde el inicio de sesión hasta la edición y generación de la historia, sin percibir la complejidad técnica que hay detrás.

Capítulo 4

Desarrollo y evaluación de historias generadas

En este capítulo se expone un ejemplo de creación de una historia en el *modo detallado*. Se da un enfoque visual a través de capturas de pantalla para mostrar el resultado generado y se comentan aspectos importantes a tener en cuenta durante la creación de historias.

4.1. Ejemplo de una historia en el modo detallado

Una vez se comienza una nueva historia, se dispone de un cuadro de texto (ver Figura 3.13) donde se necesita que el usuario introduzca las características de su historia. Es válido desde una descripción mínima o nula (“*Quiero crear una historia*”, “*Genera una historia de ciencia ficción*”) a una descripción mucho más detallada como veremos en el ejemplo. Cuanto más se detalle más se ajusta la respuesta a la idea del usuario. Se envía este *prompt* y tras unos segundos de espera, el modelo nos muestra la *trama e hilo simbólico* que se trata de la primera de las secciones y única donde el usuario no puede elegir el orden. En la Figura 4.1 vemos la respuesta generada.

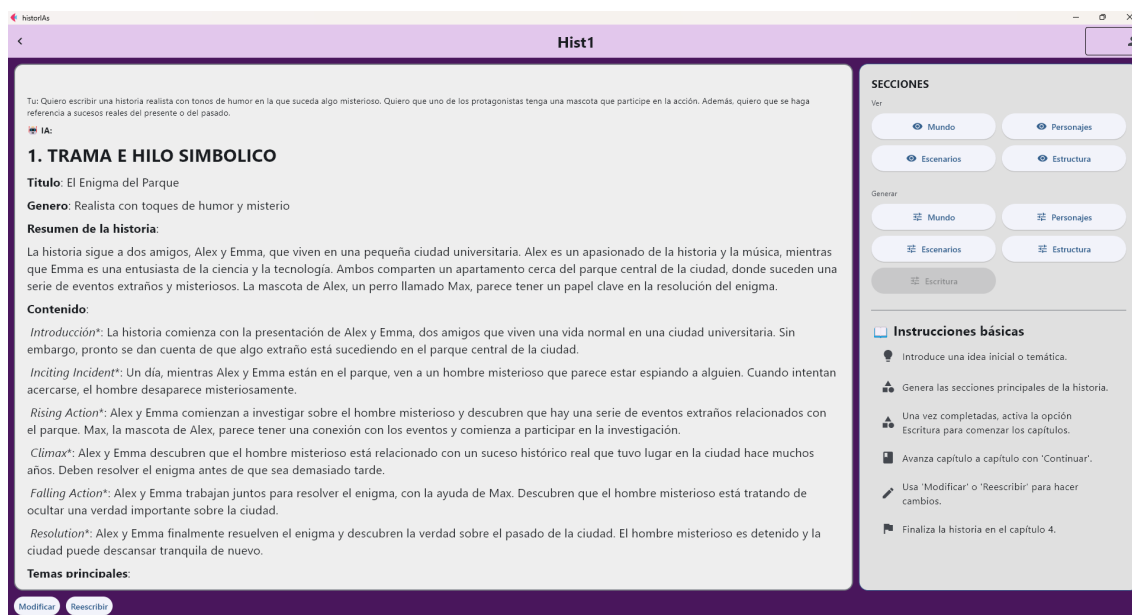


Figura 4.1: Historia de ejemplo: *Trama e hilo simbólico*. Mensaje introducido: “Quiero escribir una historia realista con tonos de humor en la que suceda algo misterioso. Quiero que uno de los protagonistas tenga una mascota que participe en la acción. Además, quiero que se haga referencia a sucesos reales del presente o del pasado.”

Como se puede observar en la Figura 4.1, el modelo introduce los aspectos que el usuario pidió. Como se ha comentado anteriormente, el modelo suele seguir una estructura en la respuesta aunque en ocasiones introduce variaciones. Una vez se muestra la primera sección, el resto de botones de las secciones se desbloquean para que sea el usuario el que elija la siguiente. El botón de *Escritura* da paso al desarrollo de los capítulos y se desbloquea cuando se completan el resto de secciones.

En este ejemplo se decide realizar una modificación mediante el botón *Modificación* y se habilita un cuadro de diálogo donde se introduce el cambio deseado (ver Figura 4.2).

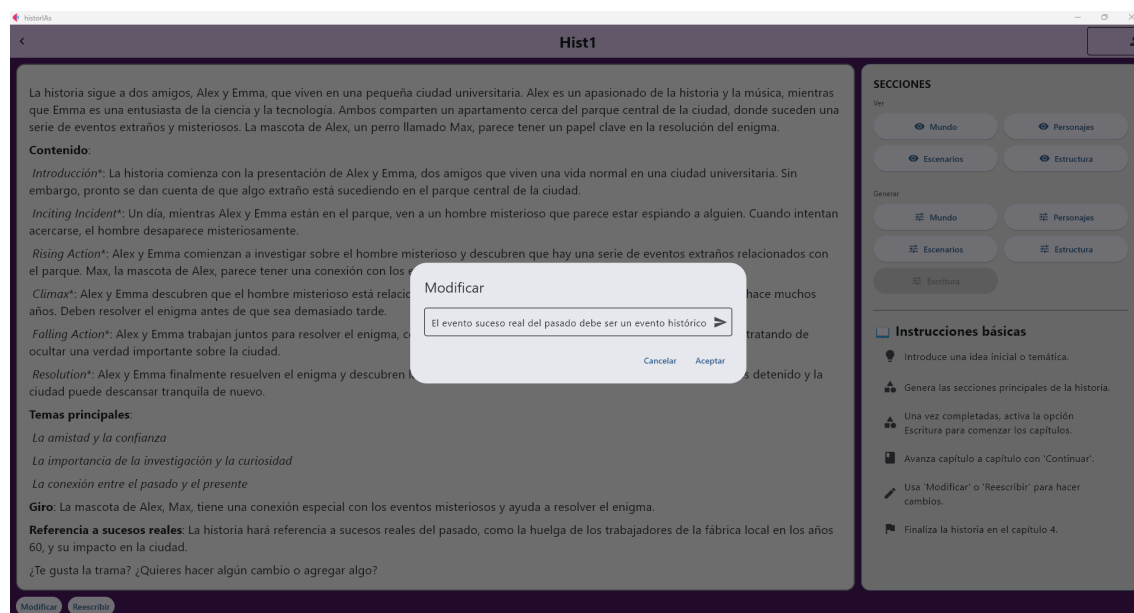


Figura 4.2: Historia de ejemplo: Modificar - *Trama e hilo simbólico*. Mensaje introducido: “El evento suceso real del pasado debe ser un evento histórico de la realidad.”

Tras esto el modelo vuelve a generar la sección con el cambio introducido (ver Figura 4.3).

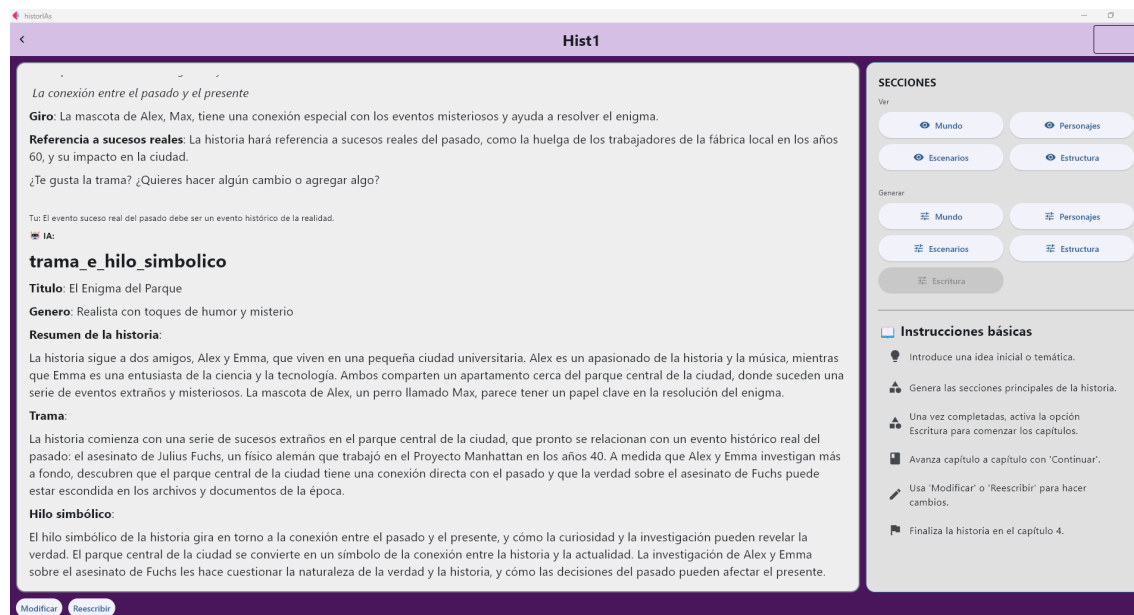


Figura 4.3: Historia de ejemplo: Modificación - *Trama e hilo simbólico*.

Se genera nuevamente la sección y comprobamos como si realiza el cambio pedido aunque acorta y reduce el contenido de la sección.

Cuando el usuario desee, elige la siguiente sección en la parte de *Generar* a la

derecha del recuadro principal. Como se muestra en la Figura 4.4, al seleccionar una sección se permite que el usuario introduzca una serie de características especiales para la sección, sin la necesidad de esperar a la respuesta y después modificar. En nuestro caso se pide una característica para los personajes.

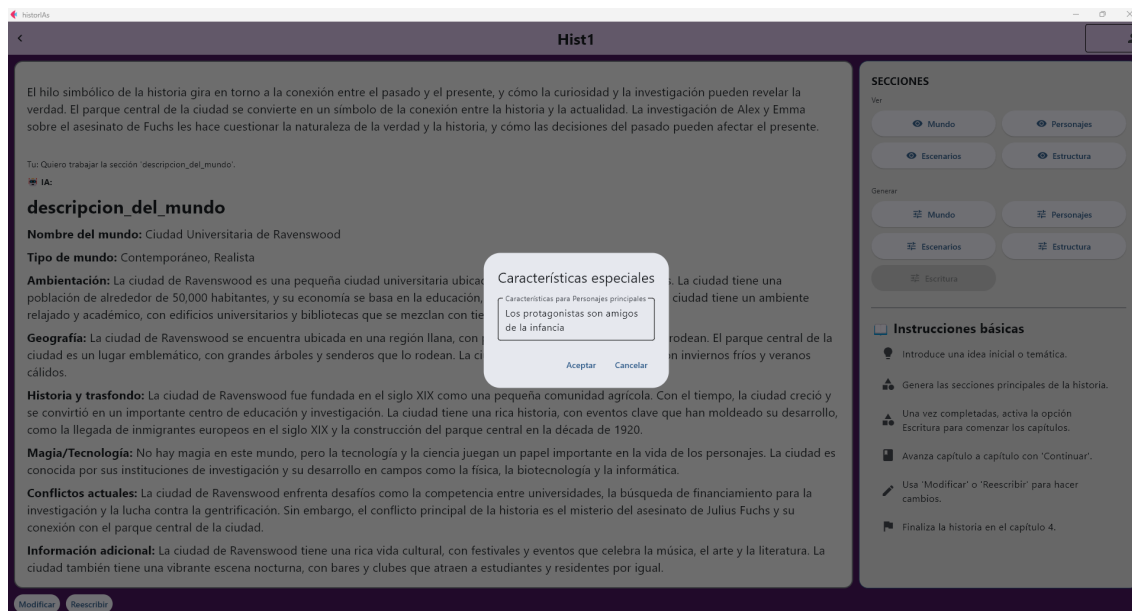


Figura 4.4: Historia de ejemplo: Características - *Personajes principales*. Mensaje introducido: “*Los protagonistas son amigos de la infancia.*”

De esta forma en la Figura 4.5 vemos como el modelo ha creado los personajes introduciendo las características del usuario.

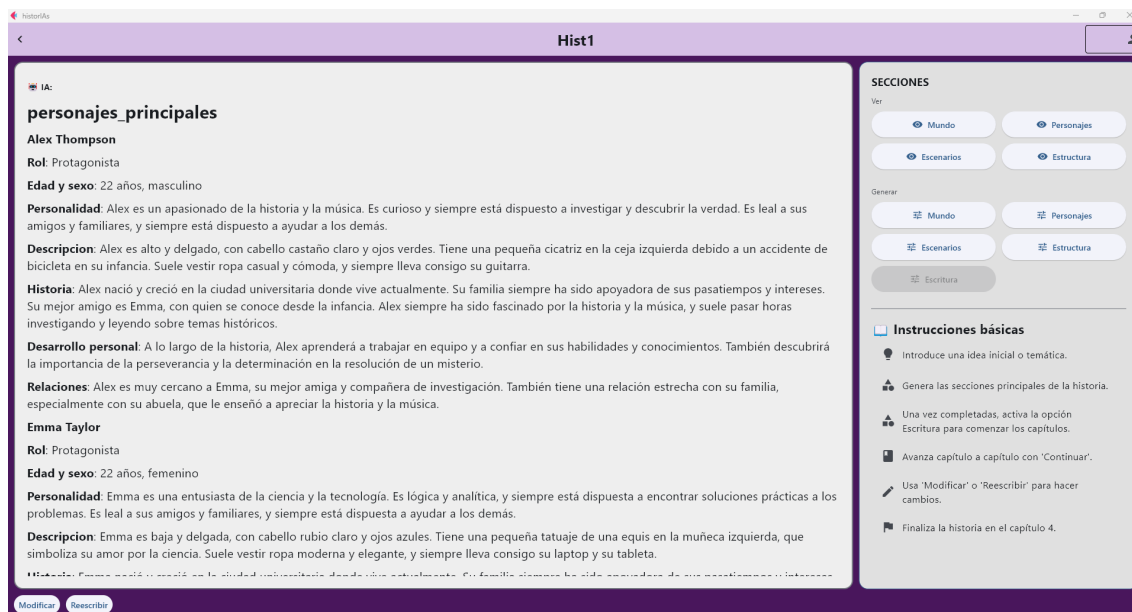


Figura 4.5: Historia de ejemplo: *Personajes principales*.

Se continua con el resto de secciones hasta que se completan todas y se permite el acceso a la parte de la escritura de capítulos mediante el botón *Escritura*. Debido a las limitaciones de *tokens* únicamente se permite realizar cuatro capítulos para no sobrepasar el límite. Como se muestra en la Figura 4.6 se eliminan los botones usados para generar las historias y se sustituye por el botón *Continuar* para avanzar los capítulos. De esta forma no se permiten realizar modificaciones ni volver a generar las secciones en mitad de los capítulos evitando futuros problemas.

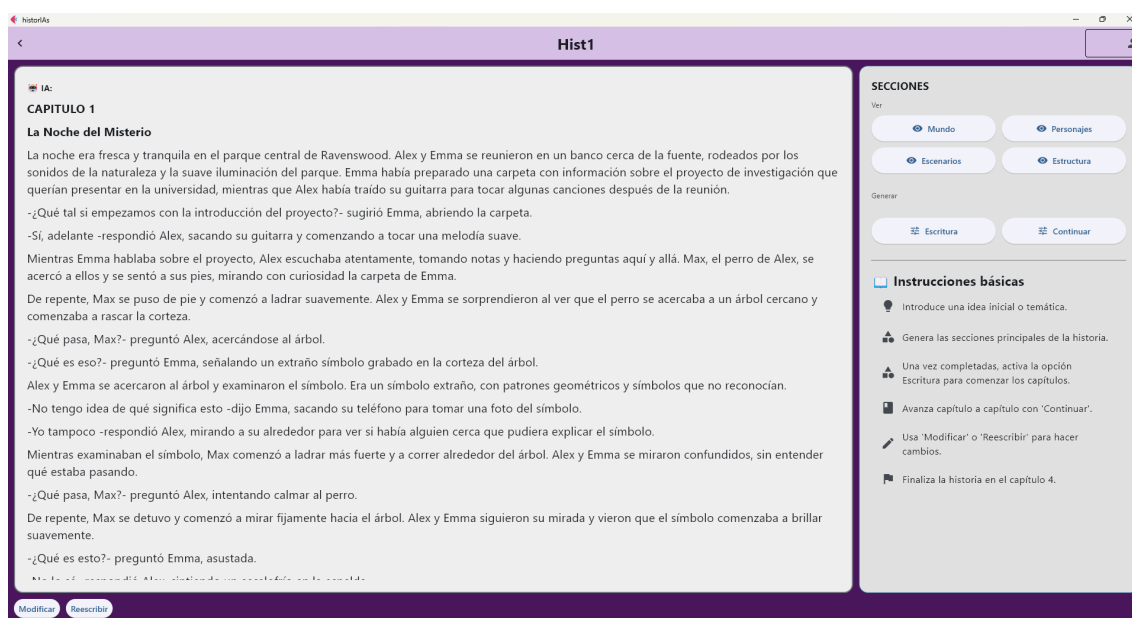


Figura 4.6: Historia de ejemplo: *Capítulo 1*.

Se continua con los siguientes capítulos hasta llegar al cuarto y último capítulo donde la historia concluye. Como se observa en la Figura 4.7 al ser el último capítulo, el botón *Continuar* se sustituye por el botón *Finalizar* permitiendo que el usuario guarde la historia generada en formato pdf y volviendo a la pantalla principal (ver Figura 3.12).

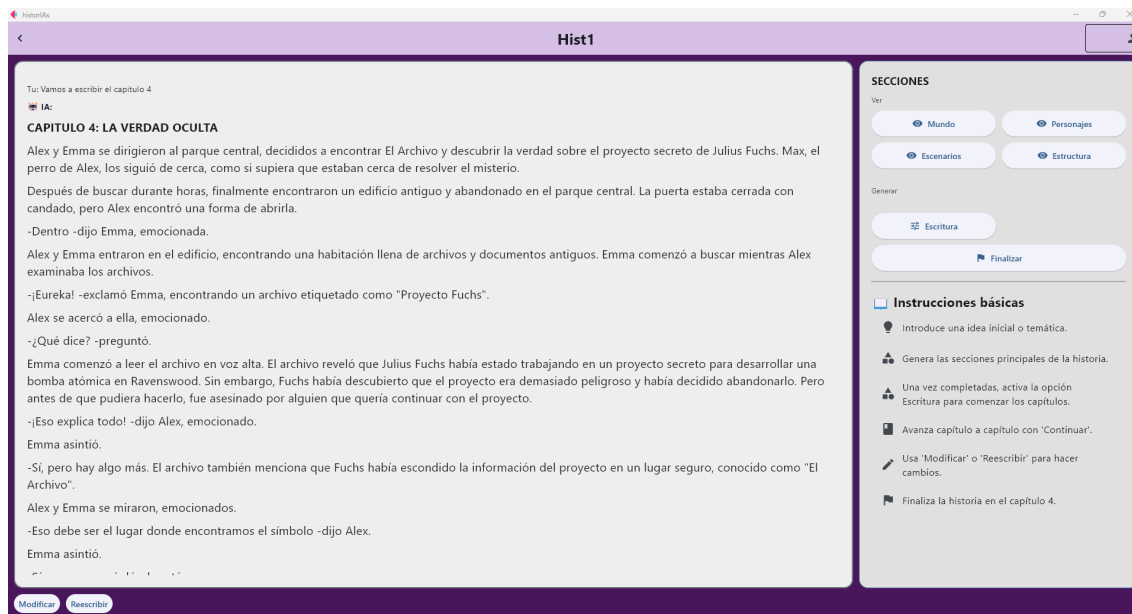


Figura 4.7: Historia de ejemplo: *Capítulo 4*.

Este sería un ejemplo de creación del *modo detallado* (ver historia completa en el Apéndice B.1). En la mayoría de pruebas realizadas, el modelo implementa de forma correcta las peticiones del usuario y mantiene un estilo de salida similar, aunque en ocasiones pueda haber algún inconveniente como por ejemplo, mostrar toda la respuesta en negrita. Se puede llegar al final generando las secciones en otro orden y realizando más o menos modificaciones. Podría ser interesante analizar si estas variaciones afectan a la calidad de la historia.

4.2. Evaluación de la historia

Tras la creación de la historias (ver Sección 4.1), se pide a los usuarios que completen una encuesta para evaluar tanto la calidad narrativa de la historia como su percepción de la escritura con IA. En la encuesta, mostrada en el Apéndice B.2, se incluye una *checklist* de ítems valorados en una escala de 1 (muy bajo) a 5 (excelente), que abarcaban aspectos como:

- **Coherencia global de la trama** y consistencia entre capítulos.
- **Desarrollo de personajes** y tratamiento de subtramas.
- **Transiciones y ritmo** narrativo.
- **Originalidad, impacto emocional y clímax/desenlace**.
- **Calidad del lenguaje** y descripción del mundo/ambientación.

Además, se incluye un bloque sobre la opinión general frente a la escritura tradicional, con ítems como la utilidad de la IA para superar bloqueos creativos, su papel como apoyo (no sustituto), la necesidad de revisión humana final y la calidad percibida en comparación con un autor humano. Finalmente, los participantes podían dejar comentarios abiertos sobre puntos fuertes, cabos sueltos o sugerencias de mejora, así como observaciones sobre la propia aplicación.

De esta manera, se obtuvieron datos tanto cuantitativos como cualitativos (comentarios y reflexiones). El análisis permite identificar carencias recurrentes como por ejemplo, lenguaje repetitivo o eventos planificados que no llegaron a desarrollarse y, al mismo tiempo, poner en valor el potencial de la herramienta para la creación de tramas, personajes y escenarios.

Conclusiones y Trabajo Futuro

En este trabajo se presenta **historIA**s, una aplicación orientada a generar historias por secciones, haciendo uso de los LLMs, para conseguir un mayor grado de personalización y edición por parte del usuario antes y durante la creación. La aplicación guarda la información en una base de datos de modo que el usuario puede retomar la historia en cualquier momento. También existe el *modo rápido* donde, tras rellenar un pequeño cuestionario con información, el modelo genera directamente la historia usando los datos introducidos.

Tras realizar numerosas pruebas se observa como las historias generadas seguían en muchas ocasiones una serie de similitudes y sesgos. Cuando se trataba de historias con temática parecida, repetía nombres de personaje y descripciones similares. Si el *prompt* no estaba muy detallado, en cualquier momento introducía aspectos mágicos y casi siempre empleaba el recurso del protagonista teniendo que derrotar al villano. La coherencia se mantenía la mayoría de ocasiones aunque hubo casos donde por ejemplo, en la sección de *descripción de escenarios* creaba un escenario y finalmente no aparecía en la historia. Al final se ha observado como estos modelos, a pesar de mantener cada vez mejor la coherencia e hilo de la historia, siguen siendo muy inferiores a la capacidad de redacción de un escritor profesional. La riqueza lingüística es más escasa repitiendo en muchas ocasiones palabras, expresiones o incluso frases completas. La calidad de las historias es muy simple, no se producen giros y sigue una estructura lineal haciendo que no produzca una gran emoción en el lector. En el estudio de Tian, Huang, et. al. [20] en la Figura 5.1 se muestra la evolución de la fortuna de los personajes:

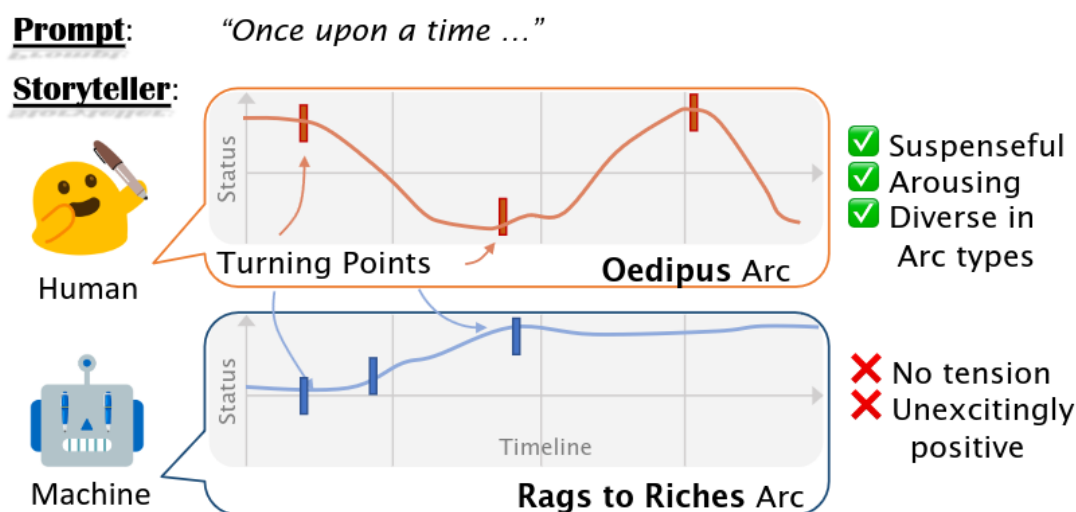


Figura 5.1: Comparación de arcos narrativos entre humanos y modelos de lenguaje (fuente del gráfico [20])

En este gráfico se expone la comparativa entre los arcos argumentales y las posiciones de los puntos de inflexión de las narrativas generadas por humanos y por LLMs. El eje vertical muestra la fortuna del personaje (de mala a buena) y el eje horizontal representa la línea del tiempo de la historia. En comparación con la escritura tradicional, los LLMs tienden a emplear arcos más alegres y menos complejos, introducen antes los puntos de inflexión en la trama y tienen menos suspense y contratiempos en sus narrativas. Esta misma tendencia se aprecia en las historias creadas con esta aplicación.

Otro aspecto importante del proyecto era analizar la comparativa en la generación de las historias entre el *modo rápido*, más similar a lo que hay en el mercado, con el *modo detallado* que se trata de la propuesta del proyecto. Tras realizar numerosas pruebas, el nivel de coherencia era similar aunque el *modo detallado* si creaba una historia más extensa y desarrollada. Uno de los aspectos más destacados del proyecto ha sido la funcionalidad de creación por secciones. Esta estructura no solo facilita la personalización y el control narrativo por parte del usuario, sino que también se ha observado como una herramienta muy útil para generar ideas y estimular la creatividad. Aunque la calidad literaria de los textos generados por los modelos aún presenta limitaciones, el proceso de construir una historia paso a paso resulta dinámico, entretenido y enriquecedor, según la experiencia de los usuarios que han probado la aplicación. En este sentido, la aplicación no solo cumple una función generativa, sino también inspiradora, sirviendo como punto de partida para que escritores desarrollen narrativas más profundas y originales a partir de las ideas creadas por el sistema.

Por este motivo considero una buena línea de trabajo continuar con la idea de crear las historias a través de secciones. Para ello se podría dar más flexibilidad al autor para que pueda elegir que secciones crear y no se limite únicamente a las cinco

predefinidas. Otro aspecto interesante es tener una mayor flexibilidad para poder definir una serie de eventos o acciones que el usuario quiera que ocurran durante la historia, así como para realizar modificaciones en secciones o capítulos que el usuario ya ha completado. La mayor dificultad durante todo el proyecto ha sido la limitación de *tokens* del modelo por parte de Groq, una mayor cantidad de *tokens* permitiría una mayor cantidad de secciones y capítulos. Otra mejora sería que el usuario pueda introducir nuevos personajes, debido a los *tokens* únicamente está limitado a un máximo de cuatro personajes. Sería interesante introducir una configuración inicial donde se especifiquen aspectos como: tipo de novela, a que público va dirigido, estilo narrativo, etc. Finalmente, se podría realizar una mejora en los *prompts* del sistema realizando técnicas de *prompting*.

Bibliografía

- [1] Amazon Web Service, <https://aws.amazon.com/es/>.
- [2] Flet, <https://flet.dev/>.
- [3] R. Goodwin, (2018). 1 the road. Ed. Jean Boîte Editions, 2018, ISBN-13: 978-2365680271.
- [4] Google Colab, <https://colab.research.google.com/>.
- [5] Groq, <https://groq.com/>.
- [6] D. Guisado Morcillo and ChatGPT, (2023). Iris. Ed. David Guisado Morcillo, 2023, ISBN-13: 978-8409477142.
- [7] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes and Yejin Choi, (2020). TThe curious case of Neural Text *D*egeneration. Published as a conference paper at ICLR 2020, arXiv:1904.09751v2 [cs.CL] 14 Feb 2020, <https://doi.org/10.48550/arXiv.1904.09751>.
- [8] Hugging Face, <https://huggingface.co/>.
- [9] Llama3.3, https://www.llama.com/docs/model-cards-and-prompt-formats/llama3_3/.
- [10] Microsoft Azure, <https://azure.microsoft.com/es-es>.
- [11] MongoDB, <https://www.mongodb.com/>.
- [12] J. Moor, (2006). The Dartmouth College Artificial Intelligence Conference: The Next Fifty Years. AI Magazine. 27. 87-91.
- [13] H. Naveed, A. U. Khan, S. Qiu, M. Saqib, S. Anwar, M. Usman, N. Akhtar, N. Barnes, and A. Mian, (2024). A Comprehensive Overview of Large Language Models. arxiv.org/abs/2307.06435, <https://doi.org/10.48550/arXiv.2307.06435>.

- [14] A. Newell and H. Simon, (1956). "The logic theory machine—A complex information processing system, *IRE Transactions on Information Theory*, vol. **2**, no. 3, pp. 61-79, September 1956, <https://doi.org/10.1109/TIT.1956.1056797>.
- [15] A. Newell, J. C. Shaw and H. A. Simon, (1959). Report on a general problem-solving program. *Proceedings of the International Conference on Information Processing*. pp. 256-264, http://findingaids.library.cmu.edu/repositories/2/archival_objects/22561.
- [16] Racter, (2018). *The Policeman's Beard is Half Constructed*. Ed. Grand Central Pub, 1984, ISBN-13: 978-0446380515.
- [17] Rai, Dilli Hang, (2024). Artificial Intelligence Through Time: A Comprehensive Historical Review. <https://doi.org/10.13140/RG.2.2.22835.03364>.
- [18] M. Renze and E. Guven, (2024). The effect of sampling temperature on problem solving in large language models. In *Fiddings of the Association for Computational Linguistics:EMNLP 2024*. pp. 7346-7356. Association for Computational Linguistics. <https://doi.org/10.48550/arXiv.2402.05201>.
- [19] Summarizer, <https://www.summarizer.org/>.
- [20] Y. Tian, T. Huang, M. Liu, D. Jiang, A. Spangher, M. Chen, J. May and N. Peng, (2024). Are large language models capable of generating human-level narratives? arXiv preprint arXiv:2407.13248. <https://doi.org/10.48550/arXiv.2407.13248>.
- [21] Transformers. <https://huggingface.co/docs/transformers/main/es/>.
- [22] A.M. Turing, (1956). I.-Computing machinery adn intelligence. *Mind* **LIX** n.236, pp.433-460, 1950. <https://doi.org/10.1093/mind/LIX.236.433>.
- [23] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser and I. Polosukhin, (2023). Attention is all you need. arXiv preprint arXiv:1706.03762. <https://doi.org/10.48550/arXiv.1706.03762>.
- [24] Wrizzle, <https://www.wrizzle.ai/es>.
- [25] Lili Yao, Nanyun Peng, Ralph Weischedel, Kevin Knight, Dongyan Zhao and Rui Yan, (2019). Plan-and-Write: Towards Better Automatic Storytelling. <https://doi.org/10.1609/aaai.v33i01.33017378>.

Apéndice A

Modelos Groq

A continuación se muestra una tabla con todos los LLMs disponibles en la versión gratuita de Groq [5] y sus limitaciones de *tokens*:

MODEL ID	RPM	RPD	TPM	TPD	ASH	ASD
allam-2-7b	30	7K	6K	500K	-	-
compound-beta	15	200	70K	-	-	-
compound-beta-mini	15	200	70K	-	-	-
deepseek-r1-distill-llama-70b	30	1K	6K	100K	-	-
gemma2-9b-it	30	14.4K	15K	500K	-	-
llama-3.1-8b-instant	30	14.4K	6K	500K	-	-
llama-3.3-70b-versatile	30	1K	12K	100K	-	-
llama3-70b-8192	30	14.4K	6K	500K	-	-
llama3-8b-8192	30	14.4K	6K	500K	-	-
meta-llama/llama-4-maverick-17b-128e-instruct	30	1K	6K	500K	-	-
meta-llama/llama-4-scout-17b-16e-instruct	30	1K	30K	500K	-	-
meta-llama/llama-guard-4-12b	30	14.4K	15K	500K	-	-
meta-llama/llama-prompt-guard-2-22m	30	14.4K	15K	500K	-	-
meta-llama/llama-prompt-guard-2-86m	30	14.4K	15K	500K	-	-
moonshotai/kimi-k2-instruct	60	1K	10K	300K	-	-
openai/gpt-oss-120b	30	1K	8K	200K	-	-
openai/gpt-oss-20b	30	1K	8K	200K	-	-
playai-tts	10	100	1.2K	3.6K	-	-
playai-tts-arabic	10	100	1.2K	3.6K	-	-
qwen/qwen3-32b	60	1K	6K	500K	-	-
whisper-large-v3	20	2K	-	-	7.2K	28.8K
whisper-large-v3-turbo	20	2K	-	-	7.2K	28.8K

Tabla A.1: Tabla LLM Groq

Apéndice **B**

Ejemplo de una historia y encuestas de los usuarios

A continuación se muestra el pdf con la historia generada en la Sección 4.1 junto con las encuestas:

B.1. Ejemplo de historia en modo detallado

Título: El enigma del símbolo

Capítulo 1: La Noche del Misterio

La noche era fresca y tranquila en el parque central de Ravenswood. Alex y Emma se reunieron en un banco cerca de la fuente, rodeados por los sonidos de la naturaleza y la suave iluminación del parque. Emma había preparado una carpeta con información sobre el proyecto de investigación que querían presentar en la universidad, mientras que Alex había traído su guitarra para tocar algunas canciones después de la reunión.

-¿Qué tal si empezamos con la introducción del proyecto?- sugirió Emma, abriendo la carpeta.

-Sí, adelante -respondió Alex, sacando su guitarra y comenzando a tocar una melodía suave.

Mientras Emma hablaba sobre el proyecto, Alex escuchaba atentamente, tomando notas y haciendo preguntas aquí y allá. Max, el perro de Alex, se acercó a ellos y se sentó a sus pies, mirando con curiosidad la carpeta de Emma.

De repente, Max se puso de pie y comenzó a ladrar suavemente. Alex y Emma se sorprendieron al ver que el perro se acercaba a un árbol cercano y comenzaba a rascar la corteza.

-¿Qué pasa, Max?- preguntó Alex, acercándose al árbol.

-¿Qué es eso?- preguntó Emma, señalando un extraño símbolo grabado en la corteza del árbol.

Alex y Emma se acercaron al árbol y examinaron el símbolo. Era un símbolo extraño, con patrones geométricos y símbolos que no reconocían.

-No tengo idea de qué significa esto -dijo Emma, sacando su teléfono para tomar una foto del símbolo.

-Yo tampoco -respondió Alex, mirando a su alrededor para ver si había alguien cerca que pudiera explicar el símbolo.

Mientras examinaban el símbolo, Max comenzó a ladrar más fuerte y a correr alrededor del árbol. Alex y Emma se miraron confundidos, sin entender qué estaba pasando.

-¿Qué pasa, Max?- preguntó Alex, intentando calmar al perro.

De repente, Max se detuvo y comenzó a mirar fijamente hacia el árbol. Alex y Emma siguieron su mirada y vieron que el símbolo comenzaba a brillar suavemente.

-¿Qué es esto?- preguntó Emma, asustada.

-No lo sé -respondió Alex, sintiendo un escalofrío en la espalda.

En ese momento, el símbolo desapareció, y el parque central se quedó en silencio. Alex y Emma se miraron confundidos, sin entender qué había pasado.

-¿Qué acabamos de ver?- preguntó Emma, aún asustada.

-No lo sé -respondió Alex, sacudiendo la cabeza- Pero creo que acabamos de descubrir algo importante.

Capítulo 2: El Misterio de Julius Fuchs

Alex y Emma se miraron confundidos, aún tratando de procesar lo que acababan de ver. El símbolo brillante había desaparecido tan rápido como había aparecido, dejándolos con más preguntas que respuestas. Max, el perro de Alex, seguía mirando fijamente hacia el árbol, como si esperara que algo más sucediera.

-¿Qué acabamos de ver?- repitió Emma, su voz llena de asombro.

-No lo sé -respondió Alex, sacudiendo la cabeza-. Pero creo que acabamos de descu-

brir algo importante.

Emma asintió, su mente ya trabajando en la teoría. -Tenemos que investigar más sobre ese símbolo. Puede que esté relacionado con el proyecto de investigación que queremos presentar.

Alex asintió, su propia curiosidad ya despertada. -Vamos a la biblioteca universitaria. Puede que encontremos algo sobre ese símbolo.

Emma se levantó del banco, recogiendo su carpeta y su laptop. -Vamos. Quiero saber qué significa ese símbolo.

Alex se levantó también, llamando a Max para que lo siguiera. Juntos, los tres se dirigieron hacia la biblioteca universitaria, con la intención de descubrir el secreto detrás del símbolo misterioso.

Una vez en la biblioteca, Emma y Alex se dirigieron directamente a la sección de historia y física. Emma comenzó a buscar en los estantes, mientras que Alex se sentó en una mesa con su laptop, intentando encontrar información en línea sobre el símbolo.

Después de varios minutos de búsqueda, Emma encontró un libro sobre la historia del Proyecto Manhattan. -¡Eureka! -exclamó, mostrando el libro a Alex.

Alex se acercó a ella, interesado. -¿Qué dice?

Emma comenzó a hojear el libro, buscando la información que necesitaban. -Aquí dice que Julius Fuchs, un físico alemán, trabajó en el Proyecto Manhattan durante la Segunda Guerra Mundial.

Alex se sorprendió. -¿Julius Fuchs? ¿No es el tipo que fue asesinado en Ravenswood en los años 40?

Emma asintió. -Ese mismo. Dice que su asesinato fue un misterio que nunca se resolvió.

Alex se inclinó hacia adelante, interesado. -¿Y qué relación tiene con el símbolo que vimos?

Emma continuó leyendo. -Dice que Fuchs trabajó en un proyecto secreto en Ravenswood, relacionado con la física nuclear. Puede que el símbolo sea una pista sobre ese proyecto.

Alex se sintió emocionado. -¡Eso es! Tenemos que investigar más sobre el proyecto secreto de Fuchs.

Emma asintió, su mente ya trabajando en la teoría. -Vamos a buscar más información sobre el proyecto. Puede que encontremos una conexión entre el símbolo y el asesinato de Fuchs.

Alex se levantó, decidido. -Vamos a resolver este misterio.

Emma sonrió, lista para la aventura. -Vamos a hacerlo.

Capítulo 3: La Pista del Pasado

Alex y Emma se sumergieron de lleno en la investigación sobre el proyecto secreto de Julius Fuchs. Pasaron horas en la biblioteca universitaria, buscando cualquier información que pudiera llevarlos a la verdad. Max, el perro de Alex, se quedó dormido a sus pies, cansado de tanto caminar y esperar.

Después de lo que parecía una eternidad, Emma finalmente encontró un documento que llamó la atención de Alex. Era una carta escrita por Fuchs mismo, dirigida a un colega suyo en el Proyecto Manhattan.

-“¡Eureka!” -exclamó Emma, mostrando la carta a Alex.

Alex se acercó a ella, emocionado. -¿Qué dice? -preguntó, ansioso por saber más.

Emma comenzó a leer la carta en voz alta. La carta hablaba sobre un experimento secreto que Fuchs había estado trabajando en Ravenswood, relacionado con la física nuclear. Pero lo que realmente llamó la atención de Alex y Emma fue la mención a un lugar llamado “El Archivo”.

-¿Qué es El Archivo? -preguntó Alex, intrigado.

Emma se encogió de hombros. -No tengo idea. Pero puede que sea una pista importante.

Alex asintió, decidido. -Vamos a encontrar El Archivo. Puede que contenga la clave para resolver el misterio del asesinato de Fuchs.

Emma asintió, lista para la aventura. -Vamos a hacerlo.

Juntos, Alex y Emma salieron de la biblioteca universitaria, decididos a encontrar El Archivo y descubrir la verdad sobre el proyecto secreto de Julius Fuchs.

Capítulo 4: La Verdad Oculta

Alex y Emma se dirigieron al parque central, decididos a encontrar El Archivo y descubrir la verdad sobre el proyecto secreto de Julius Fuchs. Max, el perro de Alex, los siguió de cerca, como si supiera que estaban cerca de resolver el misterio.

Después de buscar durante horas, finalmente encontraron un edificio antiguo y abandonado en el parque central. La puerta estaba cerrada con candado, pero Alex encontró

una forma de abrirla.

-Dentro -dijo Emma, emocionada.

Alex y Emma entraron en el edificio, encontrando una habitación llena de archivos y documentos antiguos. Emma comenzó a buscar mientras Alex examinaba los archivos.

-¡Eureka! -exclamó Emma, encontrando un archivo etiquetado como "Proyecto Fuchs".

Alex se acercó a ella, emocionado.

-¿Qué dice? -preguntó.

Emma comenzó a leer el archivo en voz alta. El archivo reveló que Julius Fuchs había estado trabajando en un proyecto secreto para desarrollar una bomba atómica en Ravenswood. Sin embargo, Fuchs había descubierto que el proyecto era demasiado peligroso y había decidido abandonarlo. Pero antes de que pudiera hacerlo, fue asesinado por alguien que quería continuar con el proyecto.

-¡Eso explica todo! -dijo Alex, emocionado.

Emma asintió.

-Sí, pero hay algo más. El archivo también menciona que Fuchs había escondido la información del proyecto en un lugar seguro, conocido como "El Archivo".

Alex y Emma se miraron, emocionados.

-Eso debe ser el lugar donde encontramos el símbolo -dijo Alex.

Emma asintió.

-Sí, y creo que sé dónde está.

Emma llevó a Alex y Max a un lugar en el parque central donde había un árbol con un agujero en la corteza. Emma metió la mano en el agujero y sacó un disco duro antiguo.

-Eso debe ser El Archivo -dijo Alex, emocionado.

Emma asintió y comenzó a examinar el disco duro. Después de unos minutos, encontró la información que estaban buscando.

-¡Lo hice! -exclamó Emma.

Alex se acercó a ella, emocionado.

-¿Qué dice? -preguntó.

Emma comenzó a leer la información en voz alta. El archivo reveló que Julius Fuchs había escondido la información del proyecto en un lugar seguro, para evitar que cayera

en malas manos.

También mencionaba que Fuchs había dejado una pista para que alguien pudiera encontrar la información en el futuro.

-¡Eso es increíble! -dijo Alex, emocionado.

Emma asintió.

-Sí, y creo que hemos resuelto el misterio.

Alex y Emma se miraron, satisfechos de haber resuelto el misterio del asesinato de Julius Fuchs. Max, el perro de Alex, ladró suavemente, como si estuviera contento de haber ayudado a resolver el misterio.

B.2. Encuestas realizadas por los usuarios

Se han realizado seis encuestas pero en el apéndice pondremos únicamente la correspondiente a la historia de ejemplo, una extra del *modo rápido* y la plantilla vacía.

Checklist de evaluación: usabilidad e historia generada

Rango de Edad: menor de 15 ☐ 15 – 25 ☐ 26 – 40 ☒ 41 – 60 ☐ más de 60 ☐
Modo usado: Rápido ☐ Detallado ☒

A) Evaluación de la historia generada

(1 = muy bajo · 5 = excelente)

1. Coherencia global de la trama: 1 ☐ 2 ☐ 3 ☐ 4 ☒ 5 ☐
2. Consistencia entre capítulos: 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☒
3. Desarrollo de personajes: 1 ☐ 2 ☒ 3 ☐ 4 ☐ 5 ☐
4. Decisiones y conflictos: 1 ☐ 2 ☐ 3 ☒ 4 ☐ 5 ☐
5. Subtramas: 1 ☒ 2 ☐ 3 ☐ 4 ☐ 5 ☐
6. Transiciones entre escenas/capítulos: 1 ☐ 2 ☐ 3 ☐ 4 ☒ 5 ☐
7. Ritmo de la historia: 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☒
8. Originalidad/interés de la historia: 1 ☐ 2 ☒ 3 ☐ 4 ☐ 5 ☐
9. Calidad del lenguaje: 1 ☒ 2 ☐ 3 ☐ 4 ☐ 5 ☐
10. Mundo/ambientación: 1 ☐ 2 ☐ 3 ☐ 4 ☒ 5 ☐
11. Clímax y desenlace: 1 ☒ 2 ☐ 3 ☐ 4 ☐ 5 ☐
12. Sesgos/estereotipos: 1 ☐ 2 ☐ 3 ☐ 4 ☒ 5 ☐
13. Impacto emocional: 1 ☐ 2 ☒ 3 ☐ 4 ☐ 5 ☐

Comentarios sobre la historia (puntos fuertes, cabos sueltos, sugerencias):

El lenguaje utilizado era muy repetitivo y algunas estructuras gramaticales eran dudosas.
 En la descripción de la estructura se ha hecho referencia a eventos que debían suceder en los capítulos que no han llegado a ocurrir y en la descripción de escenarios aparecía el apartamento de los protagonistas que no ha llegado a aparecer realmente en la historia.

B) Opinión: escritura tradicional e IA

(1 = muy en desacuerdo · 5 = muy de acuerdo)

1. Ayuda en bloqueos creativos: 1 ☐ 2 ☐ 3 ☒ 4 ☐ 5 ☐
2. IA como apoyo (no sustituto): 1 ☐ 2 ☐ 3 ☒ 4 ☐ 5 ☐

3. Calidad comparable a autor humano: 1 ☒ 2 ☐ 3 ☐ 4 ☐ 5 ☐
4. Estructura de capítulos y subtramas: 1 ☐ 2 ☒ 3 ☐ 4 ☐ 5 ☐
5. Revisión humana final imprescindible: 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☒
6. Recomendar IA a principiantes (como apoyo): 1 ☐ 2 ☐ 3 ☒ 4 ☐ 5 ☐

Comentarios/opinión libre:

Puede ayudar a generar una idea general o a crear escenarios y personajes pero no es fiable para llevar a cabo el desarrollo completo de la historia.

C) Observaciones finales sobre la aplicación**Lo mejor:**

Los botones hacen que sea intuitivo. Es fácil de usar.

Lo peor:**Sugerencias de mejora:**

No aparecen todas las secciones desarrolladas en la aplicación en el pdf de la historia. Sería interesante tener la descripción de personajes, escenarios y el mundo, especialmente cuando en la historia no se describe a los personajes.

Reflexión final sobre la experiencia:

Falta mejorar algunos aspectos pero en términos generales es una herramienta muy interesante con posibilidades de futuro.

Enviar PDF por correo

Checklist de evaluación: usabilidad e historia generada

Rango de Edad: menor de 15 ☐ 15 – 25 ☐ 26 – 40 ☐ 41 – 60 ☒ más de 60 ☐
Modo usado: Rápido ☒ Detallado ☐

A) Evaluación de la historia generada

(1 = muy bajo · 5 = excelente)

1. Coherencia global de la trama: 1 ☐ 2 ☒ 3 ☐ 4 ☐ 5 ☐
2. Consistencia entre capítulos: 1 ☐ 2 ☒ 3 ☐ 4 ☐ 5 ☐
3. Desarrollo de personajes: 1 ☐ 2 ☒ 3 ☐ 4 ☐ 5 ☐
4. Decisiones y conflictos: 1 ☒ 2 ☐ 3 ☐ 4 ☐ 5 ☐
5. Subtramas: 1 ☒ 2 ☐ 3 ☐ 4 ☐ 5 ☐
6. Transiciones entre escenas/capítulos: 1 ☐ 2 ☒ 3 ☐ 4 ☐ 5 ☐
7. Ritmo de la historia: 1 ☐ 2 ☐ 3 ☒ 4 ☐ 5 ☐
8. Originalidad/interés de la historia: 1 ☐ 2 ☐ 3 ☒ 4 ☐ 5 ☐
9. Calidad del lenguaje: 1 ☒ 2 ☐ 3 ☐ 4 ☐ 5 ☐
10. Mundo/ambientación: 1 ☐ 2 ☒ 3 ☐ 4 ☐ 5 ☐
11. Clímax y desenlace: 1 ☒ 2 ☐ 3 ☐ 4 ☐ 5 ☐
12. Sesgos/estereotipos: 1 ☐ 2 ☒ 3 ☐ 4 ☐ 5 ☐
13. Impacto emocional: 1 ☐ 2 ☒ 3 ☐ 4 ☐ 5 ☐

Comentarios sobre la historia (puntos fuertes, cabos sueltos, sugerencias):

Se le olvido que hacia calor (una de las condiciones de la historia)

Repite las mismas palabras todo el rato.

No es muy "razonable" en algunas cosas, no posible, aunque la historia era contemporanea y realista.

La historia es entretenida y si quieres una idea para escribir tu propia historia, da buenas ideas.

B) Opinión: escritura tradicional e IA

(1 = muy en desacuerdo · 5 = muy de acuerdo)

1. Ayuda en bloqueos creativos: 1 ☐ 2 ☐ 3 ☐ 4 ☒ 5 ☐
2. IA como apoyo (no sustituto): 1 ☐ 2 ☐ 3 ☒ 4 ☐ 5 ☐

3. Calidad comparable a autor humano: 1 ☒ 2 ☐ 3 ☐ 4 ☐ 5 ☐
4. Estructura de capítulos y subtramas: 1 ☒ 2 ☐ 3 ☐ 4 ☐ 5 ☐
5. Revisión humana final imprescindible: 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☒
6. Recomendar IA a principiantes (como apoyo): 1 ☐ 2 ☒ 3 ☐ 4 ☐ 5 ☐

Comentarios/opinión libre:

C) Observaciones finales sobre la aplicación

Lo mejor:

Puede dar ideas interesantes para crear tu propia historia (una ayuda para empezar).

Lo peor:

No tiene mucho vocabulario, no crea muchos personajes secundarios, no describe los personajes en profundidad. No da una final real a la historia.

Sugerencias de mejora:

Aumentar MUCHO el vocabulario, mas trama. Aunque sea una historia corta, que tenga desenlace.

Reflexión final sobre la experiencia:

Ha sido divertido. He podido crear más historias y puede ofrecer ideas interesantes.

Enviar PDF por correo

Checklist de evaluación: usabilidad e historia generada

Rango de Edad: menor de 15 ☐ 15 – 25 ☐ 26 – 40 ☐ 41 – 60 ☐ más de 60 ☐
Modo usado: Rápido ☐ Detallado ☐

A) Evaluación de la historia generada

(1 = muy bajo · 5 = excelente)

1. Coherencia global de la trama: 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
2. Consistencia entre capítulos: 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
3. Desarrollo de personajes: 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
4. Decisiones y conflictos: 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
5. Subtramas: 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
6. Transiciones entre escenas/capítulos: 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
7. Ritmo de la historia: 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
8. Originalidad/interés de la historia: 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
9. Calidad del lenguaje: 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
10. Mundo/ambientación: 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
11. Clímax y desenlace: 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
12. Sesgos/estereotipos: 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
13. Impacto emocional: 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

Comentarios sobre la historia (puntos fuertes, cabos sueltos, sugerencias):

B) Opinión: escritura tradicional e IA

(1 = muy en desacuerdo · 5 = muy de acuerdo)

1. Ayuda en bloqueos creativos: 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
2. IA como apoyo (no sustituto): 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

3. Calidad comparable a autor humano: 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
4. Estructura de capítulos y subtramas: 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
5. Revisión humana final imprescindible: 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
6. Recomendar IA a principiantes (como apoyo): 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

Comentarios/opinión libre:

C) Observaciones finales sobre la aplicación

Lo mejor:

Lo peor:

Sugerencias de mejora:

Reflexión final sobre la experiencia:

Enviar PDF por correo

