Generación de diálogos para historias utilizando Large Language Models
Dialog generation for stories 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

Colaborador

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

Generación de diálogos para historias utilizando Large Language Models
Dialog generation for stories 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

Colaborador

Convocatoria: *Febrero/Junio/Septiembre 2025*

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

DIA de MES de AÑO

Dedicatoria

A Pedro Pablo y Marco Antonio, por crear
TeXiS e iluminar nuestro camino

Agradecimientos

A Guillermo, por el tiempo empleado en hacer estas plantillas. A Adrián, Enrique y Nacho, por sus comentarios para mejorar lo que hicimos. Y a Narciso, a quien no le ha hecho falta el Anillo Único para coordinarnos a todos.

Resumen

Generación de diálogos para historias utilizando Large Language Models

Un resumen en castellano de media página, incluyendo el título en castellano. A continuación, se escribirá una lista de no más de 10 palabras clave.

Palabras clave

Máximo 10 palabras clave separadas por comas

Abstract

Dialog generation for stories using Large Language Models

An abstract in English, half a page long, including the title in English. Below, a list with no more than 10 keywords.

Keywords

10 keywords max., separated by commas.

Índice

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

"Frase célebre dicha por alguien inteligente"
— Autor

Según la normativa para Trabajos de Fin de Grado[1], la memoria incluirá una portada normalizada con la siguiente información: título en castellano, título en inglés, autores, profesor director, codirector si es el caso, curso académico e identificación de la asignatura (Trabajo de fin de grado del Grado en - nombre del grado correspondiente-, Facultad de Informática, Universidad Complutense de Madrid). Los datos referentes al título y director (y codirector en su caso) deben corresponder a los publicados en la página web de TFG.

La memoria debe incluir la descripción detallada de la propuesta hardware/software realizada y ha de contener:

1. un índice,
2. un resumen y una lista de no más de 10 palabras clave para su búsqueda bibliográfica, ambos en castellano e inglés,
3. una introducción con los antecedentes, objetivos y plan de trabajo,
4. resultados y discusión crítica y razonada de los mismos, con sus conclusiones,
5. bibliografía.

Para facilitar la escritura de la memoria siguiendo esta estructura, el estudiante podrá usar las plantillas en LaTeX o Word preparadas al efecto y publicadas en la página web de TFG.

La memoria constará de un mínimo de 25 páginas para los proyectos realizados por un único estudiante, y de al menos 5 páginas más por cada integrante adicional del grupo. En este número de páginas solo se tiene en cuenta el contenido correspondiente a los apartados c y d del punto anterior.

La memoria puede estar escrita en castellano o inglés, pero en el primer caso la introducción y las conclusiones deben aparecer también en inglés. Las memorias de

[1] `https://informatica.ucm.es/file/normativatfg-2021-2022?ver` (ver versión actualizada para cada curso académico)

los TFG matriculados en el grupo I deberán estar escritas íntegramente en inglés, excepto por lo especificado en los puntos 1 y 2 anteriores (título, resumen y lista de palabras clave).

En caso de trabajos no unipersonales, cada participante indicará en la memoria su contribución al proyecto con una extensión de al menos dos páginas por cada uno de los participantes.

Todo el material no original, ya sea texto o figuras, deberá ser convenientemente citado y referenciado. En el caso de material complementario se deben respetar las licencias y *copyrights* asociados al software y hardware que se emplee. En caso contrario no se autorizará la defensa, sin menoscabo de otras acciones que correspondan.

1.1. Motivación

Introducción al tema del TFG.

1.2. Objetivos

Descripción de los objetivos del trabajo.

1.3. Plan de trabajo

Aquí se describe el plan de trabajo a seguir para la consecución de los objetivos descritos en el apartado anterior.

1.4. Explicaciones adicionales sobre el uso de esta plantilla

Si quieres cambiar el **estilo del título** de los capítulos del documento, edita el fichero `TeXiS\TeXiS_pream.tex` y comenta la línea `\usepackage[Lenny]{fncychap}` para dejar el estilo básico de LaTeX.

Si no te gusta que no haya **espacios entre párrafos** y quieres dejar un pequeño espacio en blanco, no metas saltos de línea (\\) al final de los párrafos. En su lugar, busca el comando `\setlength{\parskip}{0.2ex}` en `TeXiS\TeXiS_pream.tex` y aumenta el valor de 0,2*ex* a, por ejemplo, 1*ex*.

TFGTeXiS se ha elaborado a partir de la plantilla de TeXiS[2], creada por Marco Antonio y Pedro Pablo Gómez Martín para escribir su tesis doctoral. Para explicaciones más extensas y detalladas sobre cómo usar esta plantilla, recomendamos la lectura del documento `TeXiS-Manual-1.0.pdf` que acompaña a esta plantilla.

El siguiente texto se genera con el comando `\lipsum[2-20]` que viene a continuación en el fichero .tex. El único propósito es mostrar el aspecto de las páginas

[2] `http://gaia.fdi.ucm.es/research/texis/`

usando esta plantilla. Quita este comando y, si quieres, comenta o elimina el paquete *lipsum* al final de `TeXiS\TeXiS_pream.tex`

1.4.1. Texto de prueba

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta

tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Donec odio elit, dictum in, hendrerit sit amet, egestas sed, leo. Praesent feugiat sapien aliquet odio. Integer vitae justo. Aliquam vestibulum fringilla lorem. Sed neque lectus, consectetuer at, consectetuer sed, eleifend ac, lectus. Nulla facilisi. Pellentesque eget lectus. Proin eu metus. Sed porttitor. In hac habitasse platea dictumst. Suspendisse eu lectus. Ut mi mi, lacinia sit amet, placerat et, mollis vitae, dui. Sed ante tellus, tristique ut, iaculis eu, malesuada ac, dui. Mauris nibh leo, facilisis non, adipiscing quis, ultrices a, dui.

Morbi luctus, wisi viverra faucibus pretium, nibh est placerat odio, nec commodo wisi enim eget quam. Quisque libero justo, consectetuer a, feugiat vitae, porttitor eu, libero. Suspendisse sed mauris vitae elit sollicitudin malesuada. Maecenas ultricies eros sit amet ante. Ut venenatis velit. Maecenas sed mi eget dui varius euismod. Phasellus aliquet volutpat odio. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Pellentesque sit amet pede ac sem eleifend consectetuer. Nullam elementum, urna vel imperdiet sodales, elit ipsum pharetra ligula, ac pretium ante justo a nulla. Curabitur tristique arcu eu metus. Vestibulum lectus. Proin mauris. Proin eu nunc eu urna hendrerit faucibus. Aliquam auctor, pede consequat laoreet varius, eros tellus scelerisque quam, pellentesque hendrerit ipsum dolor sed augue. Nulla nec lacus.

Suspendisse vitae elit. Aliquam arcu neque, ornare in, ullamcorper quis, commodo eu, libero. Fusce sagittis erat at erat tristique mollis. Maecenas sapien libero, molestie et, lobortis in, sodales eget, dui. Morbi ultrices rutrum lorem. Nam elementum ullamcorper leo. Morbi dui. Aliquam sagittis. Nunc placerat. Pellentesque tristique sodales est. Maecenas imperdiet lacinia velit. Cras non urna. Morbi eros pede, suscipit ac, varius vel, egestas non, eros. Praesent malesuada, diam id pretium elementum, eros sem dictum tortor, vel consectetuer odio sem sed wisi.

Sed feugiat. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Ut pellentesque augue sed urna. Vestibulum diam eros, fringilla et, consectetuer eu, nonummy id, sapien. Nullam at lectus. In sagittis ultrices mauris. Curabitur malesuada erat sit amet massa. Fusce blandit. Aliquam erat volutpat. Aliquam euismod. Aenean vel lectus. Nunc imperdiet justo nec dolor.

Etiam euismod. Fusce facilisis lacinia dui. Suspendisse potenti. In mi erat, cursus id, nonummy sed, ullamcorper eget, sapien. Praesent pretium, magna in eleifend egestas, pede pede pretium lorem, quis consectetuer tortor sapien facilisis magna. Mauris quis magna varius nulla scelerisque imperdiet. Aliquam non quam. Aliquam porttitor quam a lacus. Praesent vel arcu ut tortor cursus volutpat. In vitae pede quis diam bibendum placerat. Fusce elementum convallis neque. Sed dolor orci, scelerisque ac, dapibus nec, ultricies ut, mi. Duis nec dui quis leo sagittis commodo.

Aliquam lectus. Vivamus leo. Quisque ornare tellus ullamcorper nulla. Mauris porttitor pharetra tortor. Sed fringilla justo sed mauris. Mauris tellus. Sed non leo. Nullam elementum, magna in cursus sodales, augue est scelerisque sapien, venenatis

congue nulla arcu et pede. Ut suscipit enim vel sapien. Donec congue. Maecenas urna mi, suscipit in, placerat ut, vestibulum ut, massa. Fusce ultrices nulla et nisl.

Etiam ac leo a risus tristique nonummy. Donec dignissim tincidunt nulla. Vestibulum rhoncus molestie odio. Sed lobortis, justo et pretium lobortis, mauris turpis condimentum augue, nec ultricies nibh arcu pretium enim. Nunc purus neque, placerat id, imperdiet sed, pellentesque nec, nisl. Vestibulum imperdiet neque non sem accumsan laoreet. In hac habitasse platea dictumst. Etiam condimentum facilisis libero. Suspendisse in elit quis nisl aliquam dapibus. Pellentesque auctor sapien. Sed egestas sapien nec lectus. Pellentesque vel dui vel neque bibendum viverra. Aliquam porttitor nisl nec pede. Proin mattis libero vel turpis. Donec rutrum mauris et libero. Proin euismod porta felis. Nam lobortis, metus quis elementum commodo, nunc lectus elementum mauris, eget vulputate ligula tellus eu neque. Vivamus eu dolor.

Nulla in ipsum. Praesent eros nulla, congue vitae, euismod ut, commodo a, wisi. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Aenean nonummy magna non leo. Sed felis erat, ullamcorper in, dictum non, ultricies ut, lectus. Proin vel arcu a odio lobortis euismod. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Proin ut est. Aliquam odio. Pellentesque massa turpis, cursus eu, euismod nec, tempor congue, nulla. Duis viverra gravida mauris. Cras tincidunt. Curabitur eros ligula, varius ut, pulvinar in, cursus faucibus, augue.

Nulla mattis luctus nulla. Duis commodo velit at leo. Aliquam vulputate magna et leo. Nam vestibulum ullamcorper leo. Vestibulum condimentum rutrum mauris. Donec id mauris. Morbi molestie justo et pede. Vivamus eget turpis sed nisl cursus tempor. Curabitur mollis sapien condimentum nunc. In wisi nisl, malesuada at, dignissim sit amet, lobortis in, odio. Aenean consequat arcu a ante. Pellentesque porta elit sit amet orci. Etiam at turpis nec elit ultricies imperdiet. Nulla facilisi. In hac habitasse platea dictumst. Suspendisse viverra aliquam risus. Nullam pede justo, molestie nonummy, scelerisque eu, facilisis vel, arcu.

Curabitur tellus magna, porttitor a, commodo a, commodo in, tortor. Donec interdum. Praesent scelerisque. Maecenas posuere sodales odio. Vivamus metus lacus, varius quis, imperdiet quis, rhoncus a, turpis. Etiam ligula arcu, elementum a, venenatis quis, sollicitudin sed, metus. Donec nunc pede, tincidunt in, venenatis vitae, faucibus vel, nibh. Pellentesque wisi. Nullam malesuada. Morbi ut tellus ut pede tincidunt porta. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam congue neque id dolor.

Donec et nisl at wisi luctus bibendum. Nam interdum tellus ac libero. Sed sem justo, laoreet vitae, fringilla at, adipiscing ut, nibh. Maecenas non sem quis tortor eleifend fermentum. Etiam id tortor ac mauris porta vulputate. Integer porta neque vitae massa. Maecenas tempus libero a libero posuere dictum. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Aenean quis mauris sed elit commodo placerat. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Vivamus rhoncus tincidunt libero. Etiam elementum pretium justo. Vivamus est. Morbi a tellus eget pede tristique commodo. Nulla nisl. Vestibulum sed nisl eu sapien cursus rutrum.

Nulla non mauris vitae wisi posuere convallis. Sed eu nulla nec eros scelerisque pharetra. Nullam varius. Etiam dignissim elementum metus. Vestibulum faucibus,

metus sit amet mattis rhoncus, sapien dui laoreet odio, nec ultricies nibh augue a enim. Fusce in ligula. Quisque at magna et nulla commodo consequat. Proin accumsan imperdiet sem. Nunc porta. Donec feugiat mi at justo. Phasellus facilisis ipsum quis ante. In ac elit eget ipsum pharetra faucibus. Maecenas viverra nulla in massa.

Nulla ac nisl. Nullam urna nulla, ullamcorper in, interdum sit amet, gravida ut, risus. Aenean ac enim. In luctus. Phasellus eu quam vitae turpis viverra pellentesque. Duis feugiat felis ut enim. Phasellus pharetra, sem id porttitor sodales, magna nunc aliquet nibh, nec blandit nisl mauris at pede. Suspendisse risus risus, lobortis eget, semper at, imperdiet sit amet, quam. Quisque scelerisque dapibus nibh. Nam enim. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Nunc ut metus. Ut metus justo, auctor at, ultrices eu, sagittis ut, purus. Aliquam aliquam.

Capítulo 2

Estado de la Cuestión

En el estado de la cuestión es donde aparecen gran parte de las referencias bibliográficas del trabajo. Una de las formas más cómodas de gestionar la bibliografía en LaTeX es utilizando **bibtex**. Las entradas bibliográficas deben estar en un fichero con extensión *.bib* (con esta plantilla se proporciona el fichero biblio.bib, donde están las entradas referenciadas más abajo). Cada entrada bibliográfica tiene una clave que permite referenciarla desde cualquier parte del texto con los siguiente comandos:

- Referencia bibliografica con cite: **?**
- Referencia bibliográfica con citep: (**?**)
- Referencia bibliográfica con citet: **?**

Es posible citar más de una fuente, como por ejemplo (**???**)

Después, LaTeXse ocupa de rellenar la sección de bibliografía con las entradas **que hayan sido citadas** (es decir, no con todas las entradas que hay en el .bib, sino sólo con aquellas que se hayan citado en alguna parte del texto).

Bibtex es un programa separado de latex, pdflatex o cualquier otra cosa que se use para compilar los .tex, de manera que para que se rellene correctamente la sección de bibliografía es necesario compilar primero el trabajo (a veces es necesario compilarlo dos veces), compilar después con bibtex, y volver a compilar otra vez el trabajo (de nuevo, puede ser necesario compilarlo dos veces).

Capítulo 3

Descripción del Trabajo

3.1. Diseño de la aplicación

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. Para poder abordar con más detalle el diseño de la aplicación, haremos uso de distintos diagramas.

3.1.1. Diagramas de flujo

Se ha dividido el diagrama de flujo para facilitar el detalle y la comprensión. El primero de ellos se trata del diagrama de flujo inicial, cuando el usuario inicia la aplicación

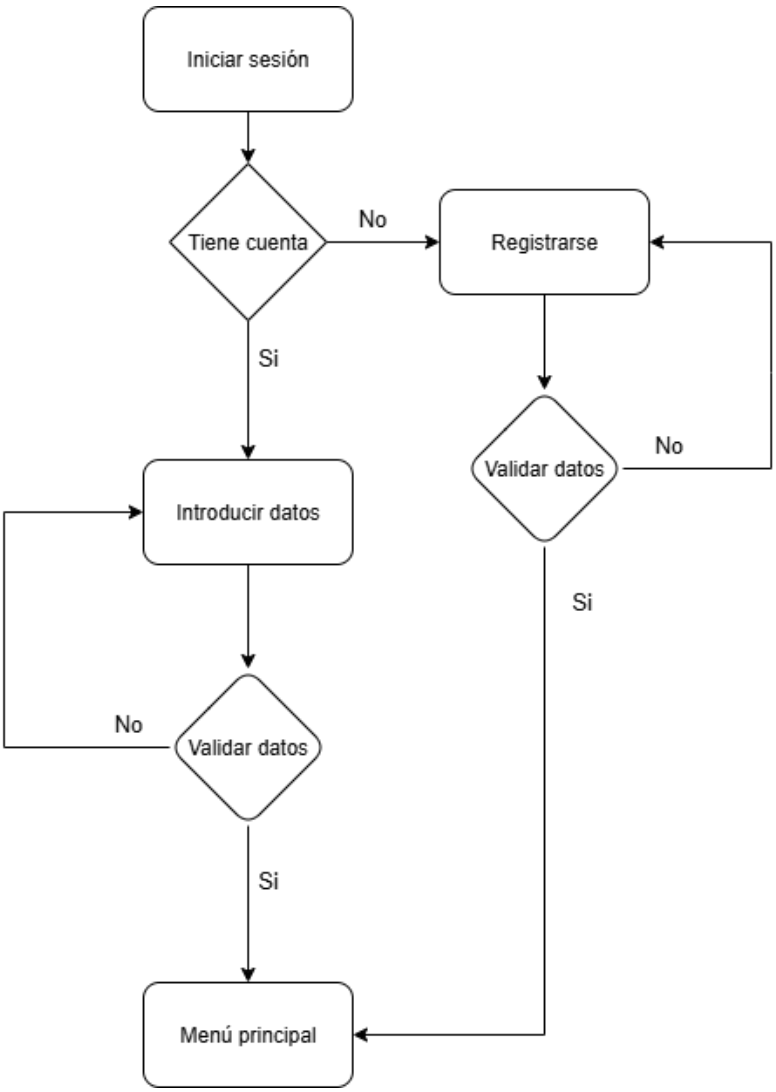

Figura 3.1

Al iniciar la aplicación el usuario se encuentra la pantalla de login donde tiene que iniciar sesión con su usuario y contraseña y si no dispone de una se tiene que registrar creando una nueva. Se validan los datos introducidos y si son válidos se accede con tu usuario al menú principal.

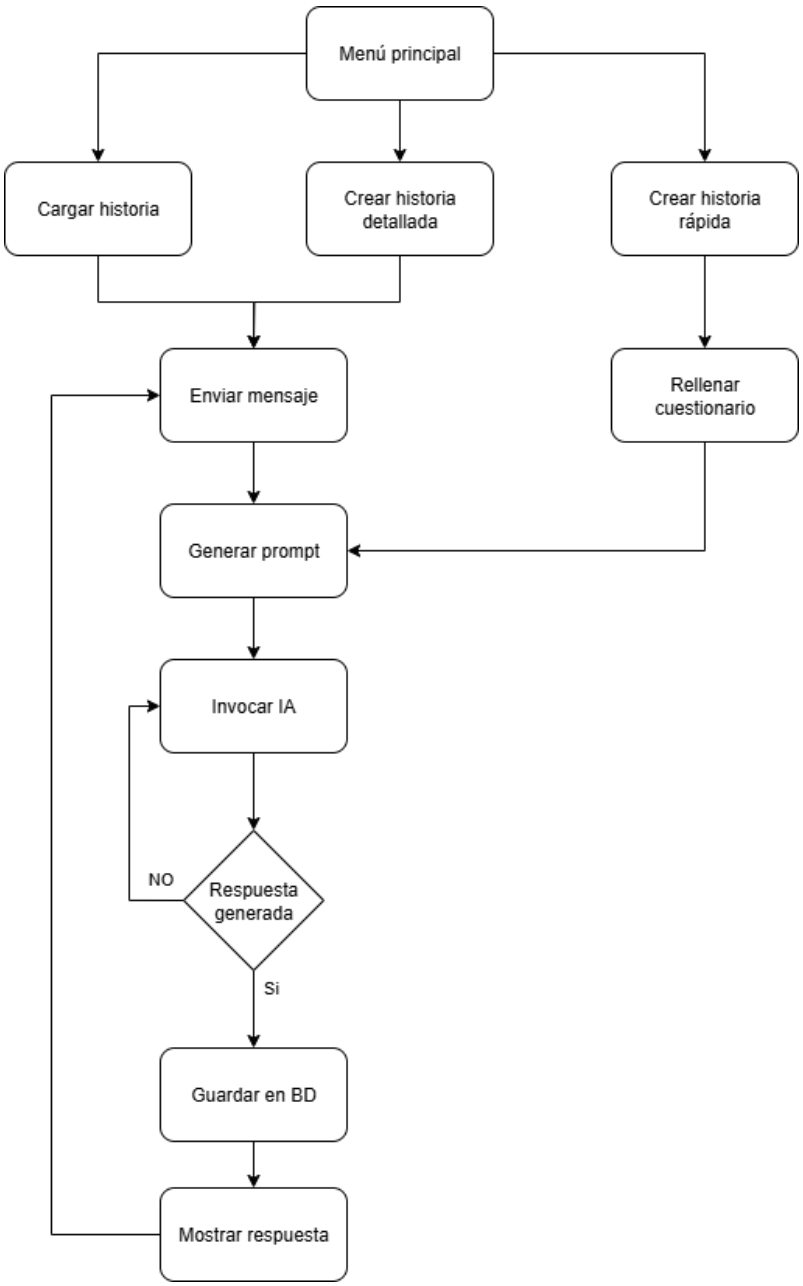

Figura 3.2

Una vez se accede al menú principal se tiene tres opciones para realizar: cargar historia, crear historia detallada o crear historia rápida. En cargar historia y crear historia detallada la interfaz es la misma y se tiene que enviar el mensaje ya sea para continuar o comenzar con una nueva historia. En el modo rápido lo que cambia es que se tiene un pequeño cuestionario con los datos más importantes que se tendrá que rellenar o dejar campos en blanco haciendo que el modelo lo genere a su gusto. Una vez enviado el cuestionario o el mensaje, se construye internamente el `prompt` y se envía al servidor de `Groq` y genere la respuesta. Si se produce algún a la hora de generar la respuesta, se vuelve a invocar a la IA para conseguir una respuesta válida. Una vez se genera la respuesta por parte del modelo, se guarda en la base de

datos y se muestra por la interfaz al usuario.

3.1.2. Diagrama de componentes

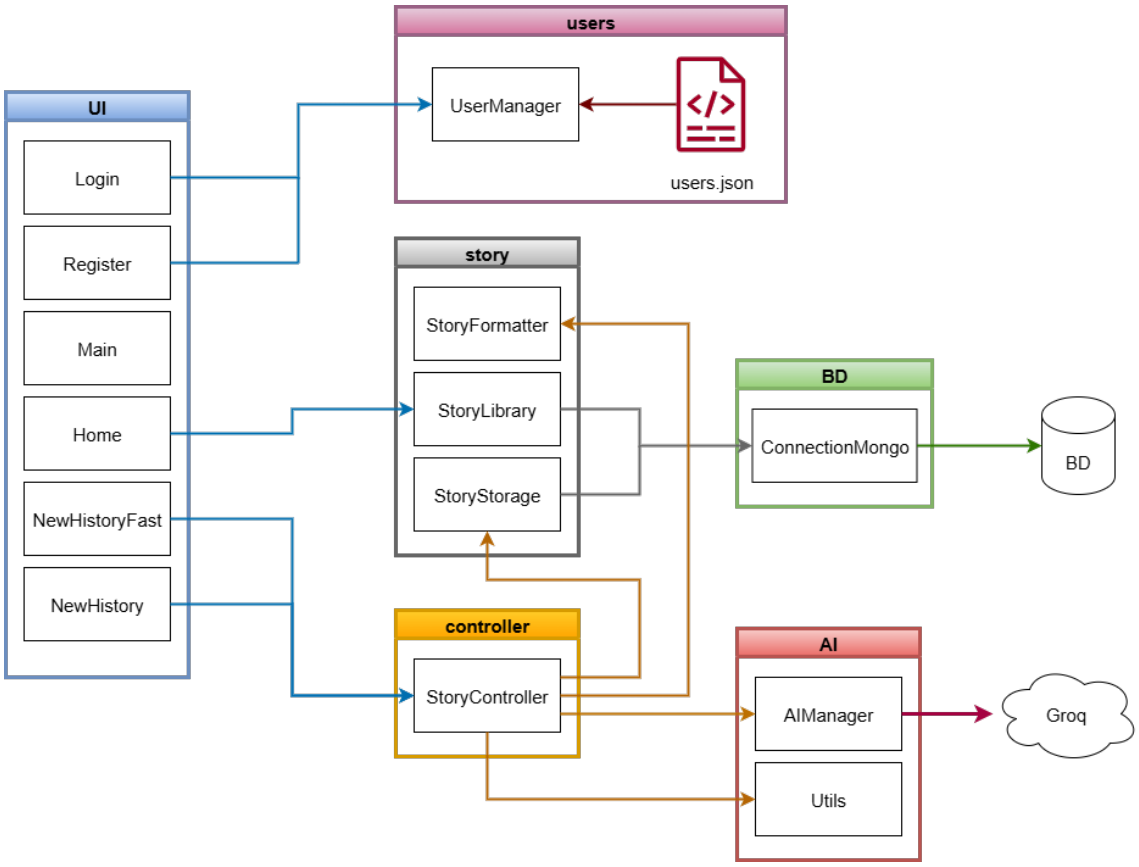

Figura 3.3

En este diagrama se observa como está estructurada la aplicación. Esta dividida en distintos paquete:

- UI: 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.

- 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 novelas. `StoryFormatter` se encarga de todo lo relacionado con formatear el texto para convertirlo en un formato `JSON` que sea compatible con `MongoDB`. `StoryLibrary` se encarga de gestionar las historias creadas por por el usuario y `StoryStorage` se encargar de crear 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 encarga de llevar la lógica del procesamiento del mensaje, detectar la sección por 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: Contiene la clase `ConnectionMongo` encargada de realizar las consultas en la base de datos.

- AI: Contiene `AIManager` encargado 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.

3.1.3. Interacción con la IA

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 `deepseek-r1-distill-llama-70b`, para generar el contenido narrativo. Este contenido se genera siguiendo una serie de pasos que se van a explicar en esta sección.(Explicar porque uso el modelo) Cada vez que el usuario quiere enviar un mensaje desde `NewHistoryFast`, 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 enviar nuevos mensajes. Esto permite que el usuario no pueda generar múltiples mensajes simultáneamente mientras se está generando una respuesta, evitando así colisiones y duplicación de peticiones al servidor. Además este *thread* se conecta con `StoryController` que se encarga de realizar el procesamiento del mensaje introducido. (De momento solo está para el modo detallado)(Añadir un diagrama)

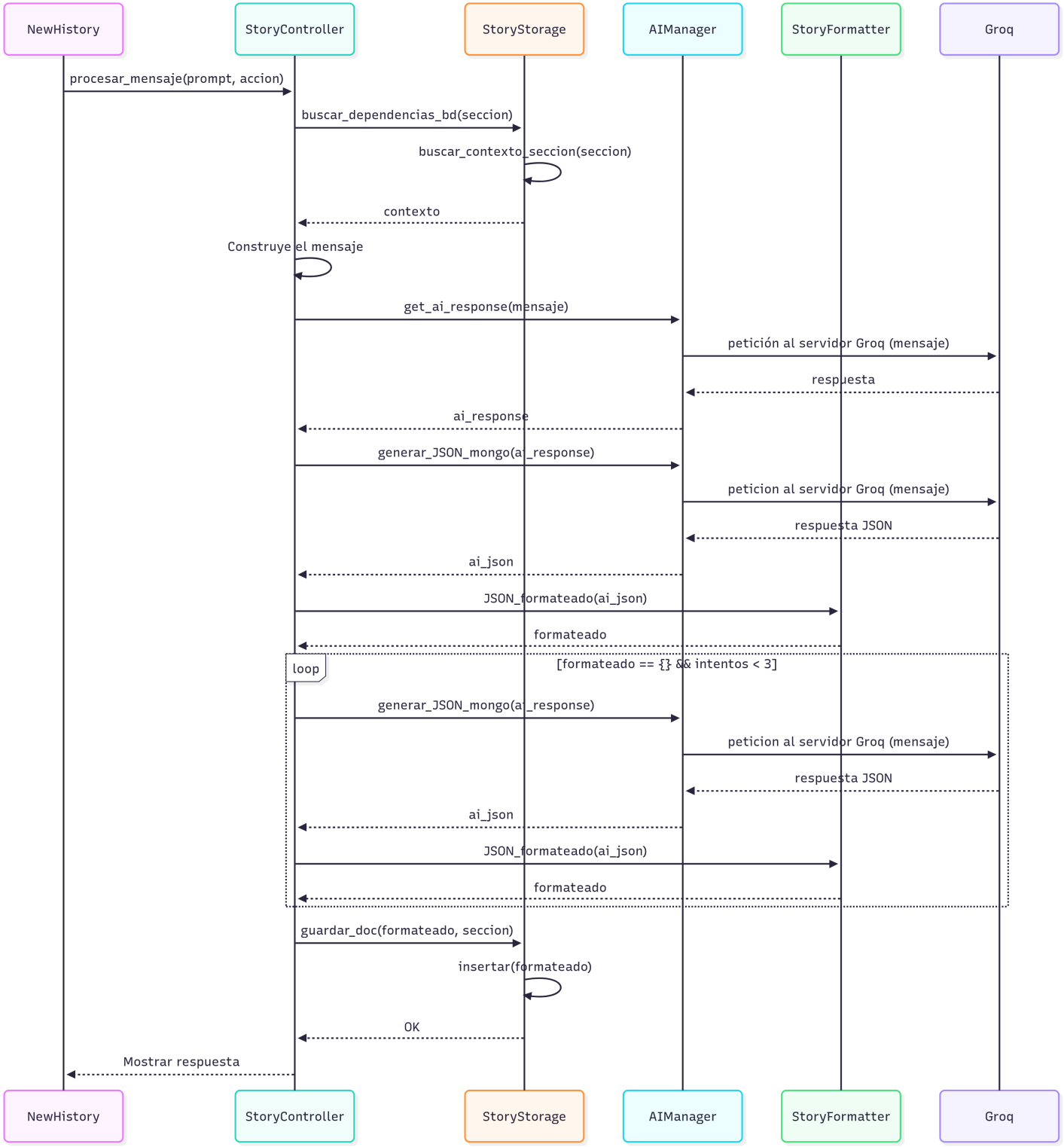

Figura 3.4: Diagrama de secuencia

Como se observar, `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` que se encarga de buscar para dicha sección que otras secciones se necesita pasar al 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`. Ahora se construye el mensaje formado por el contexto creado anteriormente, se indica 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 guardar en formato JSON. Durante el desarrollo nos encontramos con un problema fundamental, la salida del LLM, aunque en la mayoría de casos seguía la plantilla esperada, no siempre respetaba de forma exacta la misma estructura JSON. 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. Para poder solucionar estas dificultades se opta por hacer uso nuevamente del LLM enviándole directamente la misma cadena y pidiendo que lo convierta en un objeto JSON válido. De este modo, la IA actúa como "formateado" de su propio `output`, liberándonos de la necesidad de escribir código de `parsing` extremadamente específico. Al igual que antes, el modelo no siempre genera un JSON válido lo que provoca error al querer guardar en la base de datos. Para solucionar estos errores se implementa un bucle haciendo que el modelo genere un nuevo JSON, además, se añade un contador para evitar que se pueda quedar demasiado tiempo en el bucle. Tras varias pruebas nos damos cuenta que con limitar el bucle a 3 iteraciones conseguimos prácticamente una tasa perfecta de acierto.

Con este nuevo mensaje 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 de 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. 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, 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. (Añadire unas referencias y graficas de articulos)

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 3.1: Ajuste de `temperature` según el tipo de tarea

- `max_tokens`: Define el número máximo de *tokens* que puede devolver el modelo de lenguaje en una respuesta. Se ha establecido este tope a 4.096 *tokens* lo cual corresponde aproximadamente entre 2.000 y 2.500 palabras. Esta cantidad debe de convivir con el *prompt del sistema* y el mensaje generado, por eso se debe de 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.

3.1.4. Almacenamiento en la base de datos

Para este proyecto unicamente 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` 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.

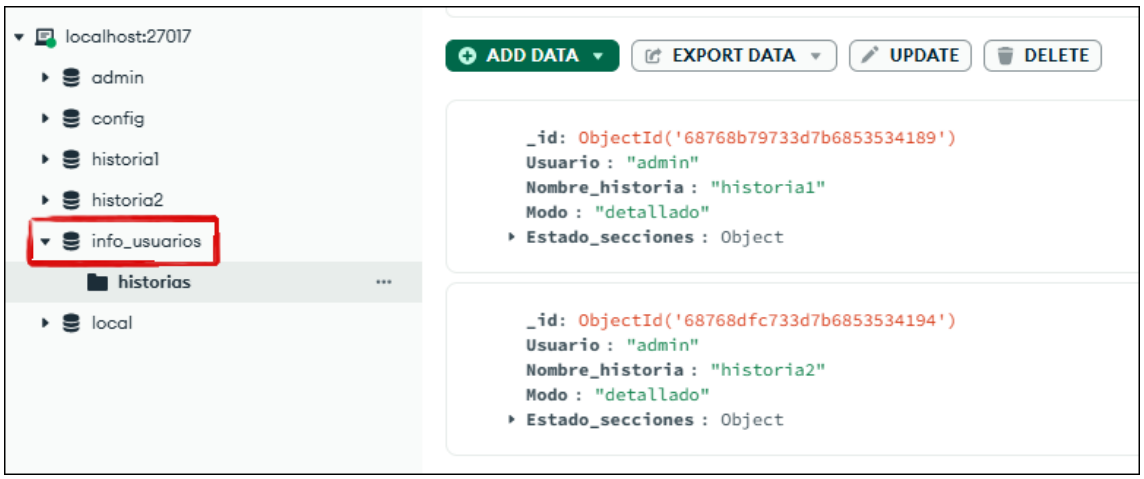

Figura 3.5

Dentro de la base de datos de cada historia se guarda las colecciones de cada una de las secciones 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` guarda el contenido de cada una de las secciones en formato de texto plano y para facilitar su uso para construir el mensaje que se le pasa al modelo.

Figura 3.6

Figura 3.7

3.1.5. Gestión de prompts

Uno de los aspectos clave en el proyecto es la creación del *prompt de sistema* para que el modelo sepa lo que tiene que hacer. En estos *prompts* se explican las reglas e instrucciones que se tiene que seguir a la hora de generar una respuesta. Es importante comentar que aunque se concreten unas reglas y formato deseado, a la hora de generar las respuestas en ocasiones falla y no sigue las normas preestablecidas. A continuación se explican los *prompts* más importantes:

- PROMPT_INICIAL: En el inicio se creaó únicamente el *PROMPT_INICIAL* donde se le da el contexto a la IA de que es un asistente para crear historias y se le informa de que tiene que crear la historia por secciones siguiendo las demandas del usuario. Este primer *prompt* ha ido sufriendo numerosas modificaciones según se avanzaba con el proyecto y se eliminaban o incorporaban nuevas secciones. Finalmente, para no alargar y aumentar la complejidad debido principalmente a la limitación de los *tokens*, se optó finalmente por mantener las secciones: trama e hilo simbólico, descripción del mundo, personajes principales, desarrollo de escenarios y estructura de los capítulos. En el *prompt* para cada una de las secciones se añaden unas reglas de estructura y un ejemplo de salida para asi la IA pueda tener una mayor referencia y pueda crear la respuesta lo más parecido a la decisión de creación propuesta. Además tanto en personajes como en escenarios, se aporta una plantilla que rellena con las distintas características. También se aporta las instrucciones que tiene que seguir a la hora de interactuar con el usuario preguntando en todo momento si está conforme o quiere realizar algún cambio, y siempre hacer caso al usuario, por ejemplo, saltar a una sección y no seguir el orden preestablecido si así lo desea el usuario.

- PROMPT_JSON: Aunque se le asigne una serie de reglas y ejemplos a la IA no siempre genera la respuesta siguiendo el mismo formato. Esto dificultaba mucho a la hora de convertir el texto de la respuesta en formato *JSON* ya que se quería que se respetase cada campo generado y no fuese únicamente texto plano en un único campo. Finalmente se vio que la mejor forma de conseguirlo

era que la IA convirtiese su respuesta a texto formato *JSON* y así únicamente se tuviese que pasar de texto a *JSON*. Para ello se crea otro *promt del sistema* al que se le da toda la información y reglas que necesita para convertir de el mensaje inicial en un *string* con un formato *JSON* válido. Se ha ido ajustando y perfeccionando el *prompt* porque inicialmente en muchas ocasiones podía cambiarte el contenido o te encapsulaba el *JSON* entre comillas o símbolos que producían error a la hora de convertir el texto.

- PROMPT_ESCRITURA_RAPIDA: Este *promt* surge cuando se tiene la idea de crear otro modo de escritura donde no se tenga que crear paso a paso, si no, que se le de una idea a la IA y te genere los capítulos y la historia directamente de forma rápida. En este *pormpt* se informa de las reglas que tiene que seguir para crear la historia en este modo, indicando principalmente que tiene que generar directamente toda la historia sin tener pasos previos.

- PROMPT_CAPITULOS: Tras desarrollar todas las secciones y pasar al momento de la escritura, se ha decidido crear un nuevo *promt* más detallado de como escribir los capítulos. Así, además, sirve para gestionar mejor la limitación de *tokens* ya que si se incluyese en el mismo *prompt* este gastaría más *tokens* limitando así la respuesta. En este apartado se especifica que tiene que escribir los capítulos de uno en uno utilizando la información que se le pasa como contexto.

3.2. Interfaz de usuario

Esta sección describe la aplicación desde eñ ùnto de vista del usuario final. El objetivo es que cualquier lector —tenga o no base técnica— entienda cómo se crea una novela de principio a fin usando la herramienta, y por qué la interfaz está organizada del modo en que lo está. En concreto, se cuben:

3.2.1. Objetivo de la interfaz

3.2.2. Pantallas y funcionalidades

En este apartado se detallan las vistas principales de la aplicación y lo que el usuario puede hacer en cada una. Para mantener la consistencia, cada pantalla se describe con una mini-ficha que incluye:

- Objetivo: para qué sirve esa pantalla dentro del proceso creativo.

- Acciones clave: operaciones más habituales que el usuario puede realizar.

- Flujo breve: 3–5 pasos que resumen cómo se usa.

- Consejos: atajos o buenas prácticas para trabajar más rápido y con menos fricción.

3.2.2.1. Inicio de sesión

Objetivo. Permitir que el usuario acceda su perfil
Acciones clave.

- Iniciar sesión
- Registrar nuevo usuario

Flujo breve.

0. (Si no tienes cuenta) pulsa Register"para crear una nueva cuenta
1. Escribe nombre de usuario
2. Escribe contraseña
3. Accedes a la pantalla principal

3.2.2.2. Registro de usuario

Objetivo. Registar en la aplicacion a un nuevo usuario
Acciones clave.

- Introducir los datos del usuario
- Pulsar botón registrar

Flujo breve.

1. Introducir los datos del usuario

3.2.2.3. Página principal

Objetivo. Página principal donde el usuario observa el progreso de sus historias y puede crear una nueva.
Acciones clave.

- Cargar historia
- Eliminar historia
- Crear nueva historia en modo rápido
- Crear nueva historia en modo detallado

Flujo breve.

1. Pulsar botón de nueva historia en modo rápido o detallado
2. Introducir el nombre para guardar la historia

3.2.2.4. Nueva historia detallado

Objetivo. Página principal donde se desarrolla la historia y el usuario interactua con el modelo
Acciones clave.

- Generar sección
- Ver sección
- Reescribir sección
- Modificar sección
- Editar sección

Flujo breve.

1. Introduce una primera idea o sugerencia para comenzar a crear la historia

No hay un orden establecido, se le da libertad al usuario para interactuar como quiera con la aplicacion:

- Generar: el usuario puede pasar a la sección que considere oportuna y especificar alguna característica especial para el modelo.
- Reescribir: Si el usuario no esta conforme con lo generado por el modelo puede volver a generar la respuesta.
- Modificar: Si el usuario no esta conforme con una idea o acontecimiento puede indicar que cambio quiere realiazar y se cambie esa parte.
- Ver + editar: El usuario puede ver en la sección que desee y tiene la posibilidad de editar de forma manual.

3.2.3. 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 aplicacion.

3.2.3.1. 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.

3.2.3.2. Navegacion con flet$_{r}$*oute*

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:

Figura 3.8

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)
```

Capítulo 4

Conclusiones y Trabajo Futuro

Conclusiones del trabajo y líneas de trabajo futuro.

Antes de la entrega de actas de cada convocatoria, en el plazo que se indica en el calendario de los trabajos de fin de grado, el estudiante entregará en el Campus Virtual la versión final de la memoria en PDF.

Introduction

Introduction to the subject area. This chapter contains the translation of Chapter ??.

Conclusions and Future Work

Conclusions and future lines of work. This chapter contains the translation of Chapter ??.

Contribuciones Personales

En caso de trabajos no unipersonales, cada participante indicará en la memoria su contribución al proyecto con una extensión de al menos dos páginas por cada uno de los participantes.

En caso de trabajo unipersonal, elimina esta página en el fichero `TFGTeXiS.tex` (comenta o borra la línea `\include{Capitulos/ContribucionesPersonales}`).

Estudiante 1

Al menos dos páginas con las contribuciones del estudiante 1.

Estudiante 2

Al menos dos páginas con las contribuciones del estudiante 2. En caso de que haya más estudiantes, copia y pega una de estas secciones.

Apéndice A

Título del Apéndice A

Los apéndices son secciones al final del documento en las que se agrega texto con el objetivo de ampliar los contenidos del documento principal.

Apéndice B

Título del Apéndice B

Se pueden añadir los apéndices que se consideren oportunos.

Este texto se puede encontrar en el fichero Cascaras/fin.tex. Si deseas eliminarlo, basta con comentar la línea correspondiente al final del fichero TFGTeXiS.tex.

–¿Qué te parece desto, Sancho? – Dijo Don Quijote –
Bien podrán los encantadores quitarme la ventura,
pero el esfuerzo y el ánimo, será imposible.

Segunda parte del Ingenioso Caballero
Don Quijote de la Mancha
Miguel de Cervantes

–Buena está – dijo Sancho –; fírmela vuestra merced.
–No es menester firmarla – dijo Don Quijote–,
sino solamente poner mi rúbrica.

Primera parte del Ingenioso Caballero
Don Quijote de la Mancha
Miguel de Cervantes