UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

# Desarrollo de una herramienta de ayuda para la especificación de historias de usuario

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Zhang, Shiwan

Tutor/a: Albert Albiol, Manuela

CURSO ACADÉMICO: 2023/2024

# Resum

Encara que els models de llenguatge de gran tamany(LLMs), com GPT-4, tenen capacitats impressionants en generació i raonament, tenen limitacions en termes de la seua capacitat per accedir i recuperar fets, xifres o informació contextualment rellevant específics. Una solució popular a este problema és configurar un sistema de generació augmentada de recuperació(RAG): combinar el model de llenguatge amb un proveïdor d'emmagatzematge extern i crear un sistema de programari general que puga orquestrar les interaccions entre estos components per crear una experiència de "xat amb les teues dades".

En aquest projecte, es centra en millorar l'elaboració i gestió d'històries d'usuari dins del context de la metodologia àgil, utilitzant models de llenguatge de gran mida (LLMs) i aprofitant les històries existents de l'ecosistema Inditex. Es desenvolupa específicament una eina avançada destinada a ajudar els Product Owners en la definició precisa de les seues històries d'usuari. Aquesta eina no sols identifica duplicats, sinó que també facilita la creació de noves històries i la generació d'escenaris rellevants a partir d'històries similars. A més, com que està fonamentada en LLMs, l'eina té la capacitat de generar termes de cerca a partir de la història d'usuari proporcionada, executar cerques per recopilar informació rellevant i després integrar-la en la història resultant, optimitzant significativament la gestió de requisits en projectes àgils.

**Paraules clau:** Històries d'usuari, Jira, Bases de dades Vectorials, models de llenguatge de gran tamany(LLMs), generació augmentada de recuperació(RAG), metodologia àgil.

# Resumen

Si bien los modelos de lenguaje de gran tamaño(LLMs), como GPT-4, tienen capacidades impresionantes en generación y razonamiento, tienen limitaciones en términos de su capacidad para acceder y recuperar hechos, cifras o información contextualmente relevante específicos. Una solución popular a este problema es configurar un sistema generación aumentada de recuperación(RAG): combinar el modelo de lenguaje con un proveedor de almacenamiento externo y crear un sistema de software general que pueda orquestar las interacciones entre estos componentes para crear una experiencia de "chat con tus datos".

En este proyecto, se enfoca en mejorar la elaboración y gestión de historias de usuario dentro del contexto de la metodología ágil, utilizando modelos de lenguaje de gran tamaño(LLMs) y aprovechando las historias existentes del ecosistema Inditex. Específicamente, se desarrolla una herramienta avanzada destinada a asistir a los Product Owners en la definición precisa de sus historias de usuario. Esta herramienta no solo identifica duplicados, sino que también facilita la creación de nuevas historias y la generación de escenarios relevantes a partir de historias similares. Además, al estar fundamentada en LLMs, la herramienta tiene la capacidad de generar términos de búsqueda a partir de la historia de usuario proporcionada, ejecutar búsquedas para recopilar información relevante y luego integrarla en la historia resultante, optimizando significativamente la gestión de requisitos en proyectos ágiles.

**Palabras clave:** Historias de usuario, Jira, Bases de datos Vectoriales, modelos de lenguaje de gran tamaño(LLMs), generación aumentada de recuperación(RAG), metodología ágil.

# Abstract

While large language models(LLMs) like GPT-4 have impressive capabilities in generation and reasoning, they have limitations in terms of their ability to access and retrieve specific facts, figures, or contextually relevant information. A popular solution to this problem is setting up a retrieval-augmented generation(RAG) system: combine the language model with an external storage provider, and create an overall software system that can orchestrate the interactions with and between these components in order to create a "chat with your data" experience.

In this project, the focus is on improving the elaboration and management of user stories within the context of agile methodology, utilizing Large Language Models (LLMs) and leveraging existing stories from the Inditex ecosystem. Specifically, an advanced tool is being developed to assist Product Owners in precisely defining their user stories. This tool not only identifies duplicates but also facilitates the creation of new stories and the generation of relevant scenarios from similar stories. Additionally, being grounded in LLMs, the tool has the capability to generate search terms from the provided user story, conduct searches to gather relevant information, and then integrate it into the resulting story, significantly optimizing requirement management in agile projects.

**Key words:** User stories, Jira, Vector databases, large language models(LLMs), retrieval-augmented generation(RAG), agile methodology.

# Índice general

**Índice general** VII
**Índice de figuras** IX
**Índice de tablas** X

**1 Introducción** 1
1.1 Motivación . . . 1
1.2 Objetivos . . . 2
1.3 Metodología . . . 3
1.4 Estructura de la memoria . . . 4
**2 Estado del arte** 5
2.1 Modelos de Lenguaje de Gran Tamaño(LLMs) . . . 5
2.1.1 Aplicación de LLM en la generación de historias de usuario . . . 5
2.2 El Modelo i* para la generación automática de historias de usuario . . . 6
2.3 Historias de Usuario en Inditex . . . 6
2.3.1 Aplicación de tecnologías innovadoras en Inditex . . . 7
2.3.2 Relevancia del proyecto: necesidad de herramientas de gestión de historias de usuario . . . 7
2.4 Análisis de la situación actual y solución propuesta . . . 7
**3 Análisis del problema** 9
3.1 Identificación de necesidades . . . 9
3.2 Análisis de soluciones posibles . . . 10
3.2.1 Sistemas de generación aumentada de recuperación(RAG) . . . 10
3.2.2 Extensión de Google . . . 11
3.3 Solución propuesta . . . 12
3.3.1 Desarrollo . . . 12
3.3.2 Implantación y Validación . . . 13
**4 Diseño de la solución** 14
4.1 Arquitectura del sistema . . . 14
4.1.1 Patrones de diseño . . . 15
4.2 Tecnologías utilizadas . . . 17
4.2.1 Herramientas . . . 17
4.2.2 Frontend . . . 20
4.2.3 Backend . . . 21
**5 Desarrollo de la solución propuesta** 26
5.1 Backend . . . 26
5.1.1 Arquitectura de Backend . . . 26
5.1.2 Gestión de la base de datos . . . 27
5.1.3 Lógica de negocio . . . 28
5.1.4 APIs y servicios externos . . . 30
5.2 Frontend . . . 31
5.2.1 Arquitectura del Frontend . . . 31
5.2.2 Diseño de interfaz de usuario . . . 32

5.2.3 Interactividad y funcionalidades . . . . . 33
5.2.4 Optimización . . . . . 34
**6 Implantación** **36**
6.1 Despliege en Kubernetes . . . . . 36
**7 Pruebas** **38**
7.1 Pruebas de integración . . . . . 38
7.1.1 Diseño de casos de prueba . . . . . 38
7.1.2 Resultados de prueba de API . . . . . 39
7.2 Pruebas de usabilidad . . . . . 40
7.2.1 Objetivos de usabilidad . . . . . 41
7.2.2 Planificación de tareas y Análisis de resultados . . . . . 41
**8 Conclusiones** **46**
8.1 Cumplimiento de los objetivos . . . . . 46
8.2 Trabajo futuro . . . . . 46
8.3 Relación con las asignaturas . . . . . 47

Apéndice

**A Reflexión sobre los objetivos ODS** **53**

# Índice de figuras


1.1 Historia de usuario en Jira . . . . . 2
1.2 Sistema generación aumentada de recuperación(RAG) . . . . . 3
1.3 Tablero Kanban en Jira . . . . . 4

3.1 Ejemplo de Chrome extensión . . . . . 11

4.1 Arquitectura del sistema . . . . . 14
4.2 Estructura de procesar datos . . . . . 15
4.3 Ejemplo de estructura hexagonal . . . . . 17
4.4 Proceso de Embeddings . . . . . 24

5.1 Estructura hexagonal en el proyecto . . . . . 26
5.2 Conexión con weaviate . . . . . 27
5.3 Implementación del método detectDuplicate . . . . . 27
5.4 Ejecución en paralelo realizada con Java . . . . . 28
5.5 prompt generado con promptPerfect . . . . . 29
5.6 prompt para generar frases de búsqueda . . . . . 29
5.7 API del backend . . . . . 30
5.8 resultados de la API de búsqueda . . . . . 31
5.9 Implementación de manifest.json . . . . . 32
5.10 Interfaz de la Extensión con Figma . . . . . 32
5.11 Resultado de detectar duplicaciones . . . . . 33
5.12 Resultado de enriquecer historia de usuario . . . . . 34
5.13 Estado de cargar . . . . . 34
5.14 Código del botón de detectar duplicaciones . . . . . 35

7.1 Resultado correcto de prueba con Postman . . . . . 40
7.2 Código modificado de API de enriquecer historia de usuario . . . . . 40
7.3 Pregunta 1 - Facilidad de uso . . . . . 42
7.4 Pregunta 2 - Satisfacción de usuario . . . . . 43
7.5 Pregunta 3 - Eficiencia de la interfaz . . . . . 43
7.6 Pregunta 4 - Capacidad de Respuesta . . . . . 44
7.7 Pregunta 5 - Sugerencia . . . . . 44

8.1 Mejora de la interfaz . . . . . 47


# Índice de tablas

3.1 Plantilla de Historia de Usuario Ágil . . . . . 10

7.1 Casos de prueba para la API . . . . . 39

A.1 Objetivos de Desarrollo Sostenible . . . . . 53

# CAPÍTULO 1
# Introducción

En este primer capítulo se presenta una breve explicación del proyecto y se describe la motivación, objetivos, metodología del proyecto y estructura del documento.

## 1.1 Motivación

En el desarrollo de software, la gestión eficaz de requisitos es fundamental para el éxito de cualquier proyecto. Una metodología de desarrollo de software es un marco de trabajo que se usa para estructurar, planificar y controlar el proceso de desarrollo de sistemas de información [27].

Las metodologías ágiles han adquirido una posición predominante en el desarrollo de software contemporáneo, siendo ampliamente adoptados tanto por pequeñas como por grandes organizaciones[43]. Además, han revolucionado la forma en que se abordan los requisitos, priorizando la colaboración continua con los clientes y adaptándose a los cambios en los requisitos a lo largo del ciclo de desarrollo.

Las historias de usuario se usan, en el contexto de la ingeniería de requisitos ágil, como una herramienta de comunicación que combina las fortalezas de ambos medios: escrito y verbal. Describen, en una o dos frases, una funcionalidad de software desde el punto de vista del usuario, con el lenguaje que éste emplearía. El foco está puesto en qué necesidades o problemas soluciona lo que se va a construir[29].

Sin embargo, incluso con los principios ágiles, la elaboración y gestión de historias de usuario sigue siendo un desafío significativo. Los Product Owners, como representantes del cliente o usuario final en un equipo ágil de desarrollo de software, gastan una gran parte de su tiempo en escribir historias de usuario claras que comuniquen las expectativas completas del producto a desarrollar.

En el ecosistema de Inditex, los Product Owners(PO) utilizan Jira para crear historias de usuario(Figura 1.1). Sin embargo, se enfrentan a varios desafíos al hacerlo.

1. Duplicidades: Es posible que una historia de usuario ya haya sido creada previamente, pero Jira carece de funcionalidades para detectar automáticamente historias duplicadas. Este aspecto puede resultar especialmente relevante al incorporarse a un proyecto totalmente desconocido.

2. Tareas repetitivas: En muchas ocasiones, los PO deben completar descripciones y criterios de validación de cada historia de usuario de forma repetitiva y trivial debido a la gran cantidad de historias similares.

3. Falta de soporte: Actualmente no existe una herramienta específica para ayudar a los PO a mejorar la eficiencia en la creación de historias de usuario. En muchos casos, los PO deben buscar información adicional en Internet, ya sea a través de motores de búsqueda como Google o mediante el uso de herramientas como ChatGPT, para enriquecer las historias de usuario.

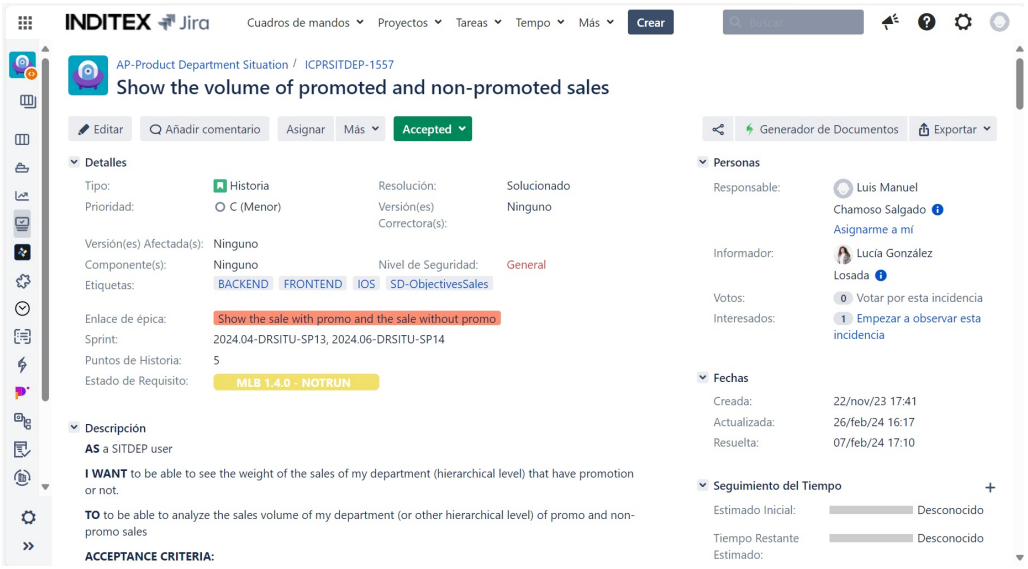


**Figura 1.1:** Historia de usuario en Jira

## 1.2 Objetivos

El principal objetivo de este TFG es el diseño y desarrollo de una herramienta que facilite a los PO la definición de historias de usuario. Esta herramienta debe funcionar como una extensión de Google usando Java para backend y TypeScript para frontend. La herramienta proporcionará una interfaz única que permitirá enriquecer las historias de usuario, aprovechando tanto las historias existentes como la información relevante relacionada, incluyendo su origen de URL.

Este sistema está basado en un sistema generación aumentada de recuperación(RAG) que es una técnica ampliamente aplicada para mejorar el rendimiento de los modelos de lenguaje de gran tamaño(LLM) en tareas de generación intensiva de conocimiento, como la respuesta a preguntas basadas en documentos. Dada una pregunta, la técnica(Figura 1.2) incluye el uso de un modelo de recuperación para obtener múltiples párrafos relevantes de posiblemente diferentes documentos, luego se ingresan estos pasajes a un modelo lector como contextos adicionales para generar una respuesta[21].

Esta herramienta también tiene varios subobjetivos tanto para los Product Owners como para los desarrolladores:

- **Facilitar la detección de historias duplicadas:** Cuando un Product Owner recién llegado a una empresa quiere crear una nueva historia de usuario, puede encontrarse en la situación de no estar seguro si esa historia ya existe. Dado que hay una gran cantidad de historias de usuario existentes, sería beneficioso contar con una herramienta que permita detectar duplicados de la nueva historia en comparación con las ya existentes, evitando así la duplicación de esfuerzos.
- **Simplificar el trabajo de Product Owner:** Se propone desarrollar una herramienta capaz de generar automáticamente historias de usuario a partir de los resúmenes

proporcionados, de modo que el Product Owner solo necesite revisarlas y realizar modificaciones en pocos lugares. Con esto, se busca ahorrar aún más tiempo en la redacción, permitiendo al Product Owner dedicar el tiempo ahorrado a aprender más sobre el tema y comunicarse con los interesados para clarificar los requisitos.

- **Ayudar a los desarrolladores a entender mejor las historias de usuario:** En muchas situaciones, los desarrolladores reciben historias de usuario confusas que no especifican claramente qué se pide. Por lo tanto, un subobjetivo de gran importancia es proporcionarles una comprensión más clara mediante la implementación de historias de usuario completas, asegurando así que los requisitos del cliente se cumplan adecuadamente.

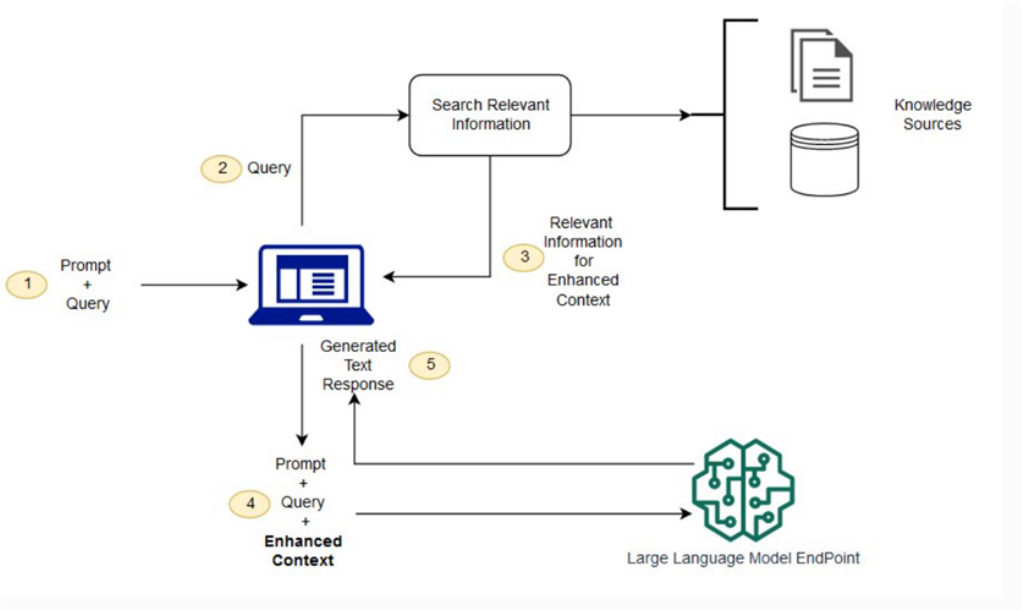


**Figura 1.2:** Sistema generación aumentada de recuperación(RAG)

## 1.3 Metodología

El desarrollo de software se beneficia enormemente de metodologías que estructuran y organizan las tareas de manera efectiva. Entre las corrientes más destacadas se encuentran las metodologías tradicionales y ágiles, cada una con sus ventajas e inconvenientes.

Las metodologías tradicionales son adecuadas para proyectos en los que existe un alto grado de experiencia o se desarrollan en entornos predecibles y estables, caracterizados por una estructura más rígida e inflexible. Sin embargo, en proyectos donde hay muchos cambios, donde los requisitos son susceptibles de cambios, donde rehacer partes del código no es una actividad costosa, los equipos son pequeños, las fechas de entrega del software son cortas y el desarrollo rápido es fundamental, no puede haber requisitos estáticos, por lo que se requieren metodologías ágiles.

El presente proyecto software se desarrollará conforme la metodología ágil Kanban[8], dado que los requisitos del proyecto son propensos a evolucionar a lo largo del tiempo y el proyecto tiene plazos de entrega cortos y se requiere un desarrollo rápido e incremental. Además, es relevante destacar que Kanban no es una técnica de planificación sino de organización de las tareas.

La división de tareas será fundamental para evaluar el progreso y los objetivos alcanzados a lo largo de todo el proceso de desarrollo. Para gestionar y dividir estas ta-

reas, se empleará la herramienta Jira, lo que facilitará enormemente la gestión de proyectos(Figura 1.3).

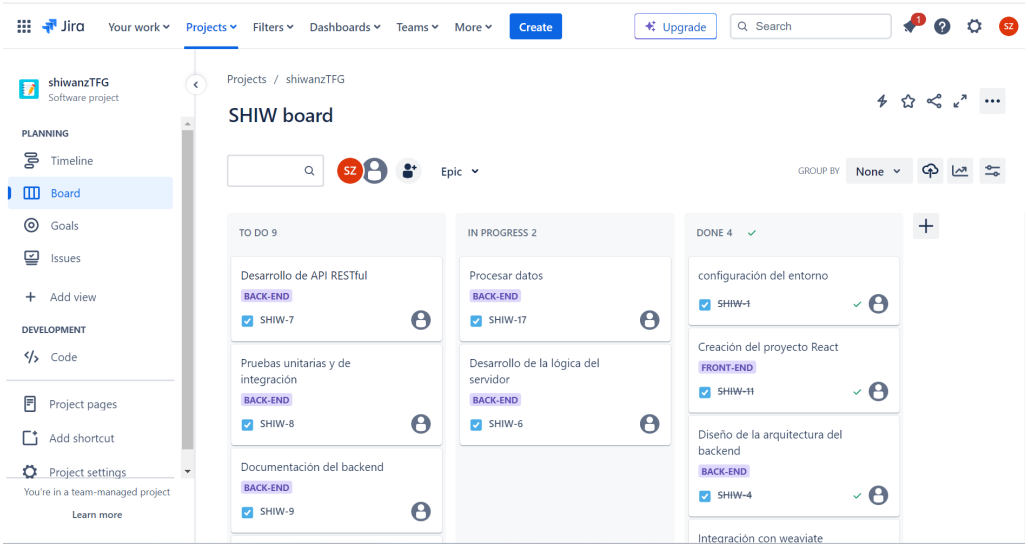


**Figura 1.3:** Tablero Kanban en Jira

## 1.4 Estructura de la memoria

El presente documento se divide en varios capítulos, los cuales se presentan y desarrollan de la siguiente forma:

- **Estado del arte:** Se revisarían las herramientas y tecnologías existentes utilizadas en la generación de historias de usuario, y se identificarían sus limitaciones o áreas donde podrían mejorarse. Además, dado que este Trabajo de Fin de Grado se desarrolla en colaboración con Inditex, se incluirá una presentación de la empresa, su historia y la aplicación de tecnologías en su operativa, así como la justificación de la necesidad de esta herramienta dentro de la organización.
- **Análisis del problema:** Analizar los posibles problemas y definir brevemente la solución propuesta y cómo será desarrollada.
- **Diseño de la solución.** Describe a nivel técnico de la arquitectura del proyecto y las tecnologías y herramientas utilizadas para poder llevarlo a cabo.
- **Desarrollo de la solución:** La exposición sobre la implementación se divide en dos partes: backend y frontend. Además, este apartado incluye descripciones detalladas de la API y del uso de servicios externos.
- **Implantación:** Guía de la publicación del proyecto en Google Extensión, para poder ser accedida de forma pública desde cualquier dispositivo.
- **Pruebas:** Descripción de las pruebas realizadas para este proyecto, incluyendo los casos de prueba específicos y los resultados obtenidos.
- **Conclusiones y trabajo futuro:** Resumen de las conclusiones del proyecto, evaluación de los objetivos alcanzados y descripción de posibles áreas de mejora para futuros trabajos.

# CAPÍTULO 2
# Estado del arte

En este capítulo, se explora los avances recientes para mejorar la eficiencia en la generación de historias de usuario. Además, se analizan sus ventajas y limitaciones para ofrecer una visión más completa de su impacto práctico.

Este capítulo también presentará el contexto de la empresa Inditex y cómo utiliza tecnologías avanzadas en la industria de la moda. Además, se explicará por qué esta compañía necesita crear una herramienta que acelere la eficiencia en la generación de historias de usuario.

## 2.1 Modelos de Lenguaje de Gran Tamaño(LLMs)

Los Modelos de Lenguaje de Gran Tamaño(LLM, por sus siglas en inglés) han surgido como herramientas sumamente poderosas, capaces de procesar y generar lenguaje de forma autónoma[13].

En particular, los LLMs son modelos de procesamiento de lenguaje natural que, en la actualidad, adoptan ampliamente la arquitectura de transformadores[1] y, específicamente, la de transformadores generativos preentrenados o "GPT"(por sus siglas en inglés)[41]. Estos operan con una gran cantidad de parámetros y han sido entrenados en una amplia variedad de datos de texto para ejecutar diversas tareas. En la actualidad, es comúnmente aceptado considerar a ChatGPT[2] como un LLM debido a su habilidad para generar respuestas coherentes en conversaciones. Sin embargo, es importante destacar que ChatGPT no es en sí mismo un LLM, sino más bien una aplicación web que facilita la interacción con un LLM subyacente, tal como el modelo GPT-3 de OpenAI. Este último es el encargado de procesar las conversaciones utilizando una variedad de técnicas.

### 2.1.1. Aplicación de LLM en la generación de historias de usuario

En la actualidad, los modelos de lenguaje para automatizar procesos de ingeniería de software y actividades de desarrollo está volviéndose extremadamente popular y evolucionando rápidamente tanto en el ámbito académico como en la industria.

Al mismo tiempo, se dispone de un método novedoso e innovador impulsado por inteligencia artificial, que aprovecha las avanzadas capacidades del modelo de lenguaje GPT-3.5. Este enfoque facilita una transformación fluida y eficiente del texto de los

[1] Los transformadores son un tipo de arquitectura de red neuronal que transforma o cambia una secuencia de entrada en una secuencia de salida.(https://aws.amazon.com/es/what-is/transformers-in-artificial-intelligence/)

[2] https://chatgpt.com/

requisitos de software en historias de usuario estandarizadas, mediante el análisis de diversas técnicas de generación de estímulos[36]. Por tanto, muchos Produt Owners están utilizando GPT para ayudar la generación de historias de usuario.

Sin embargo, en el ámbito de las historias de usuario, es importante tener en cuenta que GPT no tiene la capacidad de detectar datos privados de las empresas. Por lo tanto, al utilizar esta tecnología, las respuestas generadas se basarán únicamente en los requisitos y detalles proporcionados en el texto actual, sin considerar contextos anteriores. Como resultado, es posible que la historia generada no sea precisa, lo que requeriría un esfuerzo adicional para su modificación y ajuste.

## 2.2 El Modelo i* para la generación automática de historias de usuario

En esta sección, se introduce el modelo i* y su aplicación en la generación automática de historias de usuario. El marco i* es un lenguaje de modelado utilizado en la ingeniería de requisitos, que se centra en las relaciones entre los diversos actores del sistema y sus objetivos. Este método pone especial énfasis en partir de los objetivos de negocio, descomponiéndolos y analizando sus interrelaciones para transformarlos automáticamente en historias de usuario, asegurando así su integridad y coherencia. Su principal ventaja radica en la mejora de la precisión en la definición de requisitos y la reducción del riesgo de pérdida de información[19].

Aunque el método de generación automática de historias de usuario basado en el modelo i* ofrece ventajas significativas, también presenta desafíos como una curva de aprendizaje empinada, dependencia de datos de alta calidad, problemas de flexibilidad, dificultades de integración y costos asociados. Estas limitaciones sugieren que, mientras el método es efectivo en contextos adecuados, es crucial evaluar su viabilidad en función del contexto específico y los recursos disponibles de la organización.

## 2.3 Historias de Usuario en Inditex

Inditex, cuyo nombre completo es Industrias de Diseño Textil Sociedad Anónima, fue fundado en 1975 por Amancio Ortega Gaona en Arteixo, Galicia, España. La empresa comenzó su actividad con la apertura de la primera tienda Zara en La Coruña, un enfoque que revolucionó el concepto de moda al ofrecer diseño, producción y distribución rápida a través de su modelo de negocio de "fast fashion".

Desde entonces, Inditex ha crecido hasta convertirse en uno de los mayores conglomerados de moda del mundo. El grupo posee varias marcas, incluyendo Zara, Pull&Bear, Massimo Dutti, Bershka, Stradivarius, Oysho y Zara Home, cada una con su propio estilo y segmento de mercado. Inditex opera más de 7,000 tiendas en 93 mercados y tiene una fuerte presencia en línea en más de 200 regiones a través de sus diversas plataformas digitales[22].

La visión de Inditex se centra en el cliente, la innovación constante, y la sostenibilidad. La empresa ha sido pionera en integrar la tecnología en todas las fases de su cadena de valor, desde el diseño hasta la venta, y se compromete a hacer que su modelo de negocio sea compatible con los límites ambientales del planeta. Este contexto empresarial no solo subraya la importancia de Inditex en el sector global de la moda, sino que también establece el marco para la necesidad de innovaciones tecnológicas, como la herramienta

propuesta en este proyecto para ayudar a especificar historias de usuario, alineada con los valores de eficiencia e innovación de la empresa.

### 2.3.1. Aplicación de tecnologías innovadoras en Inditex

Inditex ha destacado en la industria de la moda por su capacidad para integrar tecnologías innovadoras en su modelo de negocio. Esta integración tecnológica es visible en varios aspectos clave de la empresa, desde la cadena de suministro hasta la experiencia del cliente.

- **Cadena de suministro optimizada:** Inditex utiliza sistemas avanzados de gestión de inventario y logística que permiten una producción y distribución extremadamente ágil. La tecnología RFID(identificación por radiofrecuencia) juega un papel crucial, permitiendo un seguimiento preciso de cada artículo desde el centro de producción hasta la tienda, asegurando que los productos correctos estén disponibles en el momento y lugar adecuados[42].

- **Experiencia de compra personalizada:** Inditex ha invertido significativamente en el desarrollo de plataformas digitales que mejoran la experiencia de compra en línea y en tienda. Utilizan análisis de datos para personalizar las recomendaciones de productos y optimizar los niveles de stock en tiempo real. Además, la implementación de tecnologías de realidad aumentada en más de 120 tiendas mundiales permite a los clientes visualizar cómo les quedarían ciertas prendas sin necesidad de probárselas físicamente[47].

### 2.3.2. Relevancia del proyecto: necesidad de herramientas de gestión de historias de usuario

El vitalidad y la competitividad del sector de la moda requieren que empresas como Inditex no solo mantengan, sino que también innoven continuamente sus procesos para mejorar la eficiencia y la efectividad. En este entorno, la gestión ágil de proyectos es imprescindible, siendo las historias de usuario una herramienta fundamental en el desarrollo de software y productos que respondan efectivamente a las necesidades del cliente.

En Inditex, las historias de usuario se han documentado utilizando Jira. Estas narrativas breves y claras permiten a los desarrolladores centrarse en las necesidades reales de los usuarios, asegurando que las características del producto final estén alineadas con las expectativas del mercado. Sin embargo, la generación de historias de usuario enfrenta desafíos como la duplicidad, tareas repetitivas y falta de soporte. Esto puede llevar a ineficiencias y retrasos en el lanzamiento de nuevos productos.

## 2.4 Análisis de la situación actual y solución propuesta

Según este capítulo, es evidente que las herramientas y métodos existentes para ayudar en la generación de historias de usuario tienen sus propias desventajas y limitaciones. Por otro lado, aún no se ha identificado una herramienta específica diseñada para asistir a los Product Owners en la redacción de historias de usuario, lo que puede provocar una disminución en la eficiencia durante la creación de estas historias.

En este TFG se abordará el desarrollo de una herramienta destinada a mejorar la especificación de historias de usuario basándose en las historias existentes dentro de la

metodología ágil en Inditex. El desafío a enfrentar es proporcionar asistencia a los Product Owners en la creación y optimización de historias de usuario, lo que facilitará un proceso de desarrollo más eficiente y conducirá a obtener resultados alineados con las necesidades de los interesados. Además, esta herramienta estará integrada con las historias de usuario en Jira para su utilización interna en Inditex.

# CAPÍTULO 3
# Análisis del problema

En este capítulo, se exploran tanto las necesidades fundamentales que este TFG busca abordar como las posibles soluciones identificadas para hacerlo de manera efectiva. La comprensión precisa de estas necesidades es crucial para el éxito del proyecto, ya que proporciona la base sobre la cual se desarrollará la herramienta propuesta. Se enfoca especialmente en la especificación de los requisitos de la herramienta a desarrollar, utilizando las historias de usuario como un método efectivo para comprender las funcionalidades necesarias desde la perspectiva del usuario final.

En respuesta a las necesidades identificadas, se han explorado diversas soluciones potenciales que podrían abordar de manera efectiva los desafíos planteados. Estas soluciones van desde la adopción de metodologías innovadoras hasta el desarrollo de herramientas específicas. Finalmente, se presentará una solución que detalla cómo combinar y ejecutar paso a paso las mejores prácticas de ambas soluciones para lograr una respuesta completa y efectiva a los desafíos identificados.

## 3.1 Identificación de necesidades

Para cumplir con todos los objetivos y subobjetivos mencionados en el capítulo 1.2, se han identificado una serie de necesidades fundamentales que desean integrar en la herramienta propuesta. Estas necesidades reflejan los desafíos y las áreas de mejora que enfrentan en sus procesos de desarrollo de software. Para abordar estas necesidades de manera efectiva, se ha optado por utilizar una metodología ágil, centrada en el usuario y basada en historias de usuario.

En la tabla 3.1, se presenta una plantilla de Historia de Usuario Ágil que resume las principales necesidades identificadas, junto con los objetivos específicos que se espera alcanzar con la implementación de cada funcionalidad. Estas historias de usuario proporcionan una visión general de las características clave que se deben incluir en la herramienta, y servirán como guía durante todo el proceso de desarrollo para garantizar que se cumplan las expectativas de los usuarios finales.

A través de este análisis detallado de las necesidades y la definición de historias de usuario, se espera establecer una base sólida para el desarrollo del TFG, asegurando que la herramienta resultante satisfaga las demandas del mercado y ofrezca un valor añadido significativo a sus usuarios.

| ID | Como *tipo de usuario* | Quiero *realizar alguna tarea* | Para que pueda *el logro algún objetivo* |
|---|---|---|---|
| 1 Enriquecer historia de usuario | Product Owner | Refinar mi historia de usuario utilizando los ejemplos relevantes de historias de usuario disponibles en nuestra base de datos. | Presentar la historia de usuario finalizada que sea coherente y cumpla con los requisitos establecidos en proyectos anteriores. |
| 2 Generar escenarios y criterios de aceptación | Product Owner | Utilizar una herramienta o sistema que genere automáticamente los escenarios de prueba y criterios de aceptación basados en la estructura y requisitos de mi historia de usuario. | Aumentar la eficiencia y consistencia en la creación de nuevas historias de usuario y garantizar una mejor cobertura de pruebas. |
| 3 Detectar duplicaciones | Product Owner | Esta herramienta verifica automáticamente si una historia de usuario ya existe en la base de datos antes de permitir su creación. | Evitar duplicados y asegurar que todas las historias de usuario sean únicas y necesarias. |
| 4 Proporcionar información relevante | Product Owner | La historia de usuario generada considera información externa relacionada con el tema de la historia y proporciona enlaces para acceder a dicha información. | Entender profundamente las necesidades del cliente y asegurar que la solución propuesta sea relevante y de valor añadido. |

**Tabla 3.1:** Plantilla de Historia de Usuario Ágil

## 3.2 Análisis de soluciones posibles

### 3.2.1. Sistemas de generación aumentada de recuperación(RAG)

Los modelos de lenguaje de gran tamaño(LLM) tienen limitaciones para acceder y manipular el conocimiento de manera precisa. Por lo tanto, en tareas que requieren mucho conocimiento, su rendimiento es inferior al de las arquitecturas específicas diseñadas para esas tareas[26].

La Generación con Recuperación Aumentada(RAG) ha surgido recientemente como un método para ampliar más allá del conocimiento preentrenado de los Modelos de Lenguaje de Gran Tamaño mediante la ampliación de la consulta original con pasajes o documentos relevantes recuperados por un sistema de Recuperación de Información. RAG se ha vuelto cada vez más importante para las soluciones de IA Generativa, especialmente en entornos empresariales o en cualquier dominio en el que el conocimiento se actualice constantemente y no pueda ser memorizado en el LLM[12].

Por lo tanto, se ha decidido crear una herramienta basada en el modelo RAG para integrar las historias de usuario previamente creadas con el nuevo requisito. Esto permitirá retroalimentar la generación de la nueva historia mientras se detecta la posible

duplicación entre ellas. Además, gracias al uso de LLM, esta herramienta podrá generar automáticamente los escenarios de prueba y criterios de aceptación basados en la estructura y requisitos de la historia de usuario.

Para mejorar la precisión y la satisfacción del usuario final, la herramienta va a realizar búsqueda en Google para obtener más información relacionada con la historia de usuario, como investigaciones de mercado, retroalimentación de clientes y análisis competitivos. Esto permitirá al product owner obtener un mayor conocimiento sobre el tema y facilitará la modificación de la historia de usuario y la comunicación con los interesados involucrados.

### 3.2.2. Extensión de Google

Las extensiones de Google Chrome(Figura 3.1) son programas que pueden instalarse en Chrome para cambiar la funcionalidad del navegador[25]. La extensión puede proporcionar nuevas funcionalidades al combinar características existentes del navegador web y permitir que los usuarios realicen múltiples tareas simultáneamente[28].

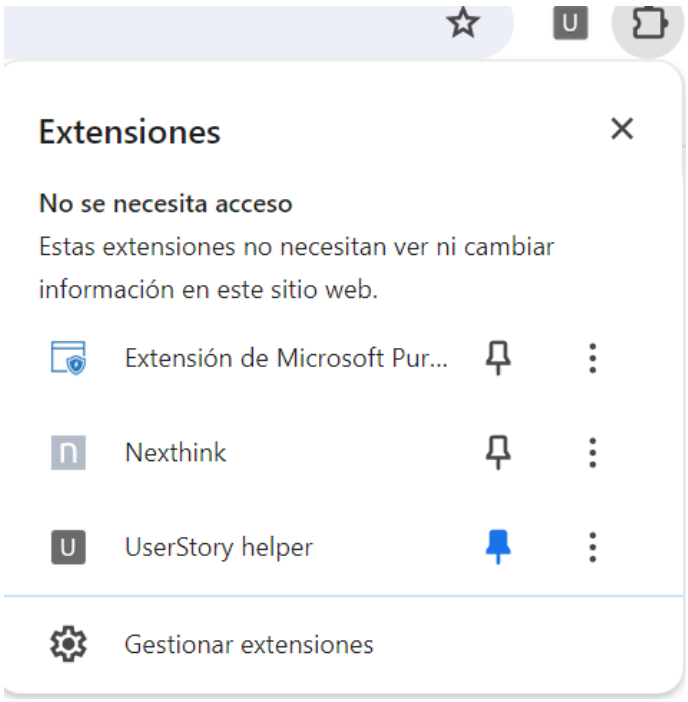


**Figura 3.1:** Ejemplo de Chrome extensión

La extensión de Google Chrome se presenta como el producto final ideal en este proyecto por diversas razones:

- La extensión de Google Chrome se instala dentro del navegador Chrome, lo que tiene un impacto mínimo en el uso de los recursos del sistema informático. Chrome es un navegador muy popular en todo el mundo y los usuarios pueden acceder a él fácilmente.
- La mayoría de los Product Owners en Inditex escriben historias de usuario a través de Jira en los navegadores, lo que significa que la función integrada del navegador, como las extensiones, será la forma más directa y efectiva de generar historias de usuario.

- La extensión será de tamaño pequeño y fácil de instalar dentro del navegador. La extensión de Google Chrome es segura y fácil de usar porque solo necesita un clic para instalarla. Cuando los usuarios deseen detenerla, solo necesitan un clic para desactivar la extensión[11].

- Publicar la extensión en otros navegadores como Edge, Firefox o Safari podría ser más difícil debido a que el proceso de revisión en cada plataforma puede prolongar el desarrollo y retrasar la disponibilidad para los usuarios. Por tanto, enfocarse en un único navegador como Google Chrome permite una implementación más rápida y eficiente, aprovechando su amplia base de usuarios, lo que puede influir en la decisión de priorizar el desarrollo para esta plataforma.

En conclusión, basándonos en las ventajas y los desafíos identificados, se ha decidido desarrollar una extensión de Google para simplificar el uso de la herramienta. Esta extensión permitirá interactuar con las historias editadas en la plataforma web de Jira y devolver los resultados directamente a dicha página web. Como resultado, se reducirá significativamente el tiempo necesario para instalar y aprender a utilizar la herramienta, ya que su uso será fácil y accesible.

## 3.3 Solución propuesta

Después de analizar las dos posibles soluciones, se ha optado por desarrollar una extensión de Chrome basada en el método RAG. Esta elección se basa en nuestra intención de mejorar la interacción entre el usuario y el sistema, lo que facilitará el proceso de la edición de historias de usuario y su integración con la plataforma Jira.

Se va a dividir el TFG en tareas para poder medir el progreso de manera más precisa. Una vez que todas las tareas y sub-tareas estén listas en Jira, se comenzará el desarrollo por la tarea más prioritaria, es decir, aquellas tareas que tengan dependencias en otras tareas.

Una vez se ha realizado el análisis, las etapas restantes son dos: desarrollo e implantación y validación. A continuación, se identifican las fases de cada una de ellas.

### 3.3.1. Desarrollo

a. **Diseño de la Interfaz:** Se iniciará con el diseño de la interfaz de usuario para la extensión. Este proceso incluirá la creación de mockups que serán validados mediante pruebas de usabilidad para asegurar que la interfaz sea clara y cumpla con las expectativas de los usuarios finales.

b. **Desarrollo del Backend:** Paralelamente al diseño de la interfaz, se desarrollará el backend y una API RESTful[1]. Este componente será responsable de la lógica de negocio y la gestión de datos, asegurando que la extensión sea escalable y segura.

c. **Implementación del Frontend:** Una vez completado el backend, se procederá con la implementación del frontend. Durante esta fase, se integrará la interfaz de usuario con el backend a través de la API RESTful desarrollada, permitiendo una comunicación fluida y eficiente entre el frontend y el backend.

[1]La API RESTful es una interfaz que dos sistemas de computación utilizan para intercambiar información de manera segura a través de Internet

### 3.3.2. Implantación y Validación

a. **Pruebas de Integración:** Se realizarán pruebas de integración para asegurar que todos los componentes del sistema funcionen correctamente en conjunto. Estas pruebas ayudarán a identificar y corregir problemas de integración entre el frontend y el backend.

b. **Pruebas de Usabilidad:** Antes del lanzamiento oficial, se llevarán a cabo pruebas de usabilidad con usuarios finales para validar la interfaz y la experiencia general de uso del software.

c. **Despliegue:** Finalmente, se procederá con el despliegue de la solución. Este incluirá la configuración del entorno de producción y la migración de datos necesaria para la puesta en marcha del sistema.

# CAPÍTULO 4
# Diseño de la solución

En este capítulo se describe el diseño de la solución, que incluye una breve introducción a la arquitectura del sistema, seguida de una explicación más detallada sobre las relaciones dentro de dicha arquitectura.

También se detallarán las herramientas y tecnologías utilizadas en la realización del presente TFG, abarcando las aplicaciones y lenguajes de programación utilizados a lo largo de todo el proceso del trabajo.

## 4.1 Arquitectura del sistema

La arquitectura de este proyecto se conoce como la arquitectura cliente-servidor, los Product Owners envían una petición a través de una extensión web, que a su vez invoca la API correspondiente para comunicarse con el servidor.

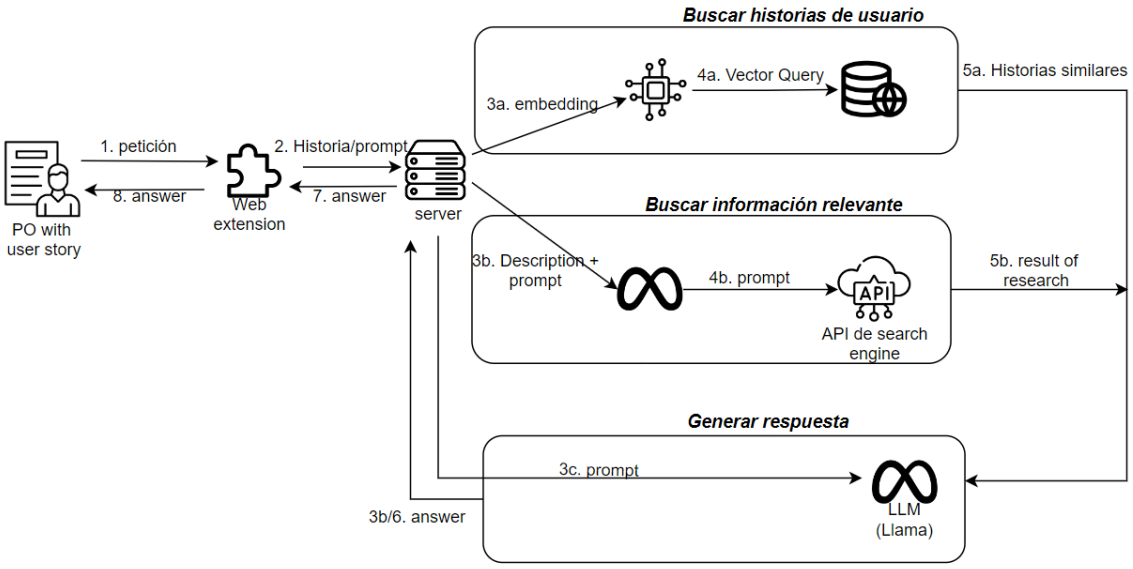


**Figura 4.1:** Arquitectura del sistema

Como se puede ver en la figura 4.1, el servidor en general va a realizar 3 partes de funcionalidades.

1. **Buscar historias de usuario:** Este proceso está diseñado para ayudar a identificar historias duplicadas y enriquecer historias de usuario a partir de aquellas similares. Inicialmente se transforma una breve historia de usuario proporcionada por

los Product Owners en un formato de embedding[1]. Posteriormente, se envía este resultado a una base de datos vectorial[2] para identificar historias similares.

2. **Buscar información relevante:** Esta etapa está diseñada para proporcionar información relevante acerca de la historia de usuario. Primero se pasa la descripción de la historia de usuario junto con un prompt preparado[3] a un modelo de lenguaje de gran tamaño(LLM). Este modelo genera frases de búsqueda que, posteriormente, se utiliza para realizar consultas en un motor de búsqueda. A partir de los resultados obtenidos, se seleccionan los primeros listados junto con sus respectivas URLs.

3. **Generar respuesta:** Para generar una respuesta completa y clara para el cliente, es necesario incorporar tanto la descripción de la historia de usuario como los resultados obtenidos en pasos anteriores. Utilizando un modelo de lenguaje de gran tamaño(LLM), se obtendrá una respuesta concisa que se puede entregar directamente al cliente.

Además, la figura 4.2 ilustra el proceso utilizado para almacenar historias de usuario de Jira en una base de datos vectorial. Se desarrollará un script que exporte las descripciones de las historias de usuario, junto con sus identificadores, en formato JSON. Estos datos serán posteriormente transformados en vectores mediante un modelo de embedding. Finalmente, se almacenarán tanto los datos como sus vectores asociados en la base de datos vectorial.

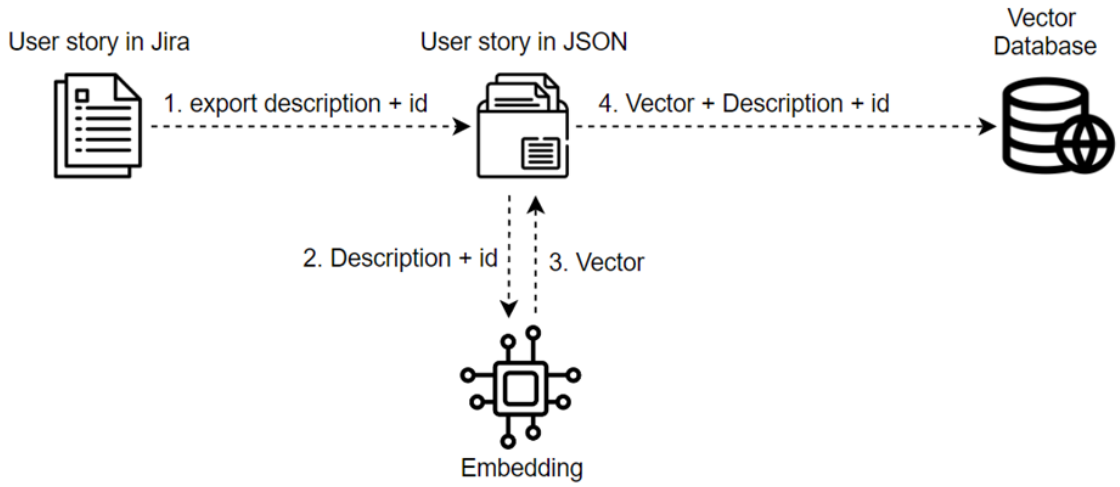


**Figura 4.2:** Estructura de procesar datos

### 4.1.1. Patrones de diseño

Dado que la herramienta tiene su frontend desarrollado en React, es relevante destacar el conjunto de patrones estructurales y creacionales que rigen el comportamiento de este framework, lo que lo convierte en una excelente opción para proyectos de gran escala:

[1]Se explicará en detalle en la sección 4.2

[2]Las bases de datos vectoriales proporcionan la capacidad de almacenar y recuperar vectores como puntos de alta dimensión. Añaden capacidades adicionales para la búsqueda eficiente y rápida de los vecinos más cercanos en el espacio N-dimensional.(https://aws.amazon.com/es/what-is/vector-databases/)

[3]Un prompt es una instrucción o texto inicial que se proporciona a una herramienta de IA generativa para guiar la generación de respuestas o resultados.

- **Patrón de Composición[2]:** React tiene una notable ventaja de permitir la creación de interfaces compuestas por múltiples componentes, que a su vez pueden estar compuestos por otros. Esta capacidad posibilita la reutilización de un mismo componente en diversas vistas, como la repetición de un componente dentro de una misma vista. Por ejemplo, en una lista de libros, donde cada resultado de la lista utiliza el mismo componente, adaptándose a la información de cada libro.

- **Patrón Estado[5]:** Los estados son fundamentales en React. Este patrón es el actor principal de su comportamiento reactivo, ya que la reactividad de React se deriva de los cambios en los estados internos de la aplicación. Esta característica posibilita que partes de la interfaz de usuario o la lógica de negocio actúen de manera condicional, basándose en el valor de un estado. Por tanto, su comportamiento puede entenderse y representarse como el de una máquina de estados.

- **Patrón Observador[3]:** Este patrón permite definir un mecanismo de suscripción para notificar a varios objetos sobre cualquier evento que le suceda al objeto que están observando. En React este es el fundamento de múltiples hooks[30], como useEffect[31].

En el backend que utiliza el framework Spring también se emplean patrones de diseño para mejorar la mantenibilidad del repositorio y facilitar ciertas implementaciones:

- **Patrón Singleton[4]:** Spring utiliza este patrón para la gestión de los beans[4] por defecto. Cada bean definido en el contexto de Spring se crea como un singleton, lo que significa que Spring crea una única instancia de cada bean por defecto.

- **Patrón de inyección de dependencias[1]:** Este es probablemente el patrón más central en Spring, utilizado para gestionar las dependencias entre los diferentes objetos. Permite la creación de aplicaciones más desacopladas y fáciles de mantener.

Además de los patrones fundamentales que gobiernan el comportamiento de React y Spring, también se ha incorporado la arquitectura hexagonal en el servidor para fortalecer la cohesión y la modularidad de nuestro sistema.

- **Arquitectura Hexagonal - patrón puertos y adaptadores[15]:** Esta arquitectura se representa comúnmente mediante un hexágono(Figura 4.3), aunque el número de lados es menos significativo que lo que simbolizan. Cada lado del hexágono representa un puerto, que sirve como interfaz para la comunicación tanto interna como externa de la aplicación. La motivación principal de este diseño es segmentar la aplicación en distintas capas, cada una con responsabilidades definidas. Esto permite un desacoplamiento efectivo de las capas, facilitando su evolución independiente. Además, al separar el sistema por responsabilidades, se mejora la reusabilidad de sus componentes.

[4]Los beans son componentes de software reutilizables que se escriben en el lenguaje de programación Java y se pueden manipular utilizando herramientas de creación pensadas para beans.

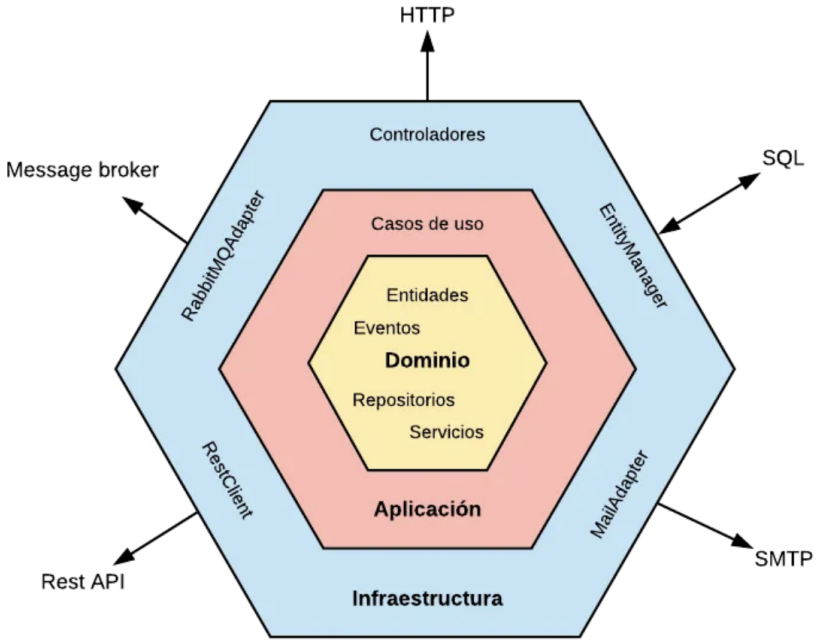


**Figura 4.3:** Ejemplo de estructura hexagonal

## 4.2 Tecnologías utilizadas

### 4.2.1. Herramientas

1. **Jira**

   Jira[7] es una herramienta de gestión de proyectos y seguimiento de problemas ampliamente reconocida y utilizada en diversos campos laborales, especialmente en el desarrollo de software. Proporciona un entorno centralizado que optimiza la planificación, el seguimiento y la colaboración en proyectos, ayudando a los equipos a mantenerse organizados y concentrados en sus tareas de manera efectiva.

   En este proyecto, se utiliza Jira en dos sitios distintos. Inicialmente, se emplea la herramienta para gestionar las tareas en el tablero Kanban, lo que mejora significativamente la organización del proyecto. Posteriormente, Jira nos permite realizar un seguimiento detallado del avance de cada tarea y de los problemas que surgen durante el desarrollo, garantizando que se atiendan los desafíos de manera oportuna y efectiva.

   Por otro lado, Inditex emplea una plataforma interna de Jira que simplifica considerablemente la creación y gestión de historias de usuario. En este proyecto, se desarrollará una extensión que se interactuará con esta plataforma, facilitando la exportación de las historias a una base de datos vectorial y permitiendo la extracción de descripciones para enriquecerlas posteriormente.

2. **Figma**

   Figma[17] es una herramienta altamente profesional de diseño y prototipado, ideal para la colaboración en proyectos de diseño. Con Figma, los usuarios pueden crear interfaces de usuario interactivas y estáticas de manera intuitiva, gracias a su am-

plia gama de herramientas para crear gráficos, que incluyen formas, iconos, imágenes y texto. Además, facilita la importación de recursos y se integra sin problemas con otras plataformas de diseño.

En cuanto a la curva de aprendizaje, Figma es adecuada tanto para principiantes como para profesionales. Su interfaz extremadamente simple y su funcionamiento intuitivo hacen que sea fácil de aprender y utilizar para cualquier nivel de habilidad en diseño. Además, ofrece la capacidad de generar código CSS, lo que agrega un valor adicional al proceso de diseño y desarrollo.

3. **Github**

   Github[18] es una plataforma de desarrollo de software basada en Git que facilita la colaboración entre desarrolladores, la gestión de versiones y la compartición eficiente de trabajo. Destaca por su enfoque comunitario, herramientas avanzadas de gestión de proyectos y robustas características de seguridad, lo que la hace indispensable para el desarrollo colaborativo de software.

   Específicamente, esta herramienta ha sido fundamental en la gestión de versiones del código, asegurando una organización clara y ofreciendo una transparencia completa en las modificaciones realizadas. Github se caracteriza por su interfaz intuitiva y fácil de usar, proporcionando herramientas accesibles para principiantes y al mismo tiempo avanzadas para desarrolladores experimentados.

4. **Visual Studio Code**

   Visual Studio Code[33], desarrollado por Microsoft, es un editor de código fuente extremadamente popular que se destaca por su flexibilidad, funcionalidad y facilidad de uso. Aunque es compatible con una amplia variedad de lenguajes de programación, está especialmente optimizado para trabajar con tecnologías web como HTML, CSS y JavaScript.

   Una característica sobresaliente de Visual Studio Code es su capacidad de extensión. Los usuarios pueden personalizar y enriquecer el editor instalando una gran variedad de extensiones disponibles, que añaden funcionalidades adicionales y soporte para diversos lenguajes y frameworks. Esta capacidad de personalización permite que cada desarrollador adapte el entorno de programación a sus necesidades específicas.

5. **IntelliJ IDEA**

   IntelliJ IDEA[23] es un entorno de desarrollo integrado(IDE) desarrollado por JetBrains, ampliamente reconocido por su robustez, funcionalidad integral y experiencia de usuario fluida. Aunque admite una multitud de lenguajes de programación, IntelliJ es especialmente famoso por su excepcional soporte para Java.

   Una de las características distintivas de IntelliJ IDEA es su asistencia inteligente para codificación, que sobrepasa la simple autocompletación al ofrecer análisis en tiempo real, detección de errores y refactorizaciones ingeniosas. Esta funcionalidad permite a los desarrolladores escribir código limpio y mantenible de manera más eficiente, reduciendo significativamente el tiempo de desarrollo y mejorando la calidad del código.

   Además, IntelliJ IDEA está diseñado para ser altamente extensible. Los usuarios pueden personalizar y ampliar el entorno instalando una amplia variedad de complementos disponibles a través del repositorio de JetBrains, lo que permite adaptar

el IDE a las necesidades específicas de cada proyecto. Esto convierte a IntelliJ IDEA en una herramienta indispensable para desarrolladores que buscan una solución efectiva y adaptable para los problemas de programación moderna.

6. **Docker**

   Docker[14] es una plataforma de software que le permite crear, probar e implementar aplicaciones rápidamente. Docker empaqueta software en unidades estandarizadas llamadas contenedores que incluyen todo lo necesario para que el software se ejecute, incluidas bibliotecas, herramientas de sistema, código y tiempo de ejecución. Con Docker, puede implementar y ajustar la escala de aplicaciones rápidamente en cualquier entorno con la certeza de saber que su código se ejecutará.

   En concreto, esta herramienta se ha utilizado para descargar contenedores de servicios de terceros, como Weaviate y LLM, y también se utilizará en la fase de despliegue, facilitando así un proceso de implementación eficiente y consistente. Docker encapsula todo el entorno de la aplicación en un contenedor autónomo, asegurando la uniformidad entre los entornos de desarrollo, prueba y producción. Esto es crucial para mantener la integridad y la fiabilidad del software a lo largo de todo el ciclo de vida del desarrollo.

7. **Kubernetes**

   Kubernetes[24] es una plataforma de código abierto diseñada para automatizar la implementación, el escalado y la gestión de aplicaciones contenerizadas. Esta herramienta facilita la gestión de aplicaciones distribuidas a gran escala, permitiendo una configuración flexible y automática según las necesidades del usuario. Kubernetes organiza los contenedores que componen una aplicación en unidades lógicas para facilitar su descubrimiento y gestión. Ofrece mecanismos robustos para el mantenimiento y la actualización sin interrupciones de las aplicaciones. Además, proporciona servicios de descubrimiento y balanceo de carga, lo que mejora la eficiencia del uso de los recursos y la disponibilidad de las aplicaciones. Su arquitectura modular y extensible permite a los desarrolladores personalizar y adaptar servicios según los requisitos específicos de su infraestructura.

8. **Overleaf**

   Overleaf[37] es una plataforma de edición en línea que permite la creación de documentos científicos y técnicos utilizando LaTeX, un sistema de composición de textos especializado en la elaboración de libros, documentos científicos y técnicos que contienen fórmulas matemáticas.

   La plataforma facilita el proceso de colaboración y compilación, ofreciendo una amplia gama de plantillas prediseñadas que cumplen con las normas de publicación de numerosas revistas científicas y académicas. Esto simplifica la preparación y el envío de trabajos para su publicación.

   Una característica destacada de Overleaf es su interfaz de usuario intuitiva, que no requiere conocimientos previos de LaTeX para comenzar. Los usuarios pueden simplemente escribir su contenido y Overleaf se encarga del formateo, compilando los documentos en PDF con calidad profesional. Esto hace que la plataforma sea accesible para aquellos que son nuevos en LaTeX, al tiempo que sigue siendo lo suficientemente poderosa para usuarios experimentados que desean aprovechar todas las capacidades avanzadas que ofrece LaTeX.

9. **PromptPerfect**

   PromptPerfect[40] es una herramienta diseñada para mejorar el rendimiento de los modelos de lenguaje. Su objetivo es ayudar a los usuarios a optimizar el texto de sus indicaciones para guiar mejor la generación de contenido por parte del modelo de lenguaje.

   Esta herramienta ofrece una variedad de funciones para satisfacer las necesidades de los usuarios. Puede ayudar a generar automáticamente varias variantes de las indicaciones para ampliar las posibilidades y proporcionar retroalimentación en tiempo real para guiar a los usuarios hacia la creación de indicaciones más efectivas.

   Además, PromptPerfect es muy fácil de aprender. Ofrece una interfaz intuitiva y funciones sencillas de usar, lo que permite a los usuarios editar, ajustar y mejorar sus indicaciones de manera fácil. Incluso aquellos sin experiencia profesional pueden dominar rápidamente las funciones básicas de PromptPerfect.

10. **Postman**

    Postman[39] es una herramienta de desarrollo de API extremadamente popular que apoya el diseño, la prueba, la depuración, el monitoreo y la publicación de APIs. Proporciona una interfaz de usuario intuitiva que facilita el envío de solicitudes de red, la recepción de respuestas, la visualización de los datos devueltos por el servidor y la realización de pruebas de rendimiento.

    Además, puede generar automáticamente documentación detallada de la API a partir de colecciones de solicitudes, la cual puede ser compartida públicamente. Postman también ofrece funciones de monitoreo programado que verifican regularmente la disponibilidad y el tiempo de respuesta de la API para ayudar a los equipos a identificar y resolver problemas de rendimiento de manera oportuna.

    Estas características hacen de Postman una herramienta indispensable que soporta varios tipos de API, como REST, SOAP y GraphQL, simplificando enormemente el proceso de prueba y desarrollo de API y mejorando la eficiencia del desarrollo.

### 4.2.2. Frontend

1. **ReactJS**

   ReactJS[16] es una de las bibliotecas más populares dentro de JavaScript para el desarrollo de aplicaciones móviles y web. Se centra en la creación de componentes reutilizables, lo que facilita el desarrollo eficiente y modular de aplicaciones web.

   El principal rasgo distintivo de React JS es su enfoque en la construcción de interfaces de usuario de manera declarativa. Utilizando React, los desarrolladores pueden crear componentes individuales que representan partes específicas de la interfaz, y luego combinarlos para formar aplicaciones completas. Estos componentes son altamente reutilizables y se actualizan automáticamente en respuesta a cambios en los datos, facilitando así la creación de aplicaciones interactivas y responsivas.

   React JS emplea una técnica conocida como “DOM virtual”(Virtual DOM) para optimizar el rendimiento. React no modifica directamente el DOM del navegador, sino que genera una representación del DOM virtual y realiza comparaciones eficientes para determinar los cambios que deben aplicarse. Esto reduce las operaciones costosas en el DOM real, mejorando así el rendimiento general de las aplicaciones.

   Grandes empresas como Facebook, Netflix y Microsoft utilizan este framework, lo que demuestra que es una excelente opción para el desarrollo de proyectos a gran

escala. Aunque la curva de aprendizaje es moderada y requiere adaptarse a un paradigma de programación declarativo, una vez dominado, el uso de React para crear componentes es mucho más eficiente que utilizando solo JavaScript.

2. **TypeScript**

   TypeScript[32] es un lenguaje de programación que mejora JavaScript[10] al agregar tipado estático y características avanzadas. Proporciona un sistema de tipos sólido que ayuda a prevenir errores y mejora la productividad en el desarrollo de software. Gracias a su capacidad de transpilarse a JavaScript estándar y su amplia comunidad de desarrolladores, TypeScript se ha convertido en una opción popular para proyectos de gran escala y para el desarrollo de aplicaciones web y móviles.

   La curva de aprendizaje de un lenguaje como TypeScript es normal. Inicialmente es baja gracias al sistema de tipado, pero se vuelve más elevada a medida que se profundiza en el conocimiento del lenguaje y se exploran sus funcionalidades avanzadas sobre JavaScript.

3. **CSS**

   CSS[9] es un lenguaje utilizado para mejorar la presentación de documentos HTML o XML. Permite a diseñadores web y desarrolladores controlar el layout, los colores, las fuentes y otros elementos visuales de una página web. CSS es uno de los pilares del diseño web y, junto con HTML y JavaScript, forma el núcleo de las tecnologías de desarrollo web.

   El objetivo principal de CSS es separar el contenido del documento de su forma de presentación. Esta separación no solo mejora la accesibilidad del contenido, sino que también simplifica el mantenimiento del documento, ya que permite controlar los estilos de múltiples páginas desde un solo lugar. Además, el uso de CSS puede reducir el tamaño y la complejidad de los archivos HTML, lo que a su vez acelera la velocidad de carga de las páginas.

4. **JavaScript**

   JavaScript es un lenguaje de programación de alto nivel e interpretado, principalmente utilizado para implementar interactividad dinámica y mejorar la experiencia del usuario en páginas web. Se usa ampliamente en el desarrollo frontend para crear páginas web y aplicaciones web interactivas. JavaScript puede integrarse directamente en HTML y se ejecuta en el navegador web. Admite paradigmas de programación orientada a objetos, funcional y basada en prototipos. JavaScript tiene un sistema de tipos dinámicos que permite a los desarrolladores modificar y manipular datos dinámicamente durante la ejecución.

   Con JavaScript, se desarrolla un script de contenido para el frontend, lo que nos permite comunicarnos con la página web desde nuestra extensión. Este script se ejecuta en el contexto de la página web y nos brinda la capacidad de interactuar dinámicamente con los elementos de la interfaz de usuario y los datos presentes en la página.

### 4.2.3. Backend

1. **Spring Boot**

   Spring Boot[45] es una extensión del marco de trabajo Spring diseñada para simplificar la creación de aplicaciones basadas en Java, ofreciendo una configuración

predeterminada que reduce significativamente el tiempo y esfuerzo necesarios para empezar a desarrollar. Spring Boot facilita la creación de aplicaciones autónomas, listas para producción, y minimiza la configuración manual al proporcionar un entorno preconfigurado con una serie de herramientas y funcionalidades útiles.

En nuestro proyecto, se ha aprovechado Spring Boot para la parte del servidor, lo que nos ha brindado ventajas significativas. Esto incluye la simplificación del desarrollo gracias a su amplia gama de funcionalidades predefinidas, así como su modularidad y escalabilidad. La inyección de dependencias simplifica la gestión de las relaciones entre componentes, mientras que el soporte para múltiples capas arquitectónicas nos ha permitido estructurar nuestra aplicación de manera eficiente.

Una de las mayores ventajas de Spring Boot es su capacidad para iniciar rápidamente un proyecto con una configuración mínima gracias a su filosofía de çonvención sobre configuración". Esto, combinado con sus "starters", nos permite añadir dependencias y configuraciones específicas de manera sencilla y eficiente. Además, la integración con Spring Initializr proporciona un punto de partida rápido y personalizado para nuevas aplicaciones.

2. **Spring AI**

   Spring AI[44], una iniciativa relativamente nueva dentro del ecosistema de Spring, está diseñada específicamente para simplificar la integración de capacidades de inteligencia artificial y aprendizaje automático en aplicaciones Spring. Esta extensión busca aprovechar el poder de las tecnologías de IA, haciéndolas más accesibles para los desarrolladores de Java que ya están familiarizados con la arquitectura y funcionalidades de Spring.

   El objetivo principal de Spring AI es proporcionar una manera eficiente y sin interrupciones para que los desarrolladores incorporen modelos de IA y algoritmos de aprendizaje automático en sus aplicaciones basadas en Spring. Al hacerlo, responde a la creciente demanda de funcionalidades impulsadas por IA en aplicaciones empresariales, tales como análisis predictivo, procesamiento de lenguaje natural y tareas de reconocimiento de imágenes. Spring AI ofrece un conjunto de herramientas y bibliotecas que facilitan la conexión con servicios de IA y la implementación de modelos de aprendizaje automático sin requerir un profundo conocimiento en tecnologías de IA.

   En este proyecto, se va a utilizar Spring AI para integrar un LLM y una base de datos vectorial. Esto facilitará la realización de búsquedas semánticas avanzadas y potenciará la habilidad de nuestra aplicación para interpretar y manejar el lenguaje natural con mayor eficacia. La integración de estas tecnologías se llevará a cabo mediante la importación de paquetes específicos proporcionados por Spring AI, lo que simplificará significativamente la gestión de operaciones de inteligencia artificial complejas.

3. **Weaviate**

   Weaviate[5] es una base de datos de vectores administrada de código abierto y un motor gráfico de conocimiento. Weaviate almacena y busca datos vectoriales, descubriendo conexiones semánticas entre ellos. Además, ofrece búsqueda avanzada basada en contexto y similitud de vectores[38].

[5]https://github.com/weaviate/weaviate

Para este TFG, la elección de Weaviate puede ser estratégica debido a varias razones. Primero, al ser de código abierto, reduce los costos de licencia y permite una personalización extensiva según las necesidades específicas del proyecto. Además, la capacidad de Weaviate para manejar datos vectoriales es crucial. Weaviate utiliza técnicas de indexación basadas en grafos, lo que facilita búsquedas rápidas y eficientes, incluso en bases de datos de gran tamaño.

Otro aspecto destacable es la comunidad activa y en crecimiento que rodea a Weaviate, la cual ofrece un valioso soporte a través de documentación detallada, tutoriales y foros de discusión. Esto puede ahorrar tiempo y esfuerzo en el proceso de aprendizaje y desarrollo del proyecto.

4. **Maven**

Maven[6] es una herramienta de automatización y gestión de proyectos desarrollada por Apache, ampliamente utilizada en el desarrollo de software, especialmente para proyectos Java. Su enfoque se centra en el uso del modelo de objeto de proyecto(POM), que emplea un archivo XML para definir el proyecto y gestionar sus dependencias con otros módulos y componentes. Maven simplifica la configuración y la coherencia del proyecto al manejar automáticamente las dependencias a través de su repositorio central.

Además de gestionar dependencias, Maven automatiza varias tareas de construcción de proyectos, como la compilación de código, el empaquetado en formatos como JAR y WAR, y la ejecución de pruebas. Estas tareas se configuran a través del POM, lo que permite un control detallado sobre cada fase de la construcción y apoya las prácticas de integración continua. Maven también facilita la creación de proyectos basados en arquetipos, plantillas predefinidas que estandarizan la estructura y configuración de los proyectos en una organización.

La integración de Maven con una variedad de entornos de desarrollo integrado(IDE) y sistemas de control de versiones mejora su flexibilidad y compatibilidad, convirtiéndolo en una herramienta valiosa para desarrolladores que buscan eficiencia y estandarización en la construcción y gestión de proyectos de software. Esta capacidad de integración facilita un desarrollo más fluido y una gestión de proyectos más efectiva, lo que convierte a Maven en la opción preferida para muchos en la industria.

5. **Modelo de embedding - text2vec-transformers**

En este proyecto, se emplea un modelo local de embeddings llamado text2vec-transformers, desarrollado como un modelo interno de Weaviate. Este modelo se ha desplegado en Docker para optimizar la conversión de historias de usuarios en vectores matemáticos. Los embeddings son una técnica avanzada de procesamiento de lenguaje natural que transforma el lenguaje humano en vectores matemáticos(ver Figura 4.4). Esta representación vectorial captura el significado subyacente de las palabras, lo que facilita un procesamiento del lenguaje por parte de las computadoras mucho más eficiente [20]. Gracias a estos vectores, es posible identificar frases con semánticas similares, mejorando así la precisión en la búsqueda y análisis de texto.

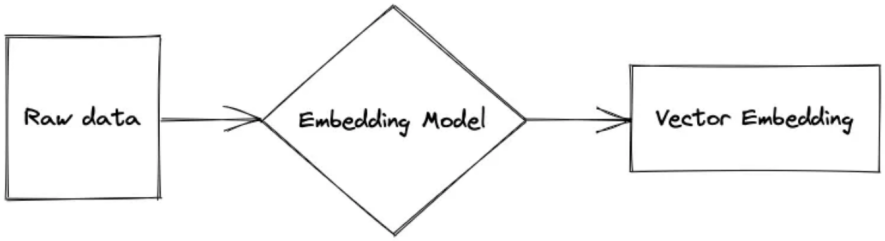


**Figura 4.4:** Proceso de Embeddings

6. **Ollama**

   Ollama[35] es una herramienta que nos permite ejecutar modelos de lenguaje en nuestros ordenadores de manera sencilla. Su principal propósito es facilitar el acceso a modelos de gran tamaño sin necesidad de utilizar la nube. Con Ollama, se puede descargar y ejecutar varios modelos, incluyendo LLaMA-2, Uncensored LLaMA, CodeLLaMA, Falcon y Mistral, entre otros. Esta herramienta es especialmente útil para personas no técnicas que desean explorar y experimentar con modelos de lenguaje sin complicaciones.

   Además, su facilidad de uso y la amplia variedad de modelos compatibles hacen de Ollama una opción atractiva para aquellos que desean experimentar con generación de texto y lenguaje natural. Si bien aún tiene algunas limitaciones, Ollama promete ser una herramienta muy útil para desarrolladores, investigadores y entusiastas del procesamiento del lenguaje natural.

7. **Restful API**

   Para facilitar una comunicación eficiente y confiable entre el cliente y el servidor, se ha seleccionado utilizar una API RESTful. Este enfoque arquitectónico, conocido como Representational State Transfer, establece un conjunto de reglas y restricciones para crear servicios web. Estas APIs son esenciales para permitir la interacción fluida entre sistemas en la web, facilitando así la comunicación entre componentes de software en Internet. Aunque REST es independiente de cualquier protocolo subyacente, comúnmente se implementa sobre HTTP.

   La idea fundamental de REST es considerar todos los componentes del sistema como recursos accesibles mediante una interfaz uniforme y predecible. Cada recurso en una API RESTful se identifica mediante una URI única. Las operaciones disponibles para manipular estos recursos se representan comúnmente mediante métodos HTTP como GET, POST, PUT y DELETE.

   Además, la arquitectura RESTful se basa en un diseño sin estado. Esto implica que cada solicitud de un cliente al servidor debe contener toda la información necesaria para entender y completar la solicitud, sin que el servidor retenga datos de estado de la sesión del cliente entre peticiones. Esta característica es particularmente beneficiosa en entornos de Internet donde se busca escalabilidad y rendimiento, ya que libera al servidor de la carga de mantener información de estado del usuario.

8. **Java**

   Java es un lenguaje de programación orientado a objetos que se caracteriza por su portabilidad entre diferentes plataformas y sistemas operativos. Es ampliamente

utilizado para desarrollar aplicaciones empresariales, móviles y web debido a su robustez, seguridad y escalabilidad.

El backend se desarrolla en Java con el framework Spring Boot y Spring AI. Esta combinación nos permite desarrollar soluciones complejas y eficientes que pueden adaptarse fácilmente a las cambiantes demandas.

# CAPÍTULO 5
# Desarrollo de la solución propuesta

En este capítulo se detalla la implementación del proyecto, poniendo especial énfasis en las secciones del código más relevantes y complejas. Se explorarán diversas áreas del backend y frontend, explicando en profundidad la arquitectura y los distintos componentes que la conforman.

## 5.1 Backend

### 5.1.1. Arquitectura de Backend

En el presente proyecto, se ha seleccionado el framework Spring Boot para desarrollar el backend. Además, se ha aprovechado Maven para simplificar la gestión de proyectos y dependencias. También se ha utilizado GitHub para la gestión del versiones del código.

Dado que Spring ha desarrollado una nueva framework Spring AI para gestionar casos de uso de IA, se optó por utilizarlo para interactuar con una base de datos vectorial y un LLM. Spring AI ofrece varias opciones para estas dos partes. Finalmente, se ha elegido utilizar Weaviate como nuestra base de datos vectorial y Ollama como el la herramienta para ejecutar modelos de lenguaje.

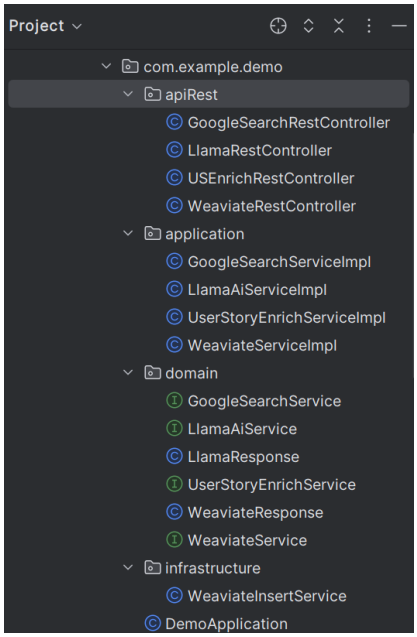


**Figura 5.1:** Estructura hexagonal en el proyecto

Además, se ha implementado una arquitectura hexagonal para segmentar los diferentes servicios en múltiples capas independientes(Figura 5.1), facilitando su evolución autónoma. Esto también aumenta la reusabilidad de cada componente. Por ejemplo, el servicio de Ollama puede utilizarse tanto para generar consultas de búsqueda basadas en la historia del usuario en GoogleSearchRestController como para crear respuestas integrando información relevante en LlamaRestController.

### 5.1.2. Gestión de la base de datos

Para desplegar un servidor de Weaviate y un modelo de embedding localmente, se utiliza Docker para instalar el contenedor de Weaviate y el modelo de embedding text2vec-transformers.

Luego, se crea un bean (Figura 5.2) para conectar con ambos.

```
no usages  new *
@Autowired
public WeaviateServiceImpl(VectorStore vectorStore, WeaviateInsertService weaviateInsertService) {
    this.vectorStore = vectorStore;
    this.weaviateInsertService = weaviateInsertService;

}
no usages  new *
@Bean
public VectorStore vectorStore(EmbeddingClient embeddingClient) {
    WeaviateVectorStore.WeaviateVectorStoreConfig config = WeaviateVectorStore.WeaviateVectorStoreConfig.builder()
            .withScheme("http")
            .withHost("localhost:8080")
            .withConsistencyLevel(WeaviateVectorStore.WeaviateVectorStoreConfig.ConsistentLevel.ONE)
            .build();
    return new WeaviateVectorStore(config, embeddingClient);
}
```

**Figura 5.2:** Conexión con weaviate

También se puede observar que se utiliza la anotación @Autowired para inyectar automáticamente el servicio de inserción de historias de usuario cada vez que se reinicia el backend. Esto asegura que los datos guardados se actualicen automáticamente.

Además, para obtener datos, se ha desarrollado un script que extrae información de las páginas web de historias de usuario en Jira. Se personalizan estos datos al formato de Document[1] proporcionado por Spring AI, lo que facilita el proceso de realizar embeddings y almacenarlos en Weaviate.

Al final, se ha implementado un servicio de Weaviate que ofrece cuatro funciones esenciales para realizar operaciones CRUD: detección de duplicados, inserción, eliminación y actualización. En lugar de acceder directamente a los datos, se utiliza el método similaritySearch(Figura 5.3) de Spring AI para buscar historias de usuario similares.

```
@Override
public List<Document> detectDuplicate(String userStory){
    List<Document> results = vectorStore.similaritySearch(
            SearchRequest
                    .query(userStory)
                    .withTopK(3));
    return results;
}
```

**Figura 5.3:** Implementación del método detectDuplicate

[1]https://docs.spring.io/spring-ai/docs/current/api/org/springframework/ai/document/Document.html

### 5.1.3. Lógica de negocio

Como se menciona en la sección 4.1, planeamos implementar tres partes principales: búsqueda de historias de usuario, búsqueda en internet y generación de respuestas.

Para la primera parte, se ha implementado en el servicio de Weaviate el método "detectDuplicate", el cual permite recuperar las tres historias de usuario más similares mediante una búsqueda por vectores. En este contexto, Weaviate cuenta con un modelo "t2v-transformers-models[2]"para hacer embedding, convirtiendo así la historia de entrada en vectores que facilitan esta búsqueda.

Para las otras dos partes, se ha desarrollado un servicio utilizando Ollama. Se despliega un contenedor de Ollama en Docker, lo que simplifica el uso del modelo de lenguaje. Este servicio permite generar frases de búsqueda basadas en el resumen de la historia y el prompt proporcionados. Luego, se realizan búsquedas en Google con estas frases utilizando Jsoup para obtener información adicional.

Finalmente, se utilizará el servicio de Ollama para generar una respuesta que enriquezca la historia del usuario, basada en el prompt, el resumen proporcionado, historias similares y los resultados de búsqueda en Google. El resultado no solo devolverá una historia de usuario completa, sino también información referenciada con sus títulos y enlaces.

Durante el proceso de desarrollo, se encontraron con dos problemas al utilizar el servicio de Ollama. En el siguiente texto se presentan los problemas identificados y cómo se han solucionado:

1. **El rendimiento de ejecución es lento:** Al conectarnos con la API para enriquecer la historia de usuario, observamos que se requiere un tiempo considerable para obtener la respuesta. Una de las razones es que el proceso de buscar historias similares y el proceso de hacer búsquedas en Google se ejecutan en secuencia. Para solucionar esto, decidimos implementar programación paralela para estas dos funciones. Como se muestra en la figura 5.4, se ha utilizado la característica Future de Java para llevarlo a cabo. Una vez obtenidos los resultados de ambas partes, se pasará el resumen de la historia de usuario introducida, la respuesta de Weaviate sobre las historias similares y los resultados de información relevante a Ollama, para que se genere una respuesta completa de la historia de usuario.

```java
@Override
public String combineResults(String resume) throws InterruptedException, ExecutionException {
    //similarity search
    Future<List<Document>> weaviateResponse = this.executorService.submit(() -> weaviateService.detectDuplicate(resume));
    //search in google
    Future<String> searchResultsFuture = this.executorService.submit(() -> {
        log.info("Searching in Google for resume: {}", resume);
        GPTResponse searchPhraseResponse = aiService.generateSearchPhrase(resume);
        List<String> searchResultsResume = googleSearchService.search(searchPhraseResponse.getResponse());
        log.info("Final Search results: {}", searchResultsResume);
        return String.join( delimiter: ", ", searchResultsResume);
    });
    //generate user story
    GPTResponse aiResponse = aiService.generateUSWithSimilarAndGoogleSearch(resume, weaviateResponse.get(), searchResultsFuture.get());
    String userStoryGenerated = aiResponse.getResponse();
    log.info("Final User story generated: {}", aiResponse.getResponse());
    return userStoryGenerated;
}
```

**Figura 5.4:** Ejecución en paralelo realizada con Java

2. **El resultado es insatisfactorio:** Cuando utilizamos el modelo de lenguaje de gran tamaño(LLM), encontramos que el resultado obtenido es a menudo impredecible

[2]https://github.com/weaviate/t2v-transformers-models

tanto en el contenido generado como en el formato. Esto se debe a que diferentes prompts pueden producir resultados diferentes. Dado que el prompt desempeña un papel crucial en mejorar el rendimiento de los LLM pre-entrenados, decidimos utilizar la herramienta PromptPerfect para optimizar el prompt introducido. La Figura 5.5 ilustra un ejemplo de cómo funciona esta herramienta para mejorar el prompt, mostrando comparaciones entre los prompt y los resultados obtenidos.

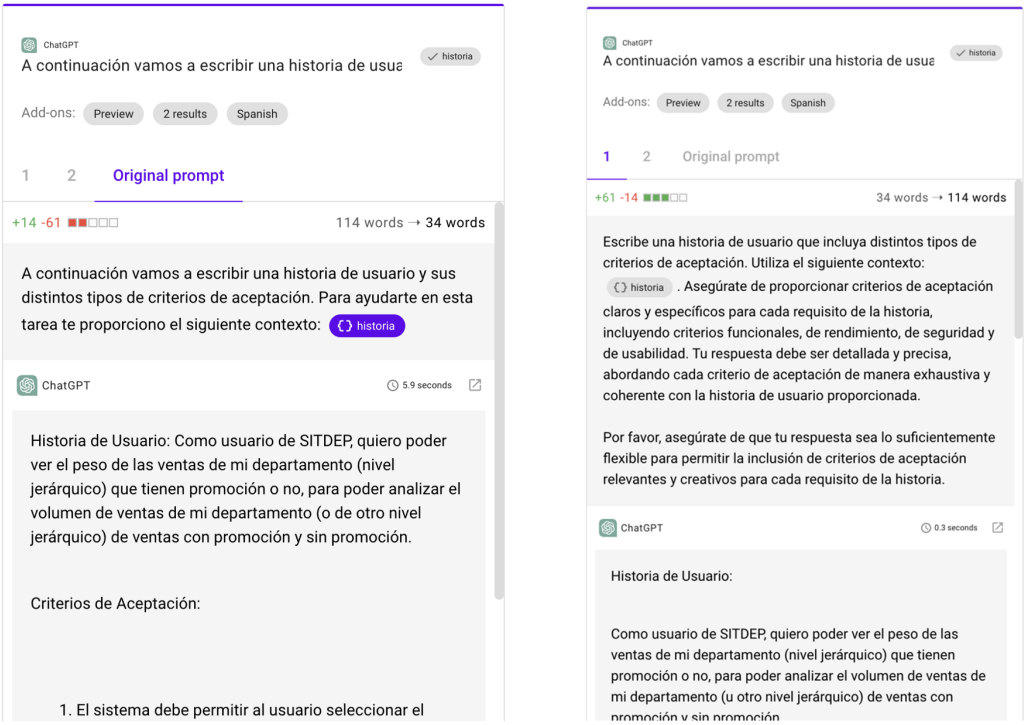


**Figura 5.5:** prompt generado con promptPerfect

Al final, definimos el formato deseado en el prompt y, con la ayuda de esta herramienta, podemos acercarnos más a obtener una respuesta deseada. En la Figura 5.6 se muestra un ejemplo del prompt utilizado para generar frases de búsqueda según un resumen de historia de usuario proporcionado. En este prompt, se ha definido el número de frases, el contenido y objetivo de estas frases, así como la estructura para facilitar la extracción posteriormente.

```
lic LlamaResponse generateSearchPhrase(String userStory) {
var prompt = """

        Crea tres frases de búsqueda en español basadas en esta historia de usuario %s
        para utilizarlas en Google en busca de información relevante y datos relacionales
        Las respuestas deben entregarse en formato JSON siguiendo esta estructura:
        {"Frase 1":.., "Frase 2:"..., "Frase 3:"...}. Asegúrate de que las frases de
        búsqueda sean específicas y relevantes para obtener información que pueda mejorar
        la historia de usuario.

      """;
final String llamaMessage = chatClient.call(String.format(prompt, userStory));
return new LlamaResponse().setMessage(llamaMessage);
```

**Figura 5.6:** prompt para generar frases de búsqueda

### 5.1.4. APIs y servicios externos

1. **API del backend**

   En el backend actual, se disponen de dos servicios de API como se muestra en la figura 5.7. El primero se utiliza para la detección de duplicaciones, donde solo se emplea la funcionalidad de buscar historias de usuario similares. En este caso, en lugar de pasar los resultados a Ollama para ajustar el formato de la respuesta, se devuelve directamente una lista de objetos. Esta mejora la velocidad de respuesta, ya que esperar la respuesta del servicio de un modelo de lenguaje tomaría mucho tiempo.

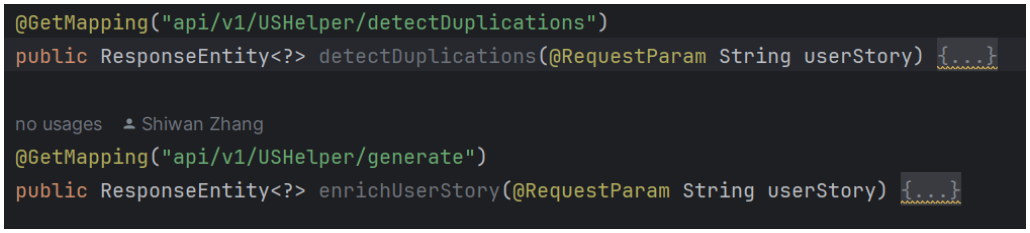


**Figura 5.7:** API del backend

   Además, el resultado de llamar a la API para enriquecer la historia de usuario consta de tres partes. La primera parte consiste en buscar historias de usuario similares a la proporcionada. La segunda parte implica generar una nueva historia de usuario basada en las historias similares encontradas. La tercera parte consiste en realizar búsquedas en Google. Finalmente, se pasa la historia generada y los resultados de Google a Ollama para refinar la historia de usuario con la ayuda de esta información relevante.

2. **Servicio de Google Search**

   Para realizar la búsqueda de información relacionada con la historia de usuario proporcionada, se ha utilizado JSoup para extraer información desde la página web de Google.

   JSoup es una biblioteca de Java utilizada para analizar, manipular y limpiar HTML. Proporciona una API conveniente que permite extraer y manipular datos de HTML desde URLs, archivos o cadenas de texto. Con JSoup, se puede extraer datos de páginas web fácilmente, enviar formularios, analizar y procesar respuestas HTML, entre otras operaciones.

   En este proyecto, utilizamos JSoup para enviar solicitudes a Google con términos de búsqueda generados por un modelo de lenguaje. Establecemos un límite en el número de resultados y recibimos la página de resultados para analizar el HTML y extraer enlaces y títulos. Esto proporciona al modelo de lenguaje acceso a más recursos y ofrece referencias detalladas a los Product Owners, enriqueciendo su capacidad para tomar decisiones informadas y agregar valor a sus proyectos.

   Dentro de la estructura hexagonal que hemos implementado, disponemos de un API para este servicio en el backend. Al llamar a este API y pasarle un breve resumen de una historia de usuario, podemos obtener las frases de búsqueda generadas junto con los resultados, incluyendo sus títulos y URLs. La figura 5.8 muestra un ejemplo de la llamada a este API, que devuelve las frases de búsqueda generadas por Ollama y los resultados correspondientes.

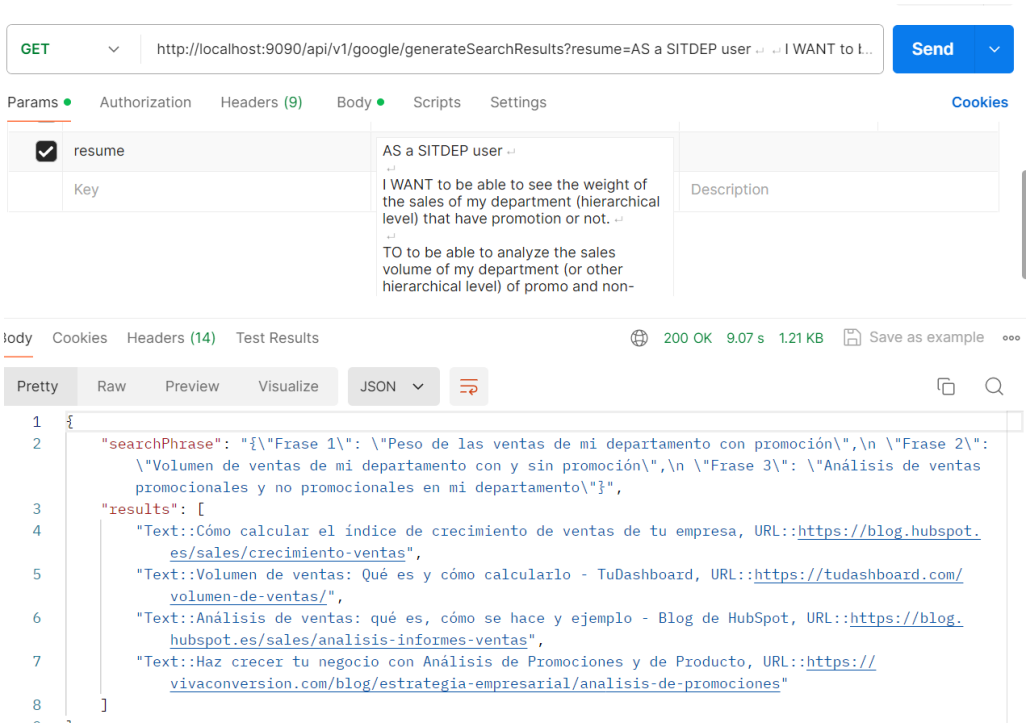


**Figura 5.8:** resultados de la API de búsqueda

## 5.2 Frontend

### 5.2.1. Arquitectura del Frontend

Para desarrollar el frontend, hemos organizado nuestro proyecto utilizando ReactJS y TypeScript. Esta combinación nos permite desarrollar componentes reutilizables y mantener un código más limpio y fácil de mantener. TypeScript agrega tipos estáticos a JavaScript, lo que nos ayuda a detectar errores durante el desarrollo y mejorar la robustez de nuestro código. En cuanto a la gestión del proyecto, se utiliza GitHub como plataforma principal.

La parte más interesante de nuestro desarrollo es la creación de una extensión para Google. Se ha definido un archivo manifest.json que contiene información sobre la extensión, como su nombre, descripción, versión y permisos necesarios. Este archivo también especifica cómo se comporta la extensión en diferentes situaciones y qué recursos utiliza. Una vez que este archivo está completo, se despliega en la página de la extensión de Google, donde los usuarios pueden instalarla y comenzar a utilizarla. La Figura 5.9 presenta el fichero manifest.json de nuestra extensión.

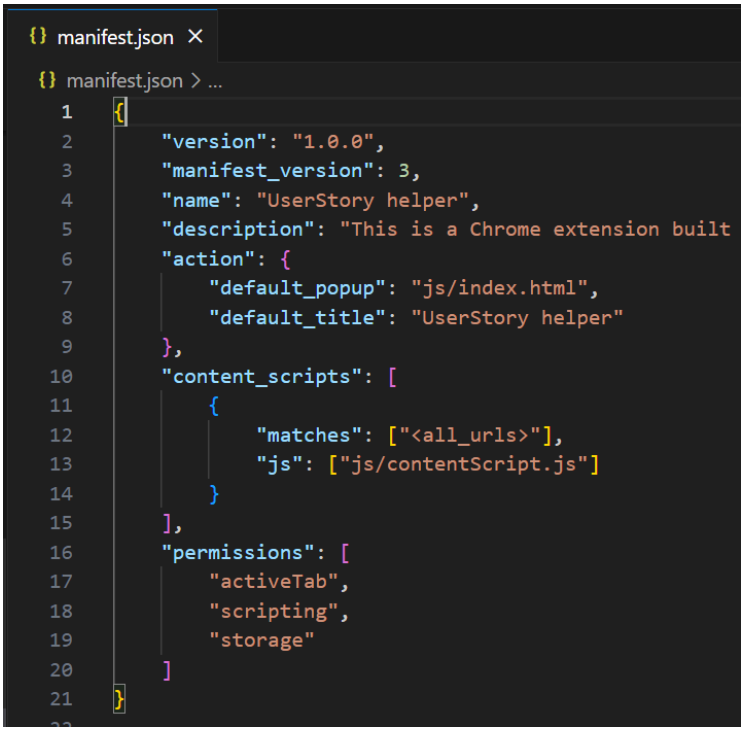


**Figura 5.9:** Implementación de manifest.json

### 5.2.2. Diseño de interfaz de usuario

Se ha utilizado Figma para diseñar la interfaz de esta extensión, la cual consta de dos botones y un área de texto(Figura 5.10). Uno de los botones se utiliza para realizar la funcionalidad de detectar las historias duplicadas, mientras que el otro se utiliza para enriquecer la historia de usuario. En este último caso, no solo se mejorará la historia de usuario, sino que también se agregarán escenarios, criterios de aceptación e información relevante.

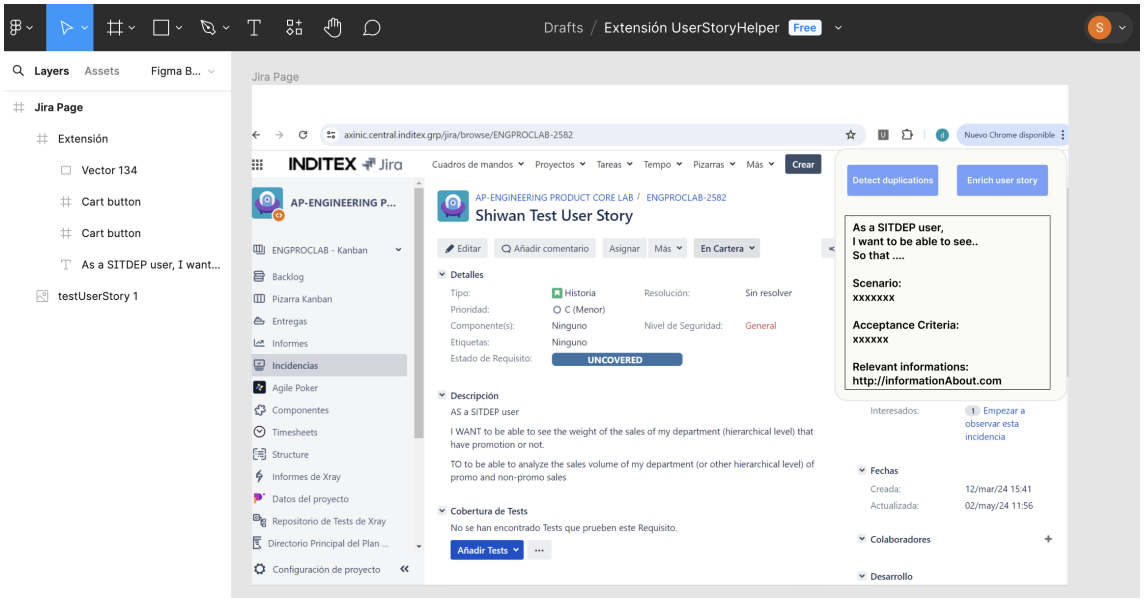


**Figura 5.10:** Interfaz de la Extensión con Figma

### 5.2.3. Interactividad y funcionalidades

En esta sección se describe cómo se gestionan los eventos de clic de cada botón.

Independientemente de cuál botón se haya clicado, primero se ejecutará un script para obtener datos de la pestaña actual y luego se extraerá la historia de usuario editada desde la caja de texto de descripción. Posteriormente, en el script, la página de Jira enviará un mensaje sobre el contenido extraído al listener de nuestra extensión.

1. **Detectar duplicaciones:** Después de recibir la historia de usuario proporcionada, se llamará a la API del servicio de detección de duplicaciones en el backend. Luego, se obtendrán los metadatos y sus enlaces correspondientes. Después de ajustar el formato de estos datos, se expondrán en el área de texto como en la figura 5.11.

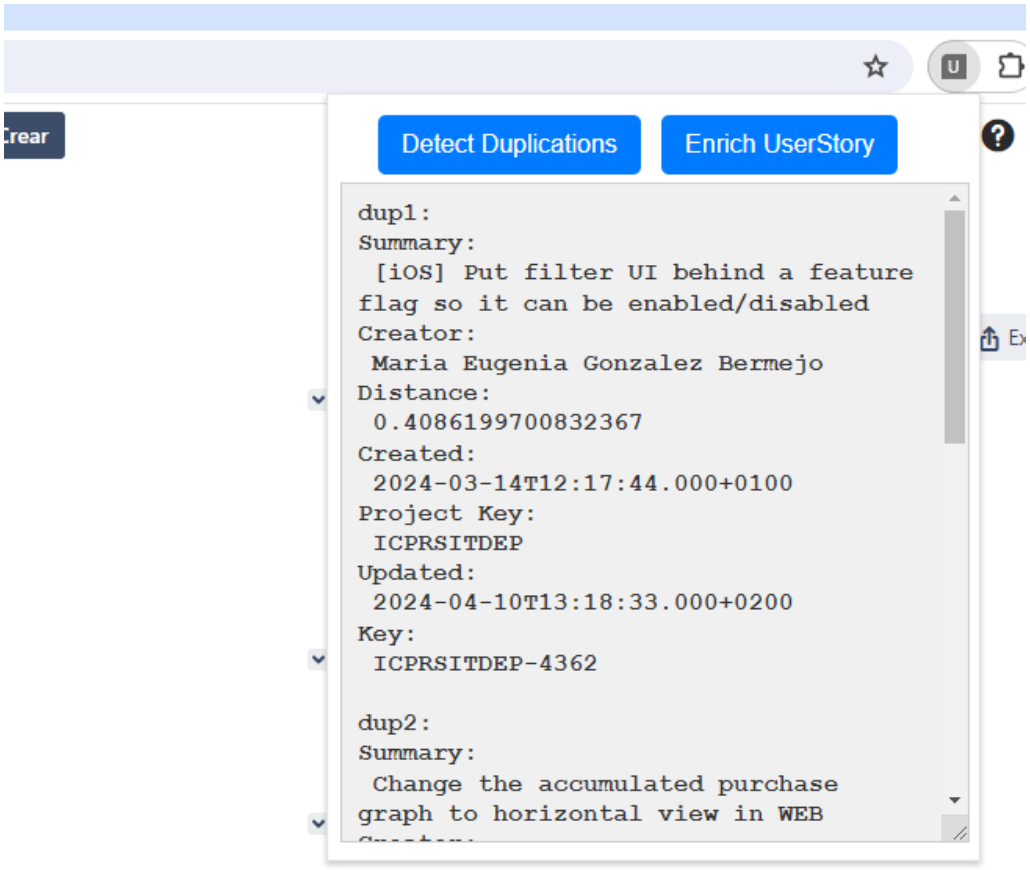


**Figura 5.11:** Resultado de detectar duplicaciones

2. **Enriquecer historia de usuario:** En este caso, se llamará a la API del servicio de enriquecimiento de historias de usuario en el backend, pasando la historia proporcionada. Luego, se generará una historia de usuario completa, basada en historias similares e información relevante obtenida de Google como en la figura 5.12. Esta incluirá escenarios y criterios de aceptación, junto con las referencias de los resultados de búsqueda de Google y sus URLs correspondientes. Posteriormente, los resultados no solo se mostrarán en el área de texto de la extensión, sino que también se integrarán en la página de Jira. Esto permite que los usuarios accedan directamente a los resultados desde Jira, lo que ahorra tiempo y mejora la eficiencia.

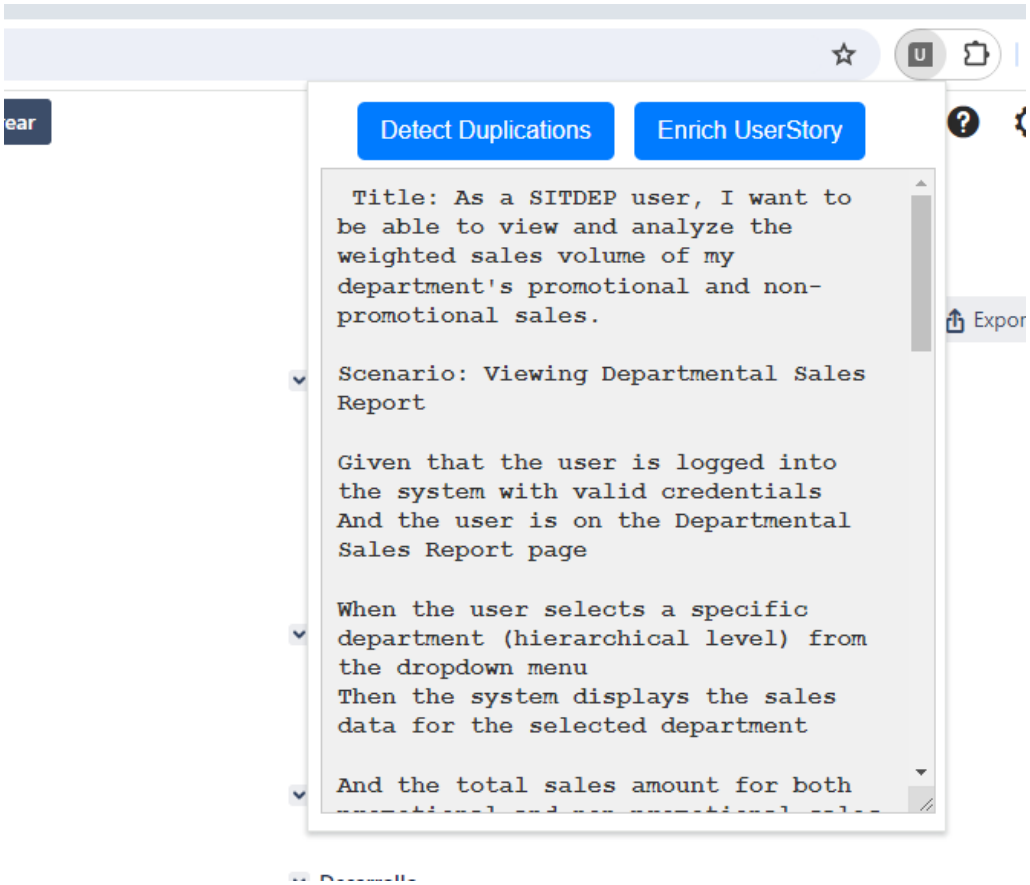


**Figura 5.12:** Resultado de enriquecer historia de usuario

### 5.2.4. Optimización

Para mejorar la interacción con el usuario y proporcionar retroalimentación visual sobre el estado de carga, se ha añadido un indicador de carga ("loading spinner") a los botones. Esto permite al usuario ver cuando se están realizando tareas en segundo plano. La figura 5.13 ilustra cómo se ve este indicador.


**Figura 5.13:** Estado de cargar

Además, hemos implementado la lógica para desactivar ambos botones mientras uno de ellos está en ejecución. Esto asegura que no se puedan realizar acciones conflictivas simultáneamente, lo que proporciona una experiencia de usuario más consistente y previene posibles errores.

En cuanto a la implementación de esta funcionalidad, representada en la figura 5.14, hemos definido dos estados: "loadingDuplicationsz "loadingUserStory", que son valores booleanos. Al hacer clic en un botón, el estado correspondiente cambia a "loading", lo que activa el indicador de carga y deshabilita ambos botones simultáneamente. Esto garantiza una navegación fluida y coherente para el usuario.

```
        const [loadingDuplications, setLoadingDuplications] = useState(false);
        const [loadingUserStories, setLoadingUserStories] = useState(false);
return (
    <div className="center">
        <button className="button" onClick={handleDetectDuplications} disabled={loadingDuplications || loadingUserStories}>
            { loadingDuplications && <i className="fa fa-refresh fa-spin"></i>}
            { loadingDuplications && <span>Detecting Duplications</span>}
            { !loadingDuplications && <span>Detect Duplications</span>}
        </button>
```

**Figura 5.14:** Código del botón de detectar duplicaciones

# CAPÍTULO 6
# Implantación

Este capítulo detalla el proceso de implementación del proyecto desarrollado en este TFG, con el objetivo de hacerlo accesible públicamente. Para ello, se ha elegido utilizar Kubernetes debido a su robustez, escalabilidad y facilidad de gestión en entornos de producción.

## 6.1 Despliege en Kubernetes

Para llevar a cabo el despliegue del backend y del frontend en Kubernetes, se han seguido varios pasos esenciales:

1. **Preparación del entorno**

   El despliegue comenzó con una preparación meticulosa del entorno, que incluyó la instalación y configuración de herramientas críticas como Docker, Kubernetes y kubectl. Esta etapa fue crucial para asegurar que todas las dependencias y herramientas estuvieran correctamente configuradas, sentando las bases para un proceso de despliegue eficiente y libre de errores.

2. **Creación y gestión de imágenes de Docker**

   Las aplicaciones del backend y del frontend fueron contenerizadas utilizando Docker, lo que las hace más flexibles y escalables. Se desarrollaron Dockerfiles específicos para cada parte de la aplicación, asegurando la inclusión de todas las dependencias necesarias. Las imágenes generadas se almacenan en un registro de contenedores, facilitando su recuperación y despliegue en un clúster de Kubernetes.

3. **Configuración y despliegue en el clúster**

   El núcleo del proceso de despliegue implica manejar los archivos de configuración YAML de Kubernetes, que detallan el comportamiento esperado de los contenedores en el clúster. Se definen recursos como deployments y services, que son cruciales para el funcionamiento y la accesibilidad de la aplicación. Mediante el uso de comandos kubectl, estas configuraciones se aplican, estableciendo el entorno operativo necesario.

4. **Pruebas y optimización**

   Una vez en el clúster, se realizaron pruebas exhaustivas para asegurar la funcionalidad y el rendimiento óptimo de las aplicaciones. Se implementaron soluciones de monitorización para evaluar el comportamiento de las aplicaciones en tiempo real y ajustar los recursos de manera proactiva. Esto incluyó la escalación de servicios

para manejar variaciones en la carga de usuarios y garantizar una alta disponibilidad.

5. **Puesta en producción**

   Finalmente, la aplicación se pone en producción y se hace accesible públicamente a través de una URL específica proporcionada por Kubernetes. Este paso marca la conclusión del proceso de despliegue y demuestra la eficacia de Kubernetes en la gestión de aplicaciones modernas en entornos en la nube.

# CAPÍTULO 7
# Pruebas

Este capítulo expone detalladamente las pruebas implementadas para garantizar la calidad, estabilidad y usabilidad del producto final.

Específicamente, se han llevado a cabo pruebas de integración y de usabilidad para abarcar tanto el backend como el frontend. En las pruebas de integración, se utilizan Postman para evaluar la funcionalidad de las APIs. Mientras tanto, en las pruebas de usabilidad, se consideran diversos objetivos de usabilidad para identificar posibles defectos en la experiencia real de uso de nuestra herramienta.

## 7.1 Pruebas de integración

Las pruebas de integración son esenciales para verificar que los diversos componentes del sistema funcionen correctamente en conjunto. En este proyecto, hemos llevado a cabo pruebas de API utilizando Postman para asegurar la eficiencia y la correcta comunicación entre las interfaces.

La prueba de API, también conocida como prueba de interfaz de programación de aplicaciones, es una técnica que se utiliza para verificar la funcionalidad, la confiabilidad y el rendimiento de una API. Implica enviar solicitudes a la API y evaluar las respuestas recibidas para garantizar que cumplan con los requisitos especificados.

Dentro del contexto de las pruebas de integración, la prueba de API se enfoca en comprobar la integración entre los diferentes componentes de software a través de sus interfaces de programación. Esto incluye enviar solicitudes a las API expuestas por estos componentes y verificar que las respuestas sean las esperadas, además de evaluar cómo se comportan cuando se comunican entre sí.

El objetivo principal de las pruebas de integración es identificar y resolver cualquier problema de interoperabilidad entre los componentes del sistema antes de su implementación final. Esto nos permite garantizar que el producto final sea coherente, estable y cumpla con los requisitos del usuario.

### 7.1.1. Diseño de casos de prueba

Para garantizar una cobertura exhaustiva, hemos diseñado casos de prueba específicos para cada una de las APIs utilizadas en el proyecto como se muestra en la tabla 7.1. Estos casos de prueba se centran en validar la funcionalidad y la integridad de las interfaces, así como en asegurar que los datos se transmitan de manera adecuada entre los diferentes componentes del sistema.

**Tabla 7.1:** Casos de prueba para la API

| Caso de Prueba | Propósito | Acción | Validación |
|---|---|---|---|
| Texto común | Verificar si el API acepta correctamente una cadena de texto común y devuelve el resultado esperado. | Ingresar una cadena de texto común y enviarla al API. | Confirmar que el API responde correctamente y devuelve el resultado esperado para la cadena de texto ingresada. |
| Texto largo | Verificar si el API puede manejar y devolver el resultado esperado para una cadena de texto larga. | Ingresar una cadena de texto larga y enviarla al API. | Confirmar que el API procesa correctamente la cadena de texto larga y devuelve el resultado esperado sin errores de procesamiento. |
| Texto con caracteres especiales | Verificar si el API puede manejar correctamente caracteres especiales, como símbolos, espacios y otros caracteres especiales, y devuelve el resultado esperado. | Ingresar una cadena de texto que contenga símbolos, espacios y otros caracteres especiales y enviarla al API. | Confirmar que el API procesa correctamente los caracteres especiales y devuelve el resultado esperado sin alteraciones en el texto o errores de procesamiento. |
| Texto vacío | Verificar si el API maneja adecuadamente las entradas de texto vacías y proporciona un mensaje de error adecuado. | Ingresar una cadena de texto vacía y enviarla al API. | Confirmar que el API responde con un mensaje de error adecuado para la entrada de texto vacía. |

### 7.1.2. Resultados de prueba de API

Se ha utilizado Postman para realizar todos los casos de prueba en ambas APIs del backend, con las acciones correspondientes, y se ha obtenido el siguiente resultado.

En los primeros tres casos de prueba, ambas APIs funcionan correctamente, como se muestra en la Figura 7.1. Sin embargo, en el último caso de prueba, al introducir una cadena de texto vacía, se observan los siguientes comportamientos:

- La API de detección de duplicaciones devuelve historias similares en lugar de un mensaje de error adecuado.

- La API de enriquecimiento de historia de usuario devuelve un mensaje de error, pero tarda un tiempo considerable en hacerlo.


**Figura 7.1:** Resultado correcto de prueba con Postman

Para mejorar el comportamiento de ambas APIs, se han realizado modificaciones en el código. Ahora, ambas APIs devuelven un mensaje de error inmediatamente cuando la historia de usuario es una cadena de texto vacía. Esta mejora se ilustra en la Figura 7.2.

```
@GetMapping("api/v1/USHelper/generate")
public ResponseEntity<?> enrichUserStory(@RequestParam String userStory) {
    try {
        final String aiResponse = userStoryEnrichService.combineResults(userStory);
        if(userStory == null || userStory.trim().isEmpty()) {
            return ResponseEntity.badRequest().body("User story is empty.");
        }
        return ResponseEntity.ok(aiResponse);
    } catch (Exception e) {
        e.printStackTrace();
        return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR)
                .body("An error occurred while generating message.");
    }
}
```

**Figura 7.2:** Código modificado de API de enriquecer historia de usuario

## 7.2 Pruebas de usabilidad

Las pruebas de usabilidad son fundamentales en el desarrollo de productos, especialmente en el ámbito digital, donde la experiencia del usuario puede ser determinante para el éxito o el fracaso de una aplicación o sitio web. Estas pruebas son esenciales para identificar problemas de interfaz y comprender cómo interactúan los usuarios con el producto, lo que permite realizar mejoras significativas en su experiencia.

### 7.2.1. Objetivos de usabilidad

Aquí se presenta algunos objetivos para realizar pruebas con los usuarios finales, explicando por qué se consideran que estas áreas son fundamentales para el éxito de este proyecto.

- **Facilidad de uso:** Un producto fácil de usar puede reducir la curva de aprendizaje para los usuarios, permitiéndoles comenzar a obtener valor del producto sin sentirse frustrados. La facilidad de uso está directamente relacionada con la adopción y la retención de los usuarios, ya que los productos complicados o confusos tienden a alejar a los usuarios. Por lo tanto, simplificar la navegación y la interacción puede mejorar la experiencia general del usuario y fomentar un mayor compromiso.

- **Satisfacción de usuario:** La satisfacción del usuario es un indicador clave del éxito de un producto. Un usuario satisfecho es más probable que recomiende el producto, vuelva a utilizarlo y proporcione comentarios positivos, lo cual es vital para el crecimiento y la sostenibilidad del producto. La satisfacción abarca todo, desde la estética del diseño hasta la consistencia en el rendimiento y la confiabilidad del producto.

- **Eficiencia de la interfaz:** La eficiencia de la interfaz se refiere a la rapidez con la que los usuarios pueden completar tareas una vez que han aprendido a usar el producto. Una interfaz eficiente minimiza el tiempo y el esfuerzo necesarios para realizar tareas, lo cual es crucial para la productividad del usuario, especialmente en aplicaciones de software destinadas a entornos profesionales. Optimizar la eficiencia puede llevar a una mejora significativa en el rendimiento general del usuario y en la percepción del producto.

- **Capacidad de Respuesta:** La capacidad de respuesta se refiere a la rapidez con la que el sistema reacciona a las entradas del usuario y cuánto tiempo lleva completar las tareas. Un sistema que responde rápidamente a las entradas mejora la percepción de eficiencia y disminuye la frustración del usuario.

### 7.2.2. Planificación de tareas y Análisis de resultados

A continuación, se presenta cómo se llevarán a cabo las pruebas con los usuarios finales, alineada con los objetivos de usabilidad previamente definidos.

En concreto, se seleccionaron a cinco personas en Inditex para participar en esta prueba: el tutor del proyecto, el líder del grupo y tres Product Owners del departamento tecnológico. La elección de estas personas se basó en las siguientes razones:

1. **Tutor del proyecto:** El tutor del proyecto proporciona una perspectiva externa y objetiva. Aunque no está familiarizado con la creación de historias de usuario en Jira, su participación es valiosa para evaluar la usabilidad general y la intuición de la herramienta desde el punto de vista de un usuario nuevo.

2. **Líder del grupo:** Al igual que el tutor del proyecto, el líder del grupo no tiene experiencia previa en la creación de historias de usuario en Jira. Su feedback es esencial para identificar posibles barreras de uso y áreas donde la herramienta puede ser mejorada para usuarios sin experiencia específica en Jira.

3. **Product Owners del departamento tecnológico:** Los tres Product Owners tienen experiencia directa y relevante en la creación de historias de usuario en Jira. Su

participación es fundamental para evaluar la efectividad y la eficiencia de la herramienta en un entorno real de trabajo. Pueden proporcionar insights profundos sobre cómo la herramienta puede integrarse en su flujo de trabajo actual y cómo puede mejorar su productividad y precisión.

En conjunto, estas personas representan una mezcla equilibrada de usuarios novatos y expertos, lo cual permite obtener una visión comprensiva sobre la usabilidad y la funcionalidad de la herramienta. Esta diversidad es crucial para identificar tanto problemas de accesibilidad como oportunidades de optimización para usuarios experimentados.

Posteriormente, se distribuyó una encuesta para que completaran, junto con una tarea específica que debían realizar. Se les pidió que crearan una historia de usuario basada en un requisito específico utilizando Jira y nuestra herramienta. Primero, después de escribir una historia de usuario simple, debían identificar si existía una historia de usuario similar a la que estaban por crear, y luego enriquecerla para generar una historia completa, incluyendo criterios de aceptación. Después de completar estos pasos, debían cumplimentar la encuesta y, opcionalmente, realizar algunos comentarios adicionales.

Al final, se obtuvieron cinco encuestas completadas junto con cuatro sugerencias que señalaban defectos y áreas de mejora. En el siguiente análisis se examinarán los resultados obtenidos:

1. En la figura 7.3, los resultados muestran que la mayoría de los usuarios encuentran la creación y completación de historias de usuario en Jira como una experiencia al menos neutral, y muchos la encuentran fácil o muy fácil. Sin embargo, hay margen para mejorar la usabilidad, particularmente para convertir las experiencias neutrales en positivas.

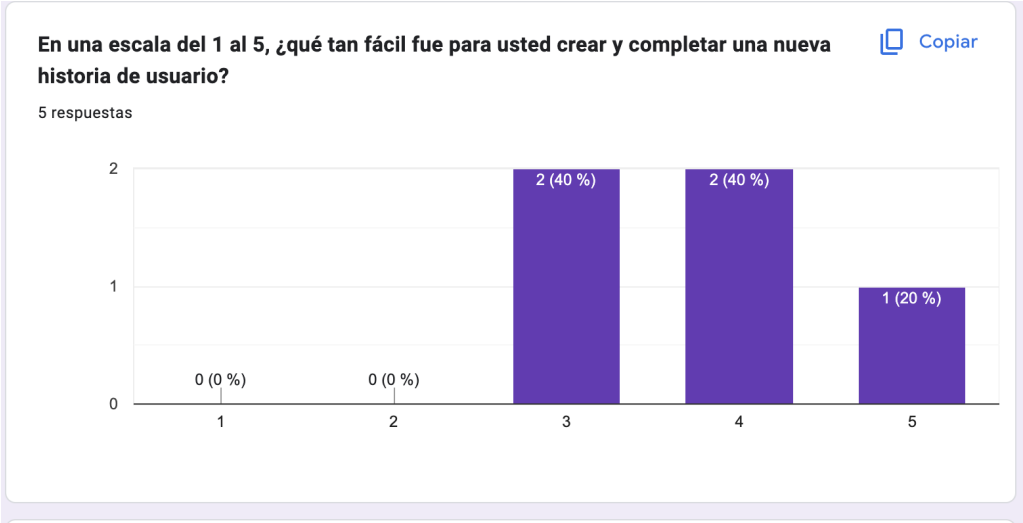


**Figura 7.3:** Pregunta 1 - Facilidad de uso

2. Los resultados de la Figura 7.4 muestran que la mayoría de los usuarios están satisfechos o muy satisfechos con la apariencia general del sistema, y ninguno de ellos está insatisfecho. Sin embargo, todavía hay una parte de los usuarios que se mantiene neutral, lo que indica que el diseño del sistema tiene un potencial de mejora adicional. Al recopilar comentarios detallados de los usuarios y realizar mejoras continuas, se puede aumentar aún más la satisfacción del usuario y la experiencia general del sistema.

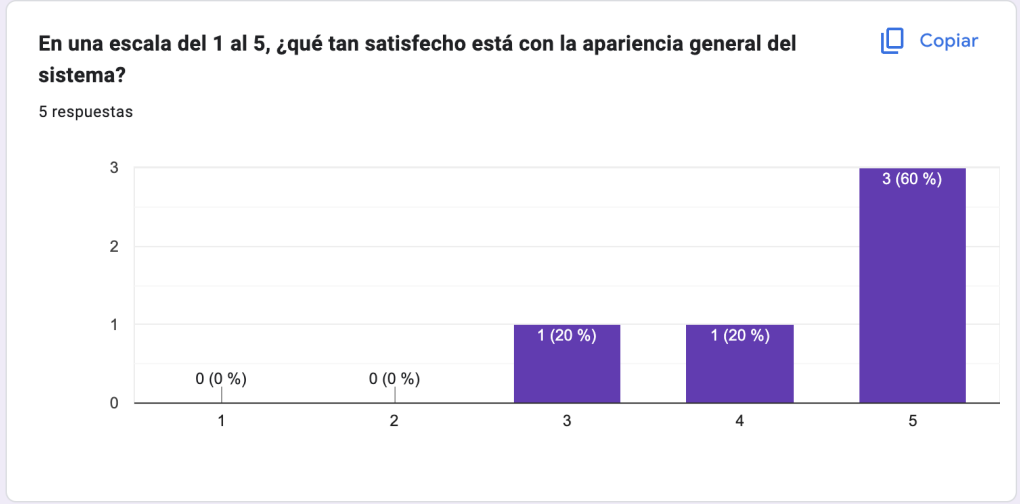


**Figura 7.4:** Pregunta 2 - Satisfacción de usuario

3. Los resultados de la Figura 7.5 indican que la mayoría de los usuarios consideran que la interfaz es eficiente o muy eficiente para tareas como detectar historias de usuario potencialmente duplicadas o completar una historia de usuario. Ningún usuario encontró la eficiencia de la interfaz insatisfactoria. Sin embargo, aún hay usuarios con una actitud neutral, lo que sugiere un potencial de optimización del sistema en términos de eficiencia. Al recopilar comentarios detallados y realizar mejoras continuas, es posible mejorar la eficiencia de la interfaz y la experiencia general del sistema.

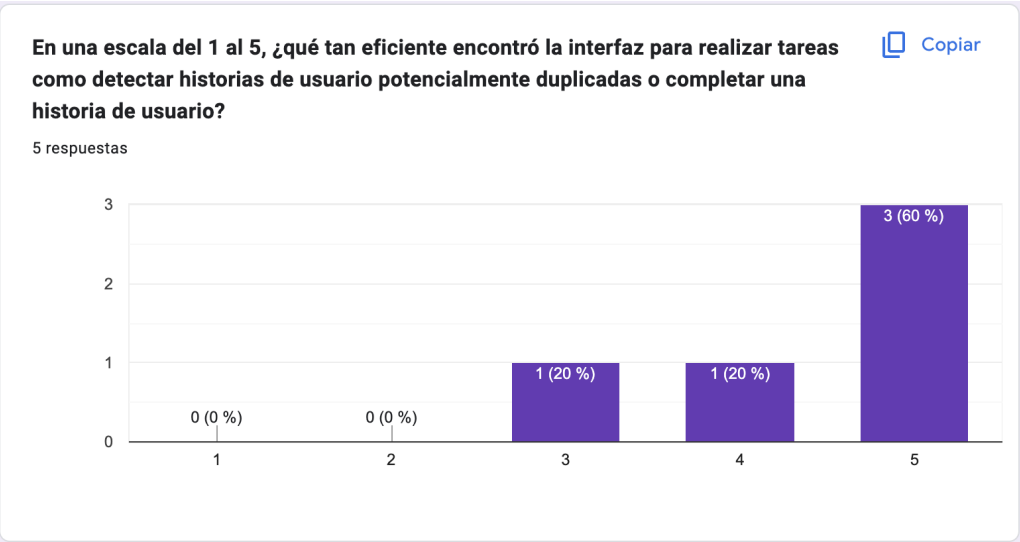


**Figura 7.5:** Pregunta 3 - Eficiencia de la interfaz

4. Los resultados de la Figura 7.6 muestran que las opiniones de los usuarios sobre la velocidad de respuesta del sistema están divididas. Mientras que el 40 % de los usuarios consideran que el sistema responde rápidamente, otro 40 % opina que el sistema es relativamente lento. Ningún usuario otorgó calificaciones extremas, lo que indica que el sistema no tiene defectos ni ventajas claras en cuanto a la velocidad de respuesta.

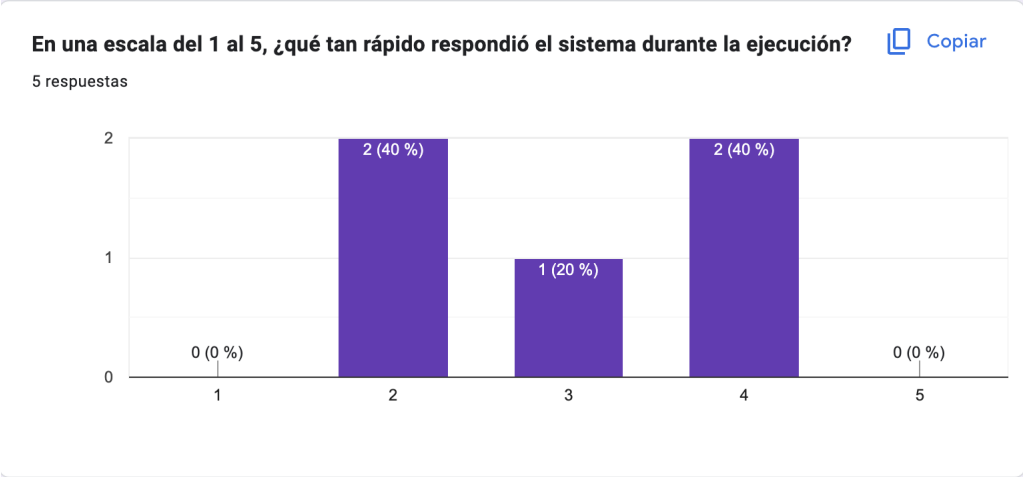


**Figura 7.6:** Pregunta 4 - Capacidad de Respuesta

5. En la última pregunta, se formula una pregunta abierta para recoger sugerencias sobre cómo mejorar el sistema. La imagen 7.7 recopila algunas de estas recomendaciones de los usuarios, destacando distintos problemas y áreas susceptibles de mejora:

   - La primera sugerencia apunta a un problema con una extensión de Google que se desactiva cada vez que se cambia de página, forzando a los usuarios a esperar extensos periodos para obtener datos, lo cual resulta incómodo.
   - La segunda propuesta sugiere un cambio en el diseño de la interfaz: los usuarios preferirían que los enlaces a historias relacionadas se mostrasen como hipervínculos, facilitando así el acceso y la navegación.
   - La tercera sugerencia aborda la integración de herramientas, proponiendo que los botones de funcionalidad en las páginas de Jira se incorporen directamente en la interfaz principal y no en ventanas externas, mejorando la fluidez de la experiencia de usuario.
   - El último comentario se centra en el rendimiento del sistema, indicando que la lentitud en la respuesta del sistema compromete la eficiencia de su uso.

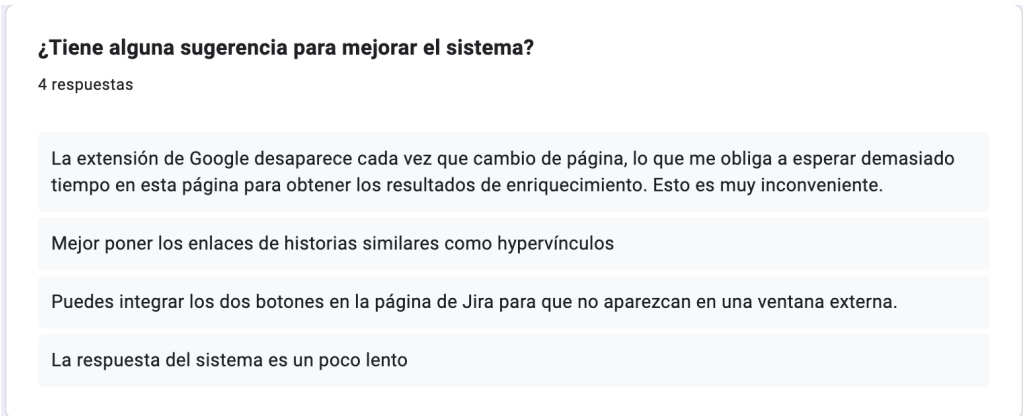


**Figura 7.7:** Pregunta 5 - Sugerencia

Estas sugerencias pueden contribuir significativamente a optimizar la funcionalidad y la experiencia general del usuario. Planeamos abordar estas recomendaciones como parte de los trabajos futuros en la sección 8.2 del documento.

En resumen, aunque el sistema ha sido ampliamente aceptado en facilidad de uso, satisfacción con la apariencia y eficiencia de la interfaz, aún hay áreas significativas que requieren mejoras. A través de una evaluación continua y la integración de sugerencias de los usuarios, se ha identificado un camino claro hacia la optimización del sistema para mejorar la experiencia del usuario. Propuestas específicas, como mejorar la integración de herramientas y la velocidad de respuesta del sistema, serán exploradas y potencialmente implementadas en futuras actualizaciones.

# CAPÍTULO 8
# Conclusiones

En este último capítulo se analiza el cumplimiento de los objetivos, se presentan algunas líneas de trabajo para mejorar esta herramienta en el futuro y se reflexiona sobre la relación de este trabajo con las asignaturas del grado en Ingeniería Informática.

## 8.1 Cumplimiento de los objetivos

Este trabajo ha demostrado la implementación del sistema RAG ha permitido una notable mejora en la calidad y consistencia de las historias de usuario. Los Product Owners ahora pueden acceder a un repositorio de historias predefinidas, lo que les ayuda a definir nuevos requisitos de manera más rápida y eficiente.

Como se puede observar en el capítulo 5, el proyecto ha cumpliendo con todos los objetivos establecidos en la sección 1.2. Además, se ha demostrado el potencial de la inteligencia artificial para mejorar procesos críticos en la ingeniería de software, destacando su eficacia especialmente en la etapa de especificación de requisitos.

En un mundo cada vez más impulsado por la tecnología, es esencial seguir explorando y adoptando nuevas herramientas y técnicas que permitan optimizar los procesos y mejorar los resultados. Este proyecto es un paso adelante en esa dirección, demostrando cómo la IA puede ser un aliado poderoso en la ingeniería de software.

## 8.2 Trabajo futuro

Aunque se han cumplido todos los objetivos establecidos, aún existen áreas que pueden mejorarse según los resultados de la prueba de usabilidad.

1. **Rendimiento y precisión de la generación de respuestas.**

   Dado que ahora estamos utilizando un modelo de lenguaje de gran tamaño en el ordenador local, esto presenta varias limitaciones. Entre ellas, se encuentran las restricciones en la capacidad de procesamiento y almacenamiento, que pueden afectar el rendimiento y la velocidad del sistema. Además, el manejo de grandes volúmenes de datos puede ser menos eficiente en comparación con soluciones basadas en la nube. Por lo tanto, en el futuro, sería mejor conectarse a un modelo remoto utilizando su API para mejorar el rendimiento.

2. **Defecto intrínseco de Google extensión.** La ventana emergente de la extensión de Google se cierra automáticamente al cambiar de página, lo cual resulta muy inconveniente ya que los resultados generados pueden desaparecer. Para resolver esta

situación, se propone integrar esta extensión directamente en la página de Jira, como se muestra en la figura 8.1. Esta integración incluiría la colocación de dos botones al lado del campo de descripción y la visualización de los resultados generados directamente en el cuadro de texto.

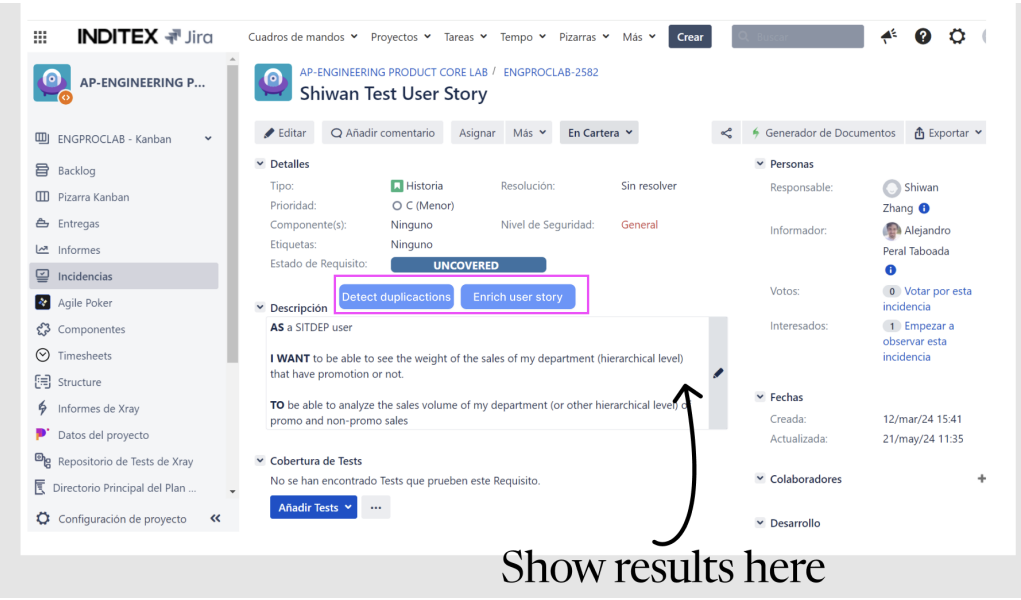


**Figura 8.1:** Mejora de la interfaz

## 8.3 Relación con las asignaturas

En esta sección se examinan las habilidades y conocimientos adquiridos a lo largo del grado de Ingeniería Informática y su aplicación en el trabajo. A continuación, se enumeran las asignaturas que guardan una mayor relación con el proyecto:

- **Análisis y especificación de requisitos(AER):** Esta asignatura proporciona las herramientas y metodologías esenciales para realizar una especificación detallada de los requisitos. A lo largo del curso, desarrollé habilidades para diseñar interfaces de usuario y funcionalidades del sistema que se ajustan precisamente a las necesidades y expectativas de los usuarios, garantizando que cada elemento del software esté en perfecta consonancia con los requisitos consensuados.

- **Ingeniería del software(ISW):** Esta asignatura cubre los fundamentos esenciales de la ingeniería de software, incluyendo el ciclo de desarrollo, la integración y las pruebas. Ha resultado extremadamente útil para comprender las interacciones técnicas entre los distintos componentes del software.

- **Diseño de software(DDS):** Esta asignatura se centró en los patrones de diseño fundamentales de la programación, aplicándolos en proyectos de clase para demostrar su eficacia y practicidad en la resolución de problemas recurrentes que estos patrones están diseñados para abordar.

- **Proyecto de Ingeniería de Software(PSW):** En esta asignatura, se trabaja en equipo para desarrollar un proyecto de software desde cero, atravesando todas las fases esenciales de un proceso de desarrollo ágil. La asignatura ha sido sumamente beneficiosa para mi experiencia práctica en el desarrollo de proyectos de software.

Además, se destacó la importancia de integrar los Objetivos de Desarrollo Sostenible(ODS) en nuestro proyecto, contribuyendo a la creación de un software más sostenible.

- **Mantenimiento y evolución de software(MES):** Este curso se centra en enseñar el mantenimiento, pruebas y gestión de versiones dentro del ciclo de desarrollo de aplicaciones. En este TFG, los conocimientos adquiridos sobre Git y pruebas de integración han sido fundamentales para asegurar la calidad y la evolución del software desarrollado.
- **Integración e Interoperabilidad(IEI):** Esta asignatura se centró en la reutilización e integración de servicios distribuidos, enseñando cómo combinar estos servicios para desarrollar sistemas que funcionen de manera coordinada y eficiente.

Por último, además de los conocimientos técnicos adquiridos en las asignaturas mencionadas, este TFG también ha sido una excelente oportunidad para destacar y aplicar diversas competencias transversales. Habilidades como la comunicación efectiva, la gestión del tiempo y la resolución de problemas complejos han sido cruciales en este proceso. Además, el aprendizaje autodidacta ha sido esencial, especialmente para adoptar tecnologías nuevas y avanzadas que no había manejado previamente. Estas competencias han sido fundamentales no solo para el desarrollo del proyecto, sino también para prepararnos para los desafíos profesionales futuros.

# Bibliografía


[1] *Alberto Casero, «Patrón de inyección de dependencias»,* [Consulta: 2024-05-12]. (s.f.). https://keepcoding.io/blog/patron-de-inyeccion-de-dependencias/

[2] *Alexander Shvets, «Composite Pattern»,* [Consulta: 2024-05-12]. (s.f.). https://refactoring.guru/es/design-patterns/composite

[3] *Alexander Shvets, «Observer Pattern»,* [Consulta: 2024-05-12]. (s.f.). https://refactoring.guru/es/design-patterns/observer

[4] *Alexander Shvets, «Singleton Pattern»,* [Consulta: 2024-05-12]. (s.f.). https://refactoring.guru/es/design-patterns/singleton

[5] *Alexander Shvets, «State Pattern»,* [Consulta: 2024-05-12]. (s.f.). https://refactoring.guru/es/design-patterns/state

[6] *Apache Software Foundation, «Apache Maven»,* [Consultado el 13 de mayo de 2024]. (s.f.). https://maven.apache.org/

[7] *Atlassian, «Jira»,* [Consulta: 2024-05-13]. (s.f.). https://www.atlassian.com/software/jira

[8] Bermejo, M. (2011). El Kanban. *Barcelona, España: UOC*, *8*.

[9] *Brendan Eich, «CSS»,* [Consultado el 13 de mayo de 2024]. (s.f.). https://developer.mozilla.org/en-US/docs/Web/CSS

[10] *Brendan Eich, «JavaScript»,* [Consultado el 13 de mayo de 2024]. (s.f.). https://developer.mozilla.org/es/docs/Web/JavaScript

[11] *CodeIT, "The Importance of Chrome Extension Development", April 2018. [Online]* [Consulta: 2024-05-11]. (s.f.). https://medium.com/@codeit_llc/the-importance-of-chrome-extension-developmentb8cbb9405bf2

[12] Cuconasu, F., Trappolini, G., Siciliano, F., Filice, S., Campagnano, C., Maarek, Y., Tonellotto, N., & Silvestri, F. (2024). The power of noise: Redefining retrieval for rag systems. *arXiv preprint arXiv:2401.14887*. https://arxiv.org/abs/2401.14887

[13] de Zárate, J. M. O., Dias, J. M., Avenburg, A., & Quiroga, J. I. G. (2024). Sesgos algorítmicos y representación social en los modelos de lenguaje generativo (LLM). https://fund.ar/wp-content/uploads/2024/04/Fundar_Sesgos_algoritmicos_y_representacion_social_en_los_modelos_de_lenguaje_generativo_CC-BY-NC-ND-4.0-1.pdf

[14] *Docker, «Docker»,* [Consultado el 13 de mayo de 2024]. (s.f.). https://www.docker.com/

[15] *Edu Salguero, «Arquitectura Hexagonal»,* [Consulta: 2024-05-12]. (s.f.). https://medium.com/@edusalguero/arquitectura-hexagonal-59834bb44b7f

[16] *Facebook, «React»,* [Consulta: 2024-05-13]. (s.f.). https://react.dev/

[17] *Figma, «Figma»,* [Consultado el 13 de mayo de 2024]. (s.f.). https://www.figma.com/

[18] *GitHub Inc., «GitHub»,* [Consulta: 2024-05-13]. (s.f.). https://github.com/

[19] Guevara, D. E. L., & Palomino, N. B. L. S. (2018). Automatización de requisitos: Historias de usuario generadas a partir de un modelo orientado a objetivos basado en el framework i. *Interfases*, (011), 57-72.

[20] *Gustavo Espíndola, ¿Qué son los embeddings y cómo se utilizan en la inteligencia artificial con python?,* [Consultado el 13 de mayo de 2024]. (s.f.). https://gustavo-espindola.medium.com/qu%C3%A9-son-los-embeddings-y-c%C3%B3mo-se-utilizan-en-la-inteligencia-artificial-con-python-45b751ed86a5

[21] Hsia, J., Shaikh, A., Wang, Z., & Neubig, G. (2024). RAGGED: Towards Informed Design of Retrieval Augmented Generation Systems. *arXiv preprint arXiv:2403.09040*.

[22] *INDITEX. Inditex en el mundo.* [Consulta: 2024-05-10]. (s.f.). https://www.inditex.com

[23] *JetBrains, «IntelliJ IDEA»,* [Consulta: 2024-05-13]. (s.f.). https://www.jetbrains.com/es-es/idea/

[24] *Kubernetes, «Kubernetes»,* [Consultado el 20 de mayo de 2024]. (s.f.). https://kubernetes.io/

[25] *L. Abrams, "What are Google Chrome Extensions?* [Consulta: 2024-05-11]. (s.f.). https://www.bleepingcomputer.com/tutorials/understanding-google-chrome-extensions/

[26] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., et al. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, *33*, 9459-9474.

[27] Maida, E. G., & Pacienzia, J. (2015). *Metodologías de desarrollo de software* [Consulta: 2024-05-01]. Universidad Católica Argentina. https://repositorio.uca.edu.ar/handle/123456789/522

[28] Mehta, P. (2016). *Creating google chrome extensions*. Springer.

[29] Menzinsky, A., López, G., Palacio, J., Sobrino, M. Á., Álvarez, R., & Rivas, V. (2018). Historias de usuario. *Ingeniería de requisitos ágil*.

[30] *Meta Inc. and React, «Built-in React Hooks»,* [Consulta: 2024-05-12]. (s.f.). https://react.dev/reference/react

[31] *Meta Inc. and React, «useEffect Hook»,* [Consulta: 2024-05-12]. (s.f.). https://react.dev/reference/react/useEffect

[32] *Microsoft, «TypeScript»,* [Consultado el 13 de mayo de 2024]. (s.f.). https://www.typescriptlang.org/

[33] *Microsoft, «Visual Studio Code»,* [Consulta: 2024-05-13]. (s.f.). https://code.visualstudio.com/

[34] *Naciones Unidas. 2015. Work of the Statistical Commission pertaining to the 2030 Agenda for Sustainable Development : resolution / adopted by the General Assembly* [Consultado el 17 de mayo de 2024]. (s.f.). https://digitallibrary.un.org/record/1291226?ln=en

[35] *Ollama, «Ollama»,* [Consultado el 13 de mayo de 2024]. (s.f.). https://ollama.com/

[36] Oswal, J. U., Kanakia, H. T., & Suktel, D. (2024). Transforming Software Requirements into User Stories with GPT-3.5-: An AI-Powered Approach. *2024 2nd International Conference on Intelligent Data Communication Technologies and Internet of Things (IDCIoT)*, 913-920.

[37] *Overleaf «Overleaf»,* [Consultado el 13 de mayo de 2024]. (s.f.). https://www.overleaf.com/

[38] Pérez Santana, M. (2023). *Diseño y desarrollo de un chatbot con gestión de memoria usando GPT* [B.S. thesis].

[39] *Postman, «Postman»,* [Consultado el 13 de mayo de 2024]. (s.f.). https://www.postman.com/

[40] *PromptPerfect, «PromptPerfect»,* [Consultado el 13 de mayo de 2024]. (s.f.). https://promptperfect.jina.ai

[41] Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. (2018). Improving language understanding by generative pre-training. *OpenAI*. https://www.mikecaptain.com/resources/pdf/GPT-1.pdf

[42] *RFID: así es la tecnología que usan Inditex, El Corte Inglés o Mango* [Consulta: 2024-05-10]. (s.f.). https://www.expansion.com/economia-digital/innovacion/2019/10/16/5da0a62ee5fdea9b6d8b46d7.html

[43] Schwaber, K., & Beedle, M. (2001). *Agile software development with Scrum*. Prentice Hall PTR.

[44] *Spring AI «Spring AI»,* [Consultado el 13 de mayo de 2024]. (s.f.). https://spring.io/projects/spring-ai

[45] *Spring Boot «Spring Boot»,* [Consultado el 13 de mayo de 2024]. (s.f.). https://spring.io/projects/spring-boot

[46] *UNESCO. 2017. Educación para los Obje?vos de Desarrollo Sostenible: obje?vos de aprendizaje.* [Consultado el 17 de mayo de 2024]. (s.f.). https://unesdoc.unesco.org/ark:/48223/pf0000252423

[47] *Zara prueba la realidad aumentada en 120 tiendas del mundo* [Consulta: 2024-05-10]. (s.f.). https://es.fashionnetwork.com/news/zara-prueba-la-realidad-aumentada-en-120-tiendas-del-mundo,957535.html

# APÉNDICE A
# Reflexión sobre los objetivos ODS

**Tabla A.1:** Objetivos de Desarrollo Sostenible

| Objetivos de Desarrollo Sostenible | Alto | Medio | Bajo | No procede |
|---|---|---|---|---|
| ODS 1. Fin de la pobreza. | | | | x |
| ODS 2. Hambre cero. | | | | x |
| ODS 3. Salud y bienestar. | | | | x |
| ODS 4. Educación de calidad. | | x | | |
| ODS 5. Igualdad de género. | | | | x |
| ODS 6. Agua limpia y saneamiento. | | | | x |
| ODS 7. Energía asequible y no contaminante. | | | | x |
| ODS 8. Trabajo decente y crecimiento económico. | x | | | |
| ODS 9. Industria, innovación e infraestructuras. | x | | | |
| ODS 10. Reducción de las desigualdades. | | | | x |
| ODS 11. Ciudades y comunidades sostenibles. | | | | x |
| ODS 12. Producción y consumo responsables. | | | | x |
| ODS 13. Acción por el clima. | | | | x |
| ODS 14. Vida submarina. | | | | x |
| ODS 15. Vida de ecosistemas terrestres. | | | | x |
| ODS 16. Paz, justicia e instituciones sólidas. | | | | x |
| ODS 17. Alianzas para lograr objetivos. | | | | x |

Los Objetivos de Desarrollo Sostenible(ODS)[46] son 17 metas globales establecidas en 2015 por la Organización de las Naciones Unidas[34], diseñadas para promover un futuro mejor y más sostenible en todo el mundo.

Este trabajo está estrechamente vinculado con los siguientes ODS:

- **ODS 4.- Educación de calidad.** Dado que este proyecto ha integrado un modelo de lenguaje de gran tamaño, se promueve el aprendizaje de inteligencia artificial y se incrementa el conocimiento sobre su aplicación. Además, las personas pueden ver los prompts escritos en el código para aprender cómo utilizarlos efectivamente.
- **ODS 8.- Trabajo decente y crecimiento económico.** Este proyecto puede ayudar a los nuevos Product Owners a aprender a editar historias de usuario rápidamente, ya que no es necesario que las editen completamente por sí mismos, solo deben modificar las partes inapropiadas. Esto aumenta la eficiencia en la creación de historias de usuario, permitiendo un trabajo más productivo y decente.

  Además, con la ayuda de esta herramienta, se puede promover la coherencia entre los requisitos de los clientes y el producto final, mejorando la comunicación entre

el cliente y el desarrollador. Esta mejora en la comunicación y en la precisión de los requisitos contribuye a un proceso de desarrollo más eficiente y efectivo, lo que a su vez impulsa el crecimiento económico sostenible.

- **ODS 9.- Industria, innovación e infraestructuras.** Este trabajo presenta una solución innovadora para facilitar la edición de historias de usuario, contribuyendo así a la innovación. Además, se ha descubierto una nueva aplicación de la tecnología de IA en el desarrollo de software, especialmente en la fase de especificación de requisitos. Esto no solo promueve el uso de tecnologías avanzadas, sino que también mejora la eficiencia y precisión en el desarrollo de software.