Universidad del País Vasco Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

GRADO EN INGENIERÍA INFORMÁTICA DE GESTIÓN DE SISTEMAS DE INFORMACIÓN

TRABAJO FIN DE GRADO

JIRAGPT NEXT: ASISTENTE BASADO EN IA PARA GESTIÓN DE PROYECTOS

Estudiante: García Escribano, Joel Moisés

Director: Egaña Aranguren, Mikel

Codirector: López Novoa, Unai

Curso: 2023-2024

Fecha: Bilbao, 7 de noviembre de 2023

Resumen Laburpena Abstract

Este proyecto se centra en el desarrollo de una aplicación que integra modelos grandes de lenguaje o LLMs en inglés con el programa de gestión de proyectos Jira. La aplicación se llama JiraGPT Next y se ha desarrollado a lo largo de unas prácticas realizadas en la empresa LKS Next con el objetivo de crear una aplicación que integrase LLMs con los datos de la empresa. Ofrece una interfaz de lenguaje natural que permite a un usuario realizar preguntas a los LLMs y utilizar datos sobre los proyectos de la empresa alojados en Jira.

Para probar la eficacia de JiraGPT Next, se ha implementado un programa de pruebas que evalúa la eficacia de diferentes LLMs en JiraGPT Next, mediante 70 consultas comunes en castellano e inglés que puede escribir un usuario de Jira en LKS Next. Califica a los LLMs según la precisión de sus respuestas.

Finalmente, para dar una visión a LKS Next sobre las capacidades actuales de los LLMs y sus posibles integraciones en aplicaciones, se ha realizado una comparativa con herramientas actuales que integran LLMs para potenciar la productividad laboral.

Palabras Clave: LLM, Jira, ChatGPT.

Proiektu honen helburua hizkuntza eredu handiak (LLM ingeleses) proiektu kudeaketa Jira programarekin integratzen dituen aplikazio bat garatzea da. Aplikazioa JiraGPT Next deitzen da eta LKS Next enpresan egindako praktiketan garatu egin da eta helburua enpresaren datuak LLM batekin integratzea da. Hizkuntza naturaleko interfaze bat eskaintzen du, erabiltzaileari LLMei galderak egiteko eta Jiran gordetako proiektuen datuak erabiltzeko aukera ematen diona.

JiraGPT Next-en eraginkortasuna frogatzeko froga-programa bat sortu egin da. Jirako erabiltzaile batek idatzi ditzakeen 70 kontsulta (bai ingeleses bai gaztelaniaz) erabiliz, JiraGT-en LLM ezberdinen eraginkortasuna ebaluatzen du. Ondoren, LLMak kalifikatzen ditu erantzunen zehaztasunaren arabera.

Azkenik, LLMak erabiltzen dituzten gaur egungo produktibitatea-sustatzeko tresnekin konparatu egin da. Honen bidez, LKS Nexti oraingo LLMen gaitasunak erakutsi zaizkio eta aplikazioetan izan ditzaketen integrazioei buruzko ikuspegia eman zaio.

Gako-hitzak: LLM, Jira, ChatGPT.

This project focuses on developing an application, JiraGPT Next, which integrates large language models (LLMs) with the Jira project management software. The application was conceived during an internship at the firm LKS Next, aiming to experimentally blend LLMs into a tool that the company employs in its operations. JiraGPT Next offers a natural language interface, enabling users to query the LLM and access company's data stored in Jira.

Furthermore, a benchmark has been designed to test the precision of any LLM within the JiraGPT Next application. This benchmark uses 70 typical queries, in both Spanish and English, that a Jira user would use. LLMs are rated based on their accuracy in their responses.

To enlighten LKS Next about the potential of LLMs and the state of the art of the tools that integrate LLMs, research has been undertaken on tools that use LLMs to bolster employee productity.

Keywords: LLM, Jira, ChatGPT.

Índice

Lista de figuras

Lista de tablas

Lista de acrónimos

LLM Large Language Model: Modelo de red neuronal profunda con miles de millones de parámetros orientado al lenguaje.

NLP Natural Language Processing: Campo de inteligencia artificial que busca dar a las máquinas la capacidad de "comprender" el lenguaje humano.

GPT-3.5 Generative Pre-trained Transformer versión 3.5: Modelo de lenguaje basado en la arquitectura de Transformers desarrollado por Open-AI capaz de generar y comprender texto coherente y diverso.

1. Introducción

En el panorama actual de la inteligencia artificial, el lanzamiento de ChatGPT ha marcado un hito en los asistentes virtuales al capturar la atención de no solo los expertos en inteligencia artificial, sino del gran público. Su auge ha supuesto un cambio de paradigma en la interacción entre humanos y máquinas y millones de personas lo usan a diario [1]. Ha impulsado un enorme aumento de inversión en el campo de los LLMs por parte de multinacionales tecnológicas como Google, Amazon y Microsoft [2]. Detrás de la interfaz de ChatGPT se encuentran los modelos grandes de lenguaje también llamados LLMs en inglés.

Es una herramienta versátil que aborda multitud de tareas relacionadas con el lenguaje. Es capaz de escribir, resumir, razonar sobre textos y más tareas que dependen de la generación o el procesamiento del lenguaje humano [3]. Además, dispone de la habilidad para entender y generar código en una amplia variedad de lenguajes de programación, por lo que una gran cantidad de programadores lo utilizan por su capacidad de generar y procesar código en muchos lenguajes de programación diferentes [4]. También lo utilizan otros profesionales de diversas áreas para facilitar tareas repetitivas o que requieren escribir grandes cantidades de texto.

Ante este escenario se presenta JiraGPT Next, una aplicación desarrollada en la empresa LKS Next orientada a usuarios de la famosa aplicación de gestión de proyectos Jira. JiraGPT Next permite integrar Jira con LLMs aunque por defecto utiliza el LLM de OpenAI GPT-3.5 mediante su API oficial de pago. Está diseñado para poder utilizar otro LLM diferente, como por ejemplo el LLM más avanzado de OpenAI GPT-4 [5].

LKS Next es una empresa de consultoría en tecnología, finanzas y legal. En su parte de consultora tecnológica provee de servicios de software a clientes tanto del sector público como privado. Es una empresa de origen vasco y dispone de 6 oficinas en el Euskadi, aunque recientemente inauguró una oficina más en Madrid con el objetivo de expandirse [6].

JiraGPT Next permite a los usuarios realizar consultas en lenguaje natural sobre todos los datos contenidos en las instancias de Jira con el objetivo de proveer de una interfaz de lenguaje natural a todos los empleados de LKS Next. La dificultad de integrar LLMs en aplicaciones radica en conseguir resultados consistentes y adecuados, ya que los LLMs son modelos no deterministas (impredecibles) que tienen un alto componente de aleatoriedad. El principal objetivo de su desarrollo es aumentar la productividad de los empleados facilitándoles las consultas a Jira.

El mayor desafío del desarrollo de JiraGPT Next ha sido crear las plantillas que se añaden a todas las consultas de los usuarios para que el LLM responda en un formato correcto y la respuesta sea la adecuada. Al interactuar con LLMs es vital que lo que escribe el usuario exprese bien lo que desea [7], por lo que se utilizaron técnicas de Prompt

Engineering[1].

Con el objetivo de evaluar las diferentes plantillas que se han diseñado, además de JiraGPT Next se ha desarrollado un programa de pruebas automatizadas que ejecuta 70 consultas que una persona puede realizar en castellano y en inglés. Se han probado 4 plantillas diferentes y se han registrado los resultados para después concluir cuál es la mejor plantilla y por qué.

Adicionalmente, se ha realizado una investigación sobre posibles herramientas que han sido lanzadas recientemente e integran LLMs para aportar una interfaz de lenguaje natural. Esta investigación se ha emprendido con el fin de proporcionar a LKS Next una perspectiva actualizada sobre las posibilidades de los LLMs y las aplicaciones emergentes que han surgido en los últimos meses. Mediante el desarrollo de JiraGPT Next, el programa de pruebas y la investigación, se busca orientar a LKS Next en posibles futuros desarrollos de la empresa en el mundo de los LLMs.

1.1. Objetivos del TFG

Se han establecido varios objetivos para este trabajo de fin de grado. Los objetivos se han acordado en conjunto con la empresa LKS Next para que el resultado sea acorde a los requerimientos y necesidades de LKS Next.

El principal propósito es el desarrollo de JiraGPT Next, una herramienta diseñada para ofrecer a los empleados de LKS Next una interacción en lenguaje natural con Jira y de esta forma aportar un valor diferencial frente a otras herramientas de productividad. JiraGPT Next debe facilitar la comunicación entre los empleados y la instancia de Jira de la empresa. Además de una mejora en la productividad, debe sentar base para futuros desarrollos de aplicaciones con LLMs.

La creación de plantillas para comunicarse con un LLM dentro de una aplicación es una parte muy importante de JiraGPT Next. Las plantillas se han diseñado siguiendo unos principios que se deben poder aplicar a plantillas de otros futuros desarrollos de aplicaciones con LLMs.

Las pruebas automatizadas deben poder funcionar con otros LLMs para poder evaluarlos. Además, se debe poder cambiar las plantillas para experimentar cambiando las plantillas y los LLMs. Todo esto con el objetivo de poder mejorar JiraGPT Next implementando un LLM mejor que GPT-3.5 como GPT-4.

Finalmente, el estudio de las herramientas que utilizan LLMs actualmente debe servir para proporcionar perspectiva a las personas en LKS Next sobre las capacidades actuales de los LLMs y su integración en aplicaciones.

En resumen, el TFG en conjunto busca dar una visión clara sobre las mejores prácticas, los posibles obstáculos que pueden surgir a la hora de desarrollar una aplicación con LLMs y servir como una guía para el futuro de LKS Next en desarrollos similares.

[1] Práctica para diseñar y optimizar las consultas (prompts) a los LLMs para obtener respuestas más precisas y útiles.

1.2. Objetivos de Desarrollo Sostenible

En esta sección se explica cómo este TFG está enmarcado en el Objetivo 9 relativo a la innovación e industria de los Objetivos de Desarrollo Sostenible establecidos por las Naciones Unidas.

El desarrollo de JiraGPT Next es original y utiliza modelos de inteligencia artificial innovadores como lo son los LLMs. El objetivo del programa es mejorar la accesibilidad e inclusividad de Jira facilitando una interfaz de lenguaje natural para favorecer la comunicación entre el empleado y la instancia en Jira. La innovación en tecnologías de lenguaje natural tiene potencial para hacer todo tipo de aplicaciones más inclusivas para una gran variedad de usuarios.

Adicionalmente a la mejora en accesibilidad, a lo largo del desarrollo se ha trabajado en minimizar los costes y, consecuentemente, reducir la huella de carbono asociada a JiraGPT Next. Como se detallará en la sección dedicada a JiraGPT Next, se ha incorporado una etapa en el programa capaz de reducir hasta el 90 % del coste inicial de un LLM. Esta optimización no solo abarata el coste de uso de JiraGPT Next, sino que también la hace más accesible para personas con pocos recursos.

1.3. Estructura de la memoria

Inicialmente se provee de un contexto para comprender la razón detrás del desarrollo de JiraGPT Next para LKS Next, del programa de pruebas y de la investigación realizada sobre herramientas con LLMs integrados. Después se explica en detalle qué son los LLMs, sus características, riesgos y capacidades y se detalla la planificación de este trabajo de fin de grado.

Finalizadas estas explicaciones, se empieza desarrollando las funcionalidades de JiraGPT Next así como la explicación de su interfaz y la implementación técnica del conjunto de la aplicación. Le sigue la sección sobre el programa de pruebas desarrollado para evaluar cómo rinden diferentes LLMs en JiraGPT Next y después se presenta el análisis llevado a cabo sobre el estado del arte de las herramientas que integran LLMs.

El documento culmina con las conclusiones sobre los objetivos cumplidos en este trabajo, una evaluación sobre el coste de su realización y posibles mejoras que se pueden realizar en un futuro.

2. Contexto

Para poder entender las razones de la realización de este TFG y cómo beneficia a la empresa LKS Next, en esta sección se explicaran las herramientas que se utilizan en los flujos de trabajo para optimizar el desarrollo de software y la entrega de los productos a los clientes Primero se explicará el concepto de gestión de proyectos tal y como se aplica en LKS Next. Después se desarrolla el método CI/CD y finalmente se explica qué es LKS Next y por qué la selección de estas herramientas es esencial para ellos.

2.1. Gestion de proyectos

La gestión de proyectos consiste en organizar y coordinar tareas, recursos y personas con el fin de alcanzar un objetivo en un plazo determinado. En el contexto de una empresa de software como LKS Next, se necesita gestionar los diferentes proyectos de la compañía de forma que toda la información esté centralizada y cualquier jefe la pueda consultar. Entre todas las soluciones de gestión de proyectos, en LKS Next se utiliza Jira, una herramienta de gestión de proyectos utilizada en todo el mundo [8].

En Jira se crean proyectos a los que pueden acceder los desarrolladores y los clientes finales. A las tareas que se asignan en los proyectos se les llama "incidencias", y pueden ser tanto incidencias para resolver un error como incidencias para desarrollar una nueva característica del software. En LKS Next los proyectos tienen fases para las incidencias que comparten con los clientes finales, para que estos puedan ver el desarrollo de las soluciones a los errores o las nuevas características que han pedido.

Estas incidencias se organizan en tableros Kanban, los cuales permiten clasificar las incidencias en diferentes fases del flujo del trabajo. Su mayor ventaja es que ofrecen una visión clara del estado actual del proyecto porque permite ver cuántas incidencias se están revisando, cuántas se están realizando y cuantas han sido realizadas.

Además, Jira permite personalizar cada proyecto porque en ocasiones es necesario tener proyectos con diferentes configuraciones para apoyar flujos de trabajo diferentes. También se puede integrar con otras herramientas que usan las empresas. En el caso de LKS Next, sus proyectos de Jira están integrados con el gestor de repositorios Gitlab.

Gitlab es una plataforma de gestión de repositorios que permite el control de versiones y la colaboración [9]. La integración de Jira con Gitlab permite que cada incidencia en Jira pueda estar relacionada con los cambios realizados en un repositorio en Gitlab. Esto hace que sea más fácil ver cómo se ha resuelto la incidencia e identificar errores que pueden surgir en el código de una forma más accesible.

En cada incidencia se pueden escribir comentarios para poder discutir las soluciones. Los clientes también pueden comentar en las incidencias para cuando no están de acuer-

do con un cambio o para cuando quieren proporcionar feedback a los desarrolladores.

2.2. Metodologías de la empresa LKS Next

En un entorno en constante evolución como lo es el desarrollo de software, LKS Next utiliza herramientas muy conocidas como el repositorio Gitlab, su servicio Gitlab CI/CD y Jira para aumentar la productividad de sus equipos de desarrolladores. Estos 3 servicios son utilizados para implantar el marco de trabajo llamado DevOps, el cual se describe a continuación.

2.2.1. ¿Qué es DevOps?

El término DevOps proviene de la combinación de las palabras Desarrollo (Dev) y Operaciones (Ops). Se trata de una filosofía de trabajo en el campo del desarrollo de software que busca mejorar la colaboración entre los equipos de desarrollo y operaciones.

Los equipos de desarrollo son los encargados de crear software correctamente basado en requisitos definidos. Los equipos de operaciones se encargan de la infraestructura donde se ejecuta el software. De esta forma, poder unir ambos equipos mediante una filosofía de trabajo tiene como objetivo promover una cultura de colaboración continua y adaptable al cambio para mejorar el desarrollo de software en todas sus etapas.

DevOps integra todas las etapas: desde la planificación y la escritura del código, pasando por la compilación, pruebas y despliegue del software, hasta el monitoreo de la aplicación. En la imagen 1 se muestra el ciclo DevOps con todas las etapas del desarrollo.

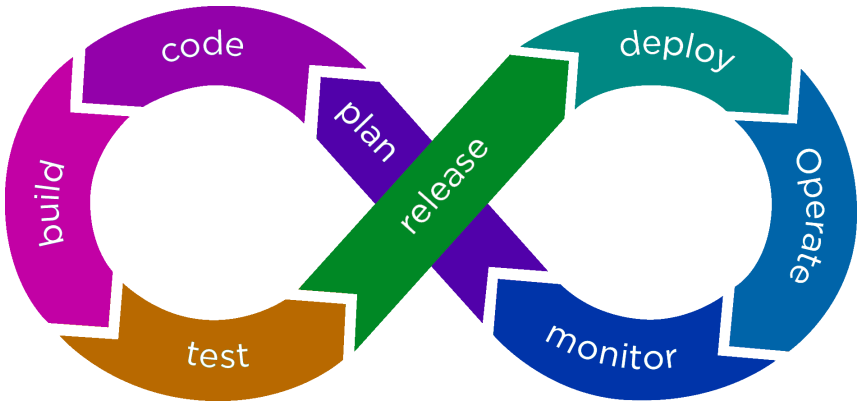

Figura 1: Diagrama mostrando el ciclo del desarrollo de software mediante DevOps [10]. Las etapas que se encuentran en la parte izquierda son de las que se encarga el equipo de desarrollo mientras que las etapas de la derecha son las del equipo de operaciones.

La característica más importante de DevOps es que integra la automatización entre todas las etapas para que las tareas repetitivas se hagan automáticamente sin supervisión humana. De esta forma se eliminan los posibles errores humanos causados por realizar tareas repetitivas y se mejora la eficiencia de todo el proceso en conjunto. Todo el ciclo DevOps se trata de un ciclo de feedback constante en el que si por ejemplo se detecta un error en una aplicación desplegada en producción, se identifica y se resuelve rápidamente. De esta forma se consigue la mejora continua sobre los procesos del desarrollo de software.

2.2.2. Despliegue CI/CD

CI/CD son las siglas para Continuous Integration (Integridad continua) y Continuous Deployment (Despliegue continuo). Es una práctica que forma parte de la filosofía DevOps y se centra en automatizar y optimizar los procesos de desarrollo y despliegue en el desarrollo de aplicaciones. Para el programa desarrollado en este TFG JiraGPT Next se ha utilizado esta metodología de trabajo.

Esta práctica se divide en las dos fases descritas. Primero se ejecuta la fase de integración continua (ámbito desarrollo) y después la de despliegue continuo (ámbito operaciones).

- Integración continua (**CI**)
    - Todos los cambios de código hechos por los desarrolladores se fusionan con el código alojado en el repositorio.
    - El código que sube cada desarrollador se somete a muchas pruebas automatizadas ya definidas que buscan que no se introduzcan errores en las nuevas versiones del código.
    - Fusionando el código y ejecutando las pruebas ambos de forma automática agiliza muchísimo el proceso de desarrollo, ya que los desarrolladores ya no tienen que ejecutar las pruebas manualmente.
- Despliegue continuo (**CD**)
    - Cada vez que se produce un cambio en el código del repositorio se despliega la aplicación de forma automática y sin intervención humana.
    - En esta fase también se ejecutan pruebas y se comprueba que la aplicación se ha desplegado de forma correcta.

Este proceso se realiza de forma constante en el desarrollo con el objetivo de que cada integración y despliegue se ejecute libre de errores y de forma automática en los entornos en los que va a trabajar el software. Por ejemplo, en el caso de la aplicación JiraGPT Next, que es una aplicación en Python, en la parte del despliegue continuo se despliega la aplicación en un entorno con Python instalado junto con las librerías necesarias. De esta forma se detectan errores que ocurren cuando el entorno del desarrollador tiene versiones de Python o de las librerías diferentes a las del entorno de despliegue. Se ha utilizado el servicio de Gitlab CI/CD.

Implantar CI/CD en una empresa de desarrollo beneficia a los equipos de desarrollo y mejora el proceso del desarrollo del software en las dos fases. CI/CD se basa en la automatización y en un ciclo de feedback y de mejora continuo para garantizar su correcto funcionamiento. Por estas razones se ha utilizado CI/CD durante el desarrollo de la aplicación JiraGPT Next.

2.2.3. Integracion de Jira con GitLab en flujos de trabajo de LKS Next

Tras explicar en qué consiste el gestor de proyectos Jira, el gestor de repositorios Gitlab y su servicio CI/CD, se va a resumir la forma en la que en LKS Next se combinan estas 3 herramientas para ser más eficientes en el desarrollo de aplicaciones y obtener mejores resultados.

En LKS Next cada vez que es necesario arreglar un error o implementar una nueva funcionalidad se crea una nueva incidencia en Jira. Al estar Jira integrado con Gitlab, cada incidencia puede asociarse directamente con cambios específicos en el código alojados en el respositorio de código Gitlab.

Por ejemplo, si un desarrollador tiene que agregar una nueva funcionalidad a la aplicación, realizará cambios en el código alojado en Gitlab y quedarán reflejados en la incidencia correspondiente en Jira. De esta forma, se proporciona un historial claro de los cambios en el código y se aumenta la transparencia para los clientes de LKS Next.

El servicio CI/CD de Gitlab también es relevante en el flujo de trabajo. Cada vez que se realiza un cambio en el código, los procesos de integración continua (CI) y despliegue continuo (CD) se ejecutan automáticamente. Gracias a la integración con Jira los resultados de los despliegues se muestran directamente en Jira sin tener que acceder a Gitlab, por lo que los clientes y desarrolladores pueden tener una visión clara en tiempo real del estado de los despliegues.

3. Modelos grandes de lenguaje (LLMs)

A continuación se presenta una introducción muy sencilla sobre la IA Generativa y por qué los LLMs entran dentro de esta categoría de modelos de inteligencia artificial. Después se define lo que son los LLMs, sus parámetros más importantes y se desarrollan sus capacidades y riesgos actuales.

3.1. IA Generativa

Los LLMs se encuadran en la categoría de la IA Generativa debido a su gran capacidad para generar texto. En esta categoría entran tanto LLMs como otros tipos de modelos IA como los generadores de imágenes, generadores de audio y generadores de video. Son los modelos que tienen la capacidad de crear contenido nuevo y original basado en cierta medida en el contenido con el que se ha entrenado.

El principal objetivo de estas IAs es intentar replicar a la mente humana y poder generar contenido con la misma creatividad con la que podría un humano. Por ejemplo, para el logo de la aplicación JiraGPT Next se ha generado la imagen 2 con el modelo generador de imágenes de OpenAI DALL·E 3.

El prompt proporcionado para crear la imagen es el siguiente:

```
Illustration of a logo for 'JiraGPT Next', where the 'Jira' part is stylized in a way
reminiscent of the Jira logo, and the 'GPT' section adopts elements from ChatGPT.
```

La palabra Next del nombre del programa desarrollado viene del nombre de la empresa LKS Next. Como se ve en la imagen 2, la IA generativa ha interpretado la palabra Next como sinónimo de avance y velocidad, y por ello el logo muestra dos flechas hacia delante y líneas que dan la sensación de velocidad.

La IA Generativa puede mejorar la productividad produciendo nuevas ideas y aportando mejoras a los procesos para generar contenido o procesarlo. Por ejemplo, se están utilizando modelos generativos en sectores como el de la medicina, donde se están utilizando para crear estructuras moleculares para desarrollar nuevos medicamentos [11]. También se investiga cómo pueden predecir los modelos generativos la respuesta de diferentes individuos a un tratamiento teniendo en cuenta sus características genéticas [12].

Otro sector en el que va a tomar parte la IA generativa es en el sector de la robótica. Los robots de DeepMind de Google ya integran LLMs con su visión por ordenador para razonar sobre objetos y comprender órdenes complejas en lenguaje natural. Son los modelos RT-2 de DeepMind y ya han demostrado la comprensión de símbolos y razonamiento como un humano, mostrando una gran mejora en comparación con modelos anteriores [13].

Figura 2: Logo de JiraGPT Next hecho por DALL·E 3.

3.2. Introducción sobre los LLM

Los LLMs son redes neuronales[1] de aprendizaje profundo[2] con gran capacidad de procesar y producir lenguaje humano. Han sido entrenados con un conjunto de datos enorme y suponen un gran coste computacional debido a su gran tamaño.

Los LLMs utilizan la arquitectura de red neuronal Transformer. El Transformer es un tipo de red neuronal introducido en 2017 en el artículo titulado .Attention is all you need"[14], muy eficaz en el campo del procesamiento natural y que ha revolucionado el campo debido a su gran eficiencia. Este tipo de red neuronal se basa en mecanismos de atención para tener en cuenta el contexto y permite la paralelización a la hora de entrenarse, por lo que es mucho más rápido que otros tipos.

Además de pertenecer a la categoría de modelos basados en redes neuronales, los LLMs también son modelos de procesamiento de lenguaje natural (NLP en inglés). Al igual que otros modelos que procesan el lenguaje natural, los LLMs pueden generar y procesar el lenguaje. Sin embargo, los LLMs difieren en que su tamaño de entrenamiento y su número de parámetros son mucho más elevados [15]:

- Tamaño de entrenamiento:
  - Los LLMs se entrenan en una cantidad de texto enorme. Por ejemplo, el LLM de OpenAI GPT-3 fue entrenado con 45TB de datos de texto [16].
- Número de parámetros:
  - Los parámetros en los LLMs son la cantidad de valores ajustables que el modelo utiliza para aprender de los datos durante el entrenamiento. Los LLMs tienen una cantidad nunca antes vista; GPT-3 tiene 175 mil millones de parámetros [17].

Tras ser entrenados, los LLMs disponen de parámetros para ajustar las respuestas generadas. El parámetro más relevante en este trabajo es la temperatura, ya que en

[1] Modelos de inteligencia artificial compuestos por neuronas que aprenden de los datos de entrenamiento utilizando pesos.

[2] Categoría de red neuronal con muchas capas de neuronas que permiten aprender representaciones complejas.

JiraGPT Next se le permite al usuario ajustar la temperatura de las respuestas para obtener las mejores respuestas posibles. La temperatura es un valor de 0 a 1 que influye en la aleatoriedad de las respuestas del LLM.

A mayor temperatura, como máximo de 1, el modelo produce respuestas más creativas y menos ancladas a su entrenamiento previo. Sin embargo, las respuestas pueden ser menos coherentes y precisas [18]. Por otro lado, una temperatura baja cercana a 0 hace que el modelo sea mucho más determinista y propenso a ceñirse más a sus datos de entrenamiento. En conclusión, ajustar la temperatura de un LLM permite al usuario controlar el equilibrio entre creatividad y estabilidad en las respuestas del LLM [19].

3.3. Capacidades

Los LLMs tienen una capacidad enorme en diversas tareas relacionadas con el procesamiento del lenguaje natural. A continuación se detallan algunos usos que pueden tener los LLMs en diferentes campos:

- Traducción de idiomas:
  - Son muy buenos realizando traducciones de idiomas en los que han sido entrenados con suficiente texto, como puede ser el caso para el inglés y español.
  - Los modelos más rápidos son muy eficaces en la traducción automática cuando se integran con modelos que convierten el habla a texto.
- Generación de texto:
  - Pueden generar grandes cantidades de texto coherente sobre uno o más temas de forma similar a la de un humano.
  - Gran potencial en la industria de la generación de contenido textual como la prensa. Claro que siempre con la revisión exhaustiva sobre todos los datos por parte de un periodista.
- Clasificación del texto
  - Son muy capaces a la hora de clasificar textos en diferentes categorías como Spam, textos ofensivos o textos no adecuados en ciertos contextos. Por esto suponen un gran avance en la moderación del contenido de texto en comparación con modelos de lenguaje anteriores.
- Generación de resúmenes:
  - Se les puede dar cantidades muy grandes de texto y realizar resúmenes o sacar las ideas más importantes de un texto. Esto es muy útil para las personas que tienen que leer grandes cantidades de texto en el trabajo, ya que resumen los textos más rápidos que los humanos.

3.4. Riesgos de los LLMs

Teniendo en cuenta todo el potencial de los LLMs, puede resultar tentador para muchos pensar que muchas personas podrán dejar de trabajar en labores como la redacción de artículos sobre nuevas noticias o la traducción de textos por ser muy reemplazables por los LLMs. Sin embargo, es importante recordar que, aunque los LLMs pueden automatizar muchas tareas repetitivas y procesar texto a una velocidad no alcanzable por personas, los LLMs cometen muchos errores y carecen de una comprensión real de características puramente humanas como la creatividad o la empatía.

Es por esto por lo que la supervisión humana de las tareas realizadas por LLMs es fundamental por ahora. Como caso reciente, Gizmodo despidió a toda su plantilla que elaboraba los artículos en español para después reemplazarlos por un LLM que se encarga de simplemente traducir las noticias de la edición en inglés al español [20]. Ahora las noticias publicadas en la edición en Español contienen errores de traducción que fácilmente se podrían arreglar si al menos una persona revisase los artículos, pero se ha optado por reemplazar completamente a las personas en pos de una reducción en los gastos de personal.

Además de errores simples, los LLMs generan desinformación ya que es inevitable que den información falsa por verdadera. Esta información puede ser fruto además de sesgos que ha podido adquirir en el entrenamiento. Por ejemplo, existe el riesgo de que el LLM que utiliza Gizmodo para las traducciones se invente información discriminatoria sobre un grupo social y como sus artículos no se revisan por un humano, este artículo tendría consecuencias muy negativas para Gizmodo y para sus lectores.

4. Planificación del proyecto

En esta sección se explica la planificación de este trabajo fin de grado desde su primera etapa hasta la realización de la documentación. En cuanto a las tareas, se utiliza el diagrama EDT en la imagen 3 para descomponer las distintas tareas que se han realizado, conteniendo una planificación de tiempo por cada tarea. Finalmente se muestra un diagrama Gantt para especificar las fechas de cada tarea de forma visual.

Además, esta sección también incluye un análisis sobre el presupuesto previsto en la primera etapa del proyecto y el estudio de las tecnologías usadas en el proyecto.

4.1. Diagrama EDT

Las tareas se han dividido en las 5 etapas durante las que se ha realizado el proyecto; gestión, investigación, implementación del programa JiraGPT Next, implementación de las pruebas y la documentación.

En la siguiente sección se detalla cada etapa y se analizan de forma individual cada una de las tareas que componen el proyecto. Por cada tarea se da una estimación del tiempo para completarla, una descripción y las entradas o salidas que genera.

A continuación se presenta el diagrama EDT con las 5 etapas y todas las tareas pertenecientes a las etapas en la imagen 3.

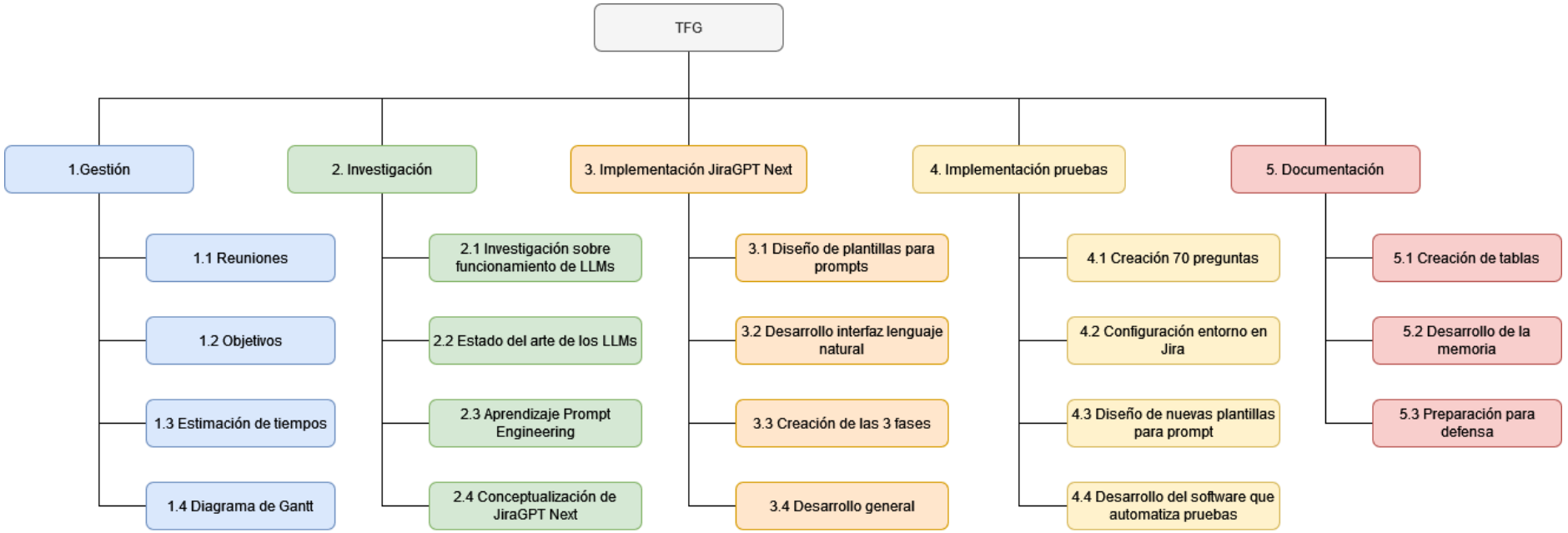

Figura 3: Diagrama EDT

4.2. Descripción de tareas

4.2.1. Gestión

En esta etapa se han definido las reuniones con los tutores Mikel Egaña y Unai Novoa de la universidad y con el tutor de la empresa LKS Next Arkaitz Carbajo, en estas reuniones se han definido los objetivos y se han estimado los tiempos de las tareas a realizar. Mediante las siguientes tablas, se mostrarán los detalles de cada tarea

1.1 Reuniones
Paquete de trabajo: Gestión.
Esfuerzo estimado: 5 horas.
Descripción: Se han realizado reuniones cada 3 semanas con los tutores del TFG. Se han tratado los problemas relacionados con el desarrollo del proyecto, también se han propuesto mejoras y se han planificado las tareas.
Salidas/Entregables: Ninguno.
Recursos empleados: Microsoft Teams.

Tabla 1: Tabla sobre la tarea 1.1. Reuniones.

1.2 Objetivos
Paquete de trabajo: Gestión.
Esfuerzo estimado: 5 horas.
Descripción: Se han definido los objetivos del proyectos y su alcance.
Salidas/Entregables: Anotación de los objetivos del proyecto en el programa de notación Obsidian.
Recursos empleados: Obsidian.

Tabla 2: Tabla sobre la tarea 1.2 Objetivos.

1.3 Estimación de tiempos
Paquete de trabajo: Gestión.
Esfuerzo estimado: 1 hora.
Descripción: Se ha estimado la duración de todo el proyecto y de las tareas a realizar.
Salidas/Entregables: Anotación de la duración de las tareas en el programa de notación Obsidian.
Recursos empleados: Obsidian.

Tabla 3: Tabla sobre la tarea 1.3 Estimación de tiempos.

1.4 Diagrama de Gantt
Paquete de trabajo: Gestión.
Esfuerzo estimado: 4 horas.
Descripción: Se ha creado el diagrama de Gantt para definir la duración de las tareas en un diagrama.
Salidas/Entregables: Diagrama de Gantt.
Recursos empleados: GanttProject.

Tabla 4: Tabla sobre la tarea 1.4 Diagrama de Gantt.

4.2.2. Investigación

En la etapa de investigación se han realizado las tareas relacionadas con el aprendizaje sobre los LLMs como con los conceptos no vistos anteriormente relativos a los LLMs como el Prompt Engineering Además, se ha concretado el desarrollo de la aplicación JiraGPT Next

2.1 Investigación sobre funcionamiento de LLMs
Paquete de trabajo: Investigación.
Esfuerzo estimado: 30 horas.
Descripción: Investigación sobre cómo funcionan los LLMs, sus características principales, sus capacidades así como sus limitaciones y sobre la posibilidad de crear, entrenar o usar un LLM de terceros.
Salidas/Entregables: Anotación en el programa de notación Obsidian.
Recursos empleados: Obsidian.

Tabla 5: Tabla sobre la tarea 2.1 Investigación sobre funcionamiento de LLMs.

2.2 Estado del arte de los LLMs
Paquete de trabajo: Investigación.
Esfuerzo estimado: 20 horas.
Descripción: Investigación sobre el estado del arte de los LLMs. Se ha estudiado las posibilidades de realizar el programa con los LLMs de OpenAI, Anthropic (Claude) y muchos otros LLMs de código abierto.
Salidas/Entregables: Anotación en el programa de notación Obsidian.
Recursos empleados: Obsidian.

Tabla 6: Tabla sobre la tarea 2.2 Estado del arte de los LLMs.

2.3 Aprendizaje Prompt Engineering
Paquete de trabajo: Investigación.
Esfuerzo estimado: 15 horas.
Descripción: Se ha estudiado el Prompt Engineering para saber la mejor forma de desarrollar aplicaciones con LLMs. Es una técnica que no garantiza resultados perfectos pero sí mejora mucho las respuestas de los LLMs.
Salidas/Entregables: Anotación en el programa de notación Obsidian.
Recursos empleados: Obsidian.

Tabla 7: Tabla sobre la tarea 2.3 Aprendizaje Prompt Engineering.

2.4 Conceptualización de JiraGPT Next
Paquete de trabajo: Investigación.
Esfuerzo estimado: 10 horas.
Descripción: Teniendo en cuenta las necesidades de LKS Next y que utilizan Jira para todos sus proyectos, se inició el concepto del programa JiraGPT Next para aumentar la productividad de los empleados.
Salidas/Entregables: Anotación en el programa de notación Obsidian.
Recursos empleados: Obsidian.

Tabla 8: Tabla sobre la tarea 2.4 Conceptualización de JiraGPT Next.

4.2.3. Implementación JiraGPT Next

En esta etapa se ha realizado la implementación del programa JiraGPT Next con sus correspondientes componentes. Desde el diseño de las plantillas para prompt utilizadas en cada fase de la aplicación hasta el desarrollo general para que todos los componentes de la aplicación funcionen correctamente.

3.1 Diseño de plantillas para prompts
Paquete de trabajo: Implementación JiraGPT Next.
Esfuerzo estimado: 50 horas.
Descripción: Se han diseñado las plantillas para los prompts de cada una de las 3 fases. Este proceso ha sido lento, ya que el modelo que oferta OpenAI en su API no devuelve resultados consistentes y ha sido muy difícil encontrar unas buenas plantillas para los prompts.
Salidas/Entregables: Escritura de código en Visual Studio Code.
Recursos empleados: Visual Studio Code.

Tabla 9: Tabla sobre la tarea 3.1 Diseño de plantillas para prompts.

3.2 Desarrollo interfaz lenguaje natural
Paquete de trabajo: Implementación JiraGPT Next.
Esfuerzo estimado: 85 horas.
Descripción: Mediante la librería Streamlit se ha creado la parte correspondiente al Frontend de la aplicación. Se han realizado muchas modificaciones a lo largo del desarrollo para que la experiencia del usuario sea la mejor posible.
Salidas/Entregables: Escritura de código en Visual Studio Code.
Recursos empleados: Visual Studio Code.

Tabla 10: Tabla sobre la tarea 3.2 Desarrollo interfaz lenguaje natural.

3.3 Creación de las 3 fases
Paquete de trabajo: Implementación JiraGPT Next.
Esfuerzo estimado: 120 horas.
Descripción: Creación de las 3 fases de JiraGPT Next de forma modular para que se puedan eliminar o añadir fases en el futuro de una forma más sencilla.
Salidas/Entregables: Escritura de código en Visual Studio Code.
Recursos empleados: Visual Studio Code.

Tabla 11: Tabla sobre la tarea 3.3 Creación de las 3 fases.

3.4 Desarrollo general
Paquete de trabajo: Implementación JiraGPT Next.
Esfuerzo estimado: 80 horas.
Descripción: El desarrollo general trata sobre la eliminación de errores, perfeccionamiento de código y unión de todos los componentes de los que está formado JiraGPT Next para que la aplicación funcione correctamente.
Salidas/Entregables: Escritura de código en Visual Studio Code.
Recursos empleados: Visual Studio Code.

Tabla 12: Tabla sobre la tarea 3.4 Desarrollo general.

4.2.4. Implementación pruebas

Tras desarrollar la aplicación JiraGPT Next se ha creado un programa que realiza pruebas de forma automatizada diseñado para evaluar cómo funcionan diferentes LLMs en la aplicación JiraGPT Next. Las tareas tratan desde la creación de las 70 preguntas, la mejora de las plantillas utilizadas mediante Prompt Engineering hasta el desarrollo del software de pruebas completo.

4.1 Creación de las 70 preguntas en castellano e inglés
Paquete de trabajo: Implementación pruebas.
Esfuerzo estimado: 30 horas.
Descripción: Esta tarea trata la búsqueda de 70 preguntas en castellano e inglés que tengan la dificultad de conseguir respuestas coherentes para en un futuro evaluar otros modelos LLM y su funcionamiento en la aplicación JiraGPT Next.
Salidas/Entregables: Toma de notas en la aplicación Obsidian.
Recursos empleados: Toma de notas en la aplicación Obsidian.

Tabla 13: Tabla sobre la tarea 4.1 Creación de las 70 preguntas en castellano e inglés.

4.2 Configuración del entorno de pruebas en Jira
Paquete de trabajo: Implementación pruebas.
Esfuerzo estimado: 40 horas.
Descripción: Se ha diseñado e implementado el entorno de pruebas en Jira con todas las incidencias distintas unas de otras para que sean recuperadas en el programa de pruebas. Este proceso ha sido largo para asegurar que no hay ningún fallo de configuración.
Salidas/Entregables: Entorno de pruebas configurado en Jira.
Recursos empleados: Jira.

Tabla 14: Tabla sobre la tarea 4.2 Configuración del entorno de pruebas en Jira.

4.3 Diseño de nuevas plantillas para prompt
Paquete de trabajo: Implementación pruebas.
Esfuerzo estimado: 20 horas.
Descripción: Tras realizar pruebas con las plantillas que se han usado anteriormente, estas se han mejorado viendo los resultados que dieron y se han utilizado mejores técnicas de Prompt Engineering para asegurar mejores resultados con los LLMs.
Salidas/Entregables: Escritura de código en Visual Studio Code.
Recursos empleados: Visual Studio Code.

Tabla 15: Tabla sobre la tarea 4.3 Diseño de nuevas plantillas para prompt.

4.4 Desarrollo del software que automatiza pruebas
Paquete de trabajo: Implementación pruebas.
Esfuerzo estimado: 60 horas.
Descripción: El desarrollo del programa que automatiza las pruebas se ha realizado para que las pruebas tengan el menor coste posible con la API de OpenAI y para que en el caso de que ocurra cualquier error de ejecución con una pregunta no se detenga todo el programa y siga realizando las pruebas restantes. Se guardan registros de todas las pruebas realizadas junto con todos los parámetros relevantes.
Salidas/Entregables: Escritura de código en Visual Studio Code.
Recursos empleados: Visual Studio Code.

Tabla 16: Tabla sobre la tarea 4.4 Desarrollo del software que automatiza pruebas.

4.2.5. Documentación

Esta última etapa trata la creación de la documentación final del trabajo fin de grado. Se desarrolla la memoria, se crean las tablas con las preguntas disponibles en el anexo y finalmente se realiza la preparación para la defensa de este trabajo.

5.1 Creación de tablas
Paquete de trabajo: Documentación.
Esfuerzo estimado: 10 horas.
Descripción: Se han desarrollado varias tablas, entre ellas las del anexo con las 70 preguntas realizadas en castellano e inglés.
Salidas/Entregables: Memoria en LaTeX.
Recursos empleados: LaTeX en Overleaf.

Tabla 17: Tabla sobre la tarea 5.1 Creación de tablas.

5.2 Desarrollo de la memoria
Paquete de trabajo: Documentación.
Esfuerzo estimado: 120 horas.
Descripción: Desarrollo de la memoria con todos los apartados presentes en este documento de trabajo de fin de grado.
Salidas/Entregables: Memoria en LaTeX.
Recursos empleados: LaTeX en Overleaf.

Tabla 18: Tabla sobre la tarea 5.2 Desarrollo de la memoria.

5.3 Preparación para la defensa
Paquete de trabajo: Documentación.
Esfuerzo estimado: 30 horas.
Descripción: Creación de la presentación y del guión para realizar la defensa de este trabajo de fin de grado.
Salidas/Entregables: Presentación en Google Slides.
Recursos empleados: Google Slides.

Tabla 19: Tabla sobre la tarea 5.3 Preparación para la defensa.

4.3. Planificación temporal

Explicadas las tareas del proyecto y sus duraciones, a continuación se presenta la tabla 4 con la duración de cada tarea en días y horas y el diagrama de Gantt en la imagen 5, el cual permite visualizar la distribución de las tareas a lo largo de los meses desde mayo hasta noviembre. Algunas tareas se han realizado a la vez que otras, como por ejemplo, el desarrollo de JiraGPT Next y del programa de pruebas; el desarrollo del programa de pruebas comenzó después de empezar el de JiraGPT Next pero después ambos desarrollos se realizaron a la vez.

Nombre	Duracion (dias)	Duración (horas)
1. Gestión	8	**15**
1.1 Reuniones	6	5
1.2 Objetivos	3	5
1.3 Estimación de tiempos	2	1
1.4 Diagrama de Gantt	7	4
2. Investigación	80	**75**
2.1 Investigación sobre funcionamiento de LLMs	80	30
2.2 Estado del arte de los LLMs	30	20
2.3 Aprendizaje Prompt Engineering	25	15
2.4 Conceptualización de JiraGPT Next	10	10
3. Implementación JiraGPT Next	90	**335**
3.1 Diseño de plantillas para prompts	60	50
3.2 Desarrollo interfaz lenguaje natural	90	85
3.3 Creación de las 3 fases	60	120
3.4 Desarrollo general	90	80
4. Implementación pruebas	80	**150**
4.1 Creación de las 70 preguntas en castellano e inglés	30	30
4.2 Configuración del entorno de pruebas en Jira	35	40
4.3 Diseño de nuevas plantillas para prompt	65	20
4.4 Desarrollo del software que automatiza pruebas	75	60
5. Documentación	170	**160**
5.1 Creación de tablas	10	10
5.2 Desarrollo de la memoria	155	120
5.3 Preparación para la defensa	15	30
Total	**170**	**735**

Figura 4: Tabla con la duración de las tareas en horas y días.

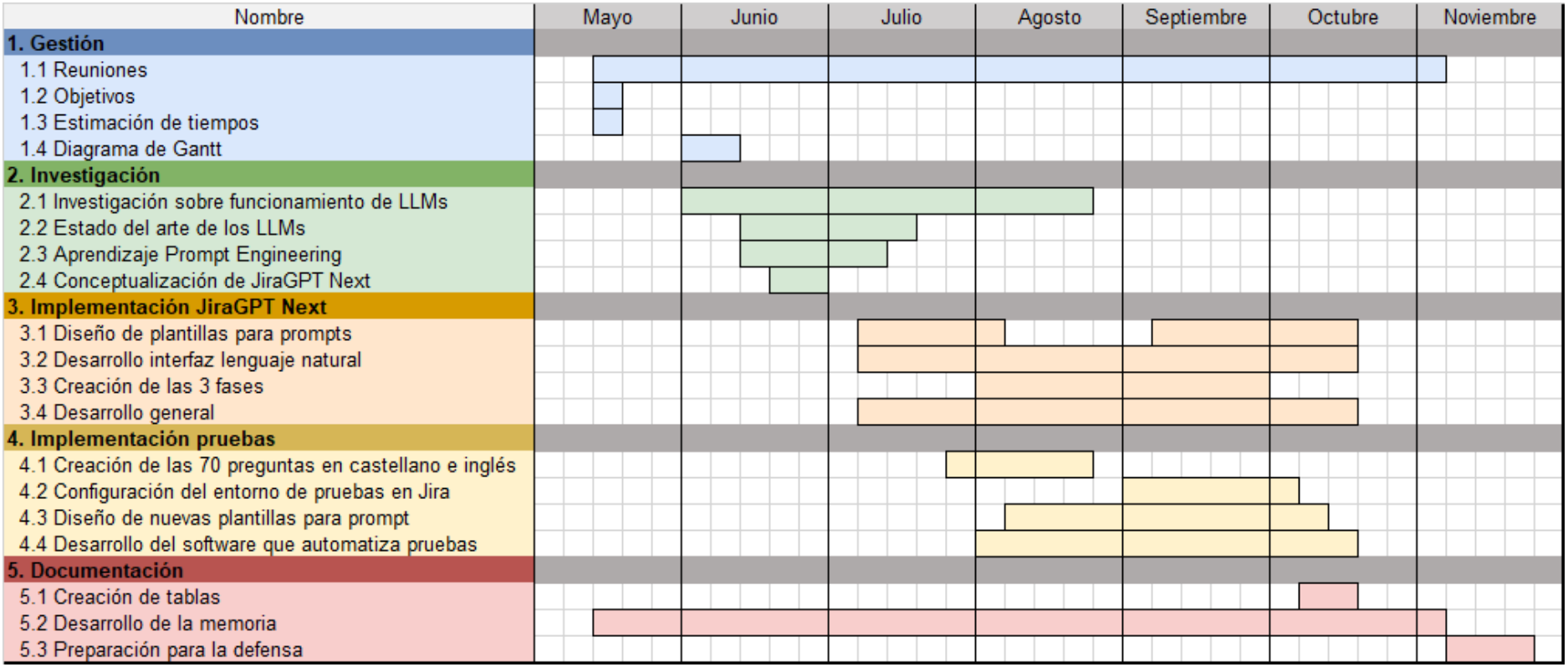

Figura 5: Diagrama de Gantt del TFG.

4.4. Análisis de presupuesto

En esta sección se presentan las estimaciones de los costes totales asociados a la realización de este trabajo. Primero, se especifican los gastos de software y hardware, seguidos de los costos de mano de obra y los costes de la API de OpenAI. Finalmente, se presentan los costes indirectos y el costo total del proyecto.

4.4.1. Costes en software y hardware

Dado que el entorno de desarrollo utilizado es VSCode y es gratuito, el coste en software es nulo

En cuanto al hardware, los recursos empleados son los siguientes:

- Se ha utilizado un portátil Lenovo ThinkPad T14 valorado el 1359€ durante el desarrollo del proyecto en las oficinas de LKS Next. Tiene una esperanza de vida de 9 años y se ha usado durante los 170 días que ha durado el proyecto. Este portátil fue facilitado por la empresa LKS Next como herramienta de trabajo para el proyecto, garantizando así un ordenador potente capaz de llevar a cabo todas las tareas necesarias.

$$\text{Coste de hardware por año} = \frac{1359€}{9 \text{ años}} = 151€/\text{año} \tag{4.1}$$

$$\text{Coste de hardware por mes} = \frac{151€}{12 \text{ años}} = 12{,}58€/\text{mes} \tag{4.2}$$

- Dado que el proyecto durante 7 meses el coste del hardware final es de 88.06€.

$$\text{Coste de hardware final} = 12{,}58 * 7 = 88{,}06€ \tag{4.3}$$

4.4.2. Costes por mano de obra

A continuación se desglosan los costes derivados del salario del estudiante que ha desarrollado este TFG, el tutor de la empresa y los dos tutores de la universidad.

Para calcular el coste de mano de obra del estudiante se ha fijado un salario común para un ingeniero informático recién graduado de 21000€ brutos. Al salario bruto hay que descontarle las retenciones fiscales de IRPF y Seguridad Social. La retención por IRPF es de 2562€ y la cuota de la Seguridad Social es de 1333€, por lo que el suelto neto es de 17105€ anuales

$$\text{Salario neto junior} = 21000€ - 2562€ - 1333€ = 17105€/\text{mes} \tag{4.4}$$

Considerando una jornada laboral de 40 horas semanales y las 735 horas dedicadas al proyecto, el coste por mano de obra y el coste total del estudiante son:

$$\text{Coste por hora} = \frac{17,105€}{40 \text{ horas} \times 4 \text{ semanas} \times 12 \text{ meses}} = 8{,}9€/\text{hora} \tag{4.5}$$

$$\text{Coste total estudiante} = 735 \text{ horas} \times 8{,}9€/\text{hora} = 6541{,}5€ \tag{4.6}$$

Al tutor de parte de la empresa se le ha estimado el sueldo de un ingeniero informático con experiencia, el cual está en torno a los 36500€ brutos [21]. Tras descontarle 6763.5€ de IRPF y 2317.8€ de Seguridad Social, el sueldo neto es 27418.8€ anuales

$$\text{Salario neto tutor de empresa} = 36500€ - 6763{,}5€ - 2317{,}8€ = 27418{,}8€/\text{mes} \tag{4.7}$$

Teniendo en cuenta que el tutor de la empresa ha estado disponible el 80 % de las horas en las que se ha realizado el proyecto (588), su coste de mano de obra total es:

$$\text{Coste total tutor empresa} = \frac{27418{,}8€}{40 \text{ horas} \times 4 \text{ semanas} \times 12 \text{ meses}} \times 588\text{horas} = 8397€ \tag{4.8}$$

Para tener en cuenta el salario de los dos tutores de la universidad se ha contabilizado el sueldo de un profesor pleno en la universidad UPV/EHU, que es de 47521.85€ netos [22].

Se han llevado a cabo reuniones que han sumado un total de 5 horas con los tutores. Sin embargo, este tiempo no refleja todo el esfuerzo y disponibilidad que ambos tutores han aportado al proyecto. Dado que durante el desarrollo del proyecto los dos tutores han estado disponibles durante el mismo periodo que el tutor de la empresa, se estima que han contribuido el 80 % de las horas del proyecto (588). El coste de mano de obra de los dos tutores es:

$$\text{Coste de los dos tutores} = \frac{47521{,}85€}{40 \text{ horas} \times 4 \text{ semanas} \times 12 \text{ meses}} \times 588 horas \times 2 = 29107€ \tag{4.9}$$

Sumados los costes de mano de obra del estudiante, del tutor de la empresa y de los dos tutores de la universidad, el coste total por mano de obra asciende a 44045.5€

$$\text{Coste total mano de obra} = 6541{,}5€ + 8397€ + 29107€ = 44045{,}5€ \tag{4.10}$$

4.4.3. Costes de la API de OpenAI

Para el desarrollo de la aplicación JiraGPT Next el único coste es el del uso de la API de GPT-3.5. En el momento que se planificó, el coste por cada 1000 tokens[1] enviados a la API de GPT-3.5 era de 0.002€

Para poder utilizar la API, la empresa LKS Next destinó un presupuesto de 100€ para hacer pruebas con GPT-3.5 y para desarrollar la aplicación.

Teniendo en cuenta el costo de la API de GPT-3.5, utilizando 100€ se podrían procesar 50.000.000 tokens (100€ / 0.002€ * 1000 tokens). Esa cantidad de tokens enviados a

[1] Fragmento de texto, que puede ser tan corto como un carácter o tan largo como una palabra y que se utiliza para procesar y entender el texto.

la API es difícil de alcanzar mediante pruebas realizadas por una persona, por lo que inicialmente ya era un presupuesto bastante elevado. Sin embargo, no estar limitado por el presupuesto agilizó mucho las pruebas y el desarrollo.

Dado que el desarrollo de la aplicación iba a durar meses y dado que no se pueden estimar las llamadas a la API de OpenAI que se podrían realizar cada día y sus cantidades de tokens, se ha estimado un costo aproximado de 20€.

Este costo podría variar bastante debido a que no se puede calcular con exactitud el número de tokens que va a generar un LLM. Además, para el desarrollo de la aplicación se ha optimizado al máximo el número de tokens que se envían a la API de GPT-3.5 y el número de tokens que genera el modelo utilizando plantillas de prompt avanzadas.

4.4.4. Costes indirectos

Los costes indirectos son los costes que no son fáciles de atribuir a las tareas o etapas de un proyecto. Pueden ser por ejemplo tanto el coste de la luz como del internet en las oficinas de LKS Next. Sin embargo, como es necesario preveer estos gastos, se ha realizado una estimación del 5 % del resto de costes.

$$\text{Costes indirectos} = (88{,}06€ + 6541{,}5€ + 20€) \times 0{,}05 = 332{,}47€ \tag{4.11}$$

4.4.5. Costes totales

Tras calcular los costes de cada categoría, a continuación se muestra el coste total del proyecto:

$$\text{Coste total del proyecto} = 88{,}06€ + 44045{,}5€ + 20€ + 332{,}47€ = 44486{,}03€ \tag{4.12}$$

Y se desglosan los costes de cada categoría en la tabla 20.

Tipo de coste	**Coste (€)**
Software	0
Hardware	88.06
Mano de obra	44045.5
API de OpenAI	20
Indirectos	332.47
Coste total del proyecto	**44486.03**

Tabla 20: Desglose de costes del proyecto por categoría de coste.

4.5. Gestión de riesgos

Durante el desarrollo de un proyecto es inevitable sufrir riesgos que pueden peligrar el buen desarrollo del mismo. Es crucial reconocer los obstáculos que se pueden presentar comprendiendo su probabilidad e impacto y anticiparse a ellos desarrollando un plan de prevención y de contingencia. A continuación, se presentan en detalle los riesgos

identificados que se van a afrontar en el desarrollo de este TFG, junto con los planes de acción correspondientes a cada riesgo.

Riesgo	**Mal funcionamiento de la API de OpenAI.**
Descripción	El proyecto depende mucho de la API de OpenAI que permite realizar consultas al LLM GPT-3.5. Cualquier mal funcionamiento puede resultar en retrasos significativos.
Plan de prevención	Mantenerse actualizado con las actualizaciones oficiales de OpenAI en su página. Tener constancia de otras opciones alternativas a la API de OpenAI como la API de Anthropic para su LLM Claude.
Plan de contingencia	En caso de fallos prolongados, buscar alternativas a la API de OpenAI.
Probabilidad	Baja.
Impacto	Muy alto.

Tabla 21: Riesgo 1.

Riesgo	**Problemas de salud.**
Descripción	Enfermedades o problemas de salud pueden resultar en la incapacidad temporal de trabajar en el proyecto.
Plan de prevención	Mantener una postura saludable en el puesto de trabajo, tener buenos hábitos y descansar adecuadamente.
Plan de contingencia	Si se presentan problemas de salud, se deben comunicar a los tutores y reorganizar el calendario de tareas.
Probabilidad	Baja.
Impacto	Alto.

Tabla 22: Riesgo 2.

Riesgo	**Falta de motivación.**
Descripción	Durante el desarrollo del proyecto se puede perder la motivación para realizarlo y esto se puede ver reflejado en la calidad del trabajo final.
Plan de prevención	Establecer objetivos a corto plazo que sean alcanzables.
Plan de contingencia	Si se trata de una falta de motivación prolongada, es conveniente comunicarlo a los tutores en busca de consejo.
Probabilidad	Media.
Impacto	Alto.

Tabla 23: Riesgo 3.

Riesgo	Mala planificación de las tareas.
Descripción	Una mala planificación puede resultar en retrasos en la finalización y en la acumulación de tareas pendientes.
Plan de prevención	Establecer un calendario de tareas coherente y revisarlo de forma continua. También se debe ajustar según las necesidades del alumno y de los.
Plan de contingencia	Si se detecta una mala planificación, realizar una revisión y reorganización de tareas.
Probabilidad	Media.
Impacto	Medio.

Tabla 24: Riesgo 4.

Riesgo	Problemas con el portátil de trabajo.
Descripción	El portátil proporcionado por LKS Next es esencial para el desarrollo del proyecto. Cualquier fallo técnico puede resultar en pérdida de datos importantes para el desarrollo del proyecto.
Plan de prevención	Mantener copias de seguridad en la nube del código y de la documentación.
Plan de contingencia	En caso de fallo, considerar la reparación o reemplazo del portátil. Asegurarse de tener copias de seguridad actualizadas para restaurar el trabajo.
Probabilidad	Baja.
Impacto	Muy alto.

Tabla 25: Riesgo 5.

5. JiraGPT Next: Integrando LLMs en DevOps

5.1. Introducción

En el capítulo anterior de Contexto 2.1 se ha explicado qué es Jira y la gestión de proyectos de forma general. A continuación se profundiza más en Jira para dar un contexto más aplicado al desarrollo del programa JiraGPT Next.

Jira es una herramienta de gestión de proyectos desarrollada por Atlassian y ampliamente utilizada por empresas de todo el mundo para gestionar todo tipo de proyectos. Implementa un sistema de seguimiento de tareas también llamadas incidencias y permite planificar y monitorizar el progreso de proyectos en tiempo real. Al utilizar metodologías que buscan mejorar la colaboración entre equipos en el proceso de desarrollo y entrega de software, se le considera una herramienta DevOps.

Las incidencias o tareas en Jira son los elementos en los que se registran trabajos específicos que deben ser resueltos en el proyecto. Las incidencias tienen distintos estados como Abierto, En Progreso y Resuelto que indican la fase en la que se encuentran dentro del ciclo de vida de las incidencias. Pueden ser asignadas a una persona, que será la responsable de que se solucione la incidencia. También pueden tener registro de horas y fecha límite de resolución de incidencia.

Una funcionalidad que proporciona Jira de forma gratuita es la posibilidad de buscar y filtrar incidencias específicas utilizando JQL (Jira Query Language), un lenguaje de consultas parecido a SQL. Para utilizar sentencias JQL, Jira provee de una interfaz accesible solo a usuarios avanzados que sepan escribir JQLs y estén familiarizados con los lenguajes de consultas.

La aparición de ChatGPT ha demostrado la capacidad actual de los LLMs y ha abierto la puerta a la creación de interfaces de lenguaje natural. En el caso de Jira, implementar una interfaz de lenguaje natural para la creación de JQLs es una muy buena idea ya que tiene potencial para impulsar la productividad de las personas que lo utilicen tengan o no conocimiento y práctica sobre el lenguaje JQL. Gracias a una interfaz de lenguaje natural que traduzca el lenguaje humano a JQL, el filtrado de incidencias de Jira sería muchísimo más accesible, además de rápido.

Por todo el potencial que puede acarrear crear una interfaz así se decidió desarrollar JiraGPT Next, un asistente virtual para Jira que toma una petición o prompt del usuario para obtener ciertas incidencias y presenta al usuario esas incidencias además de una interpretación adicional si es necesaria.

5.2. Interfaz de usuario

La interfaz de usuario de JiraGPT Next trata de aportar un diseño limpio y sencillo para facilitar el uso al usuario. El diseño integra la marca LKS Next junto con su paleta de colores. En la imagen 6 se muestra una captura de pantalla de la interfaz.

Figura 6: Interfaz de JiraGPT Next.

A continuación se explican en profundidad cada uno de los componentes que conforman esta interfaz de lenguaje natural para Jira en el orden en el que los va a percibir el usuario. En la imagen 7 se marcan los componentes de la interfaz.

Figura 7: Interfaz de JiraGPT Next con marcas de los componentes.

5.2.1. Barra lateral con opciones

Se dispone de una barra lateral que permite al usuario controlar parámetros como la temperatura, el tipo de plantilla que se usará en la generación de la sentencia JQL y el LLM que se utilizará en todas las llamadas.

- La temperatura, tal y como se explica en la sección 5.3.4, mide el nivel de aleatoriedad de la generación de la respuesta por parte del LLM.

- El tipo de plantilla determina si se le dará información al LLM sobre los estados en castellano en los que se clasifican las incidencias de un proyecto (plantilla con los estados de las incidencias en castellano) o si se omitirá esa información (plantilla básica).

- La plantilla básica funciona mejor para recuperar incidencias de proyectos que tienen estados de incidencias en inglés. Sin embargo, fallará con más frecuencia en proyectos con estados en castellano. Esto ocurre ya que el LLM suele suponer que los estados tienen el nombre en inglés por defecto, y si se filtra por estado y los estados del proyecto tienen nombres que no son exactamente los que son por defecto en inglés, la consulta no devolverá ninguna incidencia.

- La plantilla que da contexto sobre los estados que se utilizan da mucho mejores resultados en todos los proyectos de LKS, ya que se informa al LLM de que en los proyectos de Jira se utilizan unos estados en castellano concretos, y por lo tanto es más difícil que se equivoque al generar la JQL.

5.2.2. Prompts de ejemplo

Se han seleccionado 6 ejemplos de consultas que pueden ser comunes y que devuelven buenos resultados en el programa. El usuario puede inspirarse en estos ejemplos a la hora de escribir su consulta.

5.2.3. Campo de texto en lenguaje natural

El usuario debe escribir su consulta en este campo. Por defecto está el primer prompt de ejemplo con transparencia para hacer ver al usuario que debe escribir en este campo.

5.2.4. Botón de enviar incidencias

Después del campo de texto se encuentran 2 componentes: el activador de la pregunta compleja y el botón para realizar la consulta:

- El activador de la pregunta compleja habilita las funciones avanzadas de JiraGPT Next que permiten recibir una respuesta en lenguaje natural. Activarlo es mejor para la experiencia del usuario, pero tendrá un mayor coste.

- El botón con el texto "Buscar incidencias" inicia todo el proceso de enviar la consulta del usuario al LLM y recibir las incidencias de la Fase 1. Si está activada la pregunta compleja se ejecutarán tambien la Fase 2 y la Fase 3.

5.3. Funcionalidades principales

JiraGPT Next está diseñado para tener una interfaz simplificada e intuitiva que elimine la necesidad de crear sentencias JQL para que todas las personas puedan filtrar incidencias de un proyecto y además permite obtener respuestas sobre las incidencias en lenguaje natural.

Las dos funcionalidades que puede ver el usuario son la generación de JQLs mediante lenguaje humano y la capacidad de poder utilizar datos internos empresa como las incidencias para obtener respuestas en lenguaje natural a preguntas complejas. En la figura 8 se muestra las respuestas que recibe el usuario cuando escribe la petición *¿Cuantas tareas creadas este mes están en progreso?*.

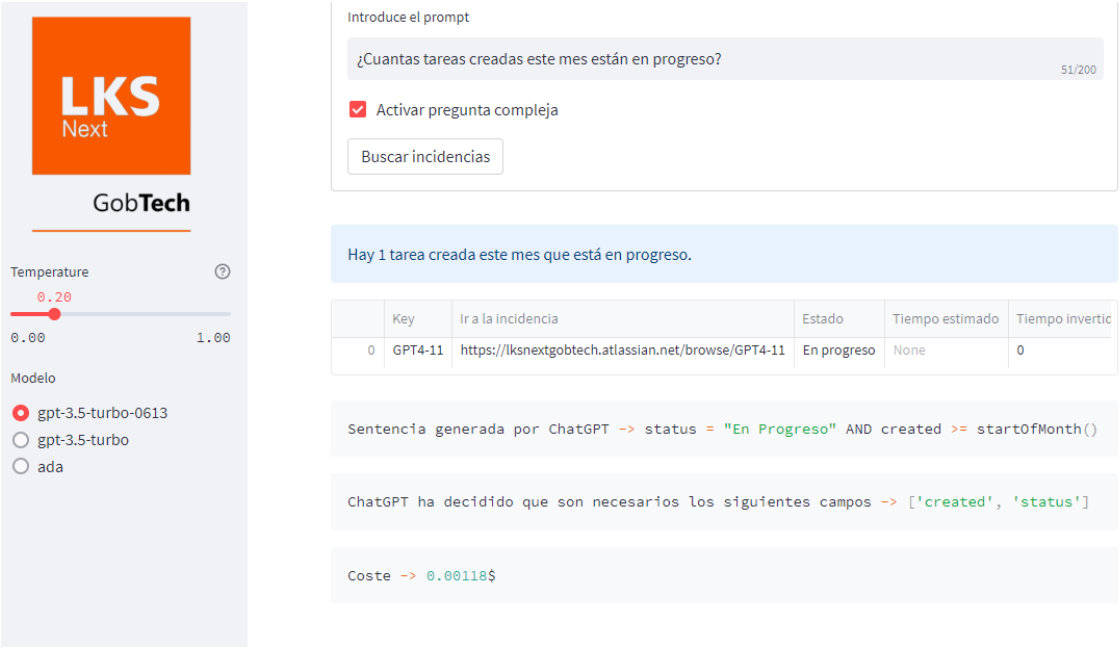

Figura 8: Captura de JiraGPT Next tras realizar una consulta.

Tras escribir la pregunta y pulsar en los botones *Activar pregunta compleja* y *Buscar incidencias*, lo primero que ve el usuario es una respuesta en lenguaje natural a su pregunta. Esta respuesta en lenguaje natural se recibe cuando se activa la pregunta compleja. Activar la pregunta compleja supone un mayor gasto, ya que se hacen 2 llamadas más a la API de GPT-3.5 y se mandan y reciben muchos más tokens[1] que si no se activa la pregunta compleja. Se recuerda que la API de pago de GPT-3.5 tiene coste según el número de tokens utilizados, por lo que el usuario debe tenerlo en cuenta.

Esta respuesta en lenguaje natural pretende resolver dudas de los usuarios de la forma más rápida posible como si ChatGPT estuviese integrado con Jira. Actualmente Jira no dispone de ninguna interfaz de lenguaje natural, por lo que, como en este ejemplo, si el usuario quisiese saber el número de incidencias en progreso creadas este mes, tendría que crear una JQL y contar el número de incidencias devueltas. Todo este proceso cuesta tiempo y presupone que el usuario domina la creación de JQLs. JiraGPT Next propone facilitar enormemente este proceso utilizando las mejores herramientas de LLMs.

Después de la respuesta en lenguaje natural, se muestran las incidencias recibidas

[1] Fragmento de texto, que puede ser tan corto como un carácter o tan largo como una palabra y que se utiliza para procesar y entender el texto.

por la generación de la JQL en una tabla con varias columnas. Se muestra información como el identificador de cada incidencia, su estado, su tiempo de resolución estimado, su descripción, su proyecto y más campos que pueden ser útiles. El propósito de esta tabla es dar al usuario una vista detallada de las incidencias devueltas tras usar el JQL generado.

A continuación se le muestra al usuario información adicional que puede ser útil si la respuesta del programa no ha sido correcta por culpa del LLM utilizado; JiraGPT Next no garantiza que las respuestas sean correctas pues depende en gran medida del comportamiento de un LLM que no es determinista.

Primero se muestra la sentencia JQL que ha generado GPT-3.5 utilizando la pregunta del usuario. En este caso se ha generado la JQL *status = En Progreso AND created = startOfMonth()* que es exactamente la que necesita el usuario. Obtener un JQL correcto depende en gran medida de que la pregunta esté bien escrita. Sin embargo, hay que tener en cuenta que una misma pregunta puede recibir respuestas correctas y erróneas, ya que un LLM puede tener comportamientos imprevisibles y no contemplados que no se pueden prevenir.

Después se muestran los campos de las incidencias que se mandan a GPT-3.5 en la última fase para resolver la pregunta compleja. Más adelante se explicará esto en detalle. Para simplificar esta explicación por ahora, estos campos se utilizan para optimizar al máximo el número de tokens y ahorrar hasta un 90 % en el coste total.

Lo último que se muestra es el coste de la respuesta del programa. El coste depende del número de tokens utilizados en la llamada a la API así como en la respuesta generada. Cuando se activa la pregunta compleja, el programa realiza 3 llamadas a la API de GPT-3.5. Cuando no se activa la pregunta compleja, el coste es mucho menor, ya que sólo se realiza 1 llamada a la API de GPT-3.5. A continuación se explican los dos modos del programa; no utilizando la pregunta compleja (Básico) y utilizando la pregunta compleja (Avanzado).

5.3.1. Modo Básico (Sin pregunta compleja)

Cuando un usuario realiza una pregunta en este modo, JiraGPT Next genera una JQL mediante la API de GPT y la ejecuta en la instancia de Jira. Jira devuelve una lista de incidencias correspondientes a la sentencia JQL y se presentan al usuario mediante una tabla con la información más relevante de cada incidencia y con un enlace a cada una.

Este modo sólo implica la Fase 1 del programa, por lo que es el más rápido y básico, además de ser el de menor coste, pues solo utiliza una llamada a la API de GPT.

En la figura 9 se puede ver que JiraGPT Next sólo muestra las incidencias correspondientes cuando no se activa la pregunta compleja. También se puede observar que el coste de esta respuesta ha sido de la mitad que el de la respuesta usando el Modo Avanzado en la figura 8.

5.3.2. Modo Avanzado (Activando pregunta compleja)

El otro modo disponible y más complejo se activa cuando se selecciona la pregunta compleja. Se necesita utilizar este modo para preguntas que no piden sólo obtener ciertas incidencias aplicando un filtro, sino para preguntas que requieren de una interpretación o cálculos que no ofrece la API de Jira. Tras escribir la pregunta, se le muestra al usuario una respuesta en lenguaje natural de acorde a su pregunta. Este modo utiliza las 3 fases

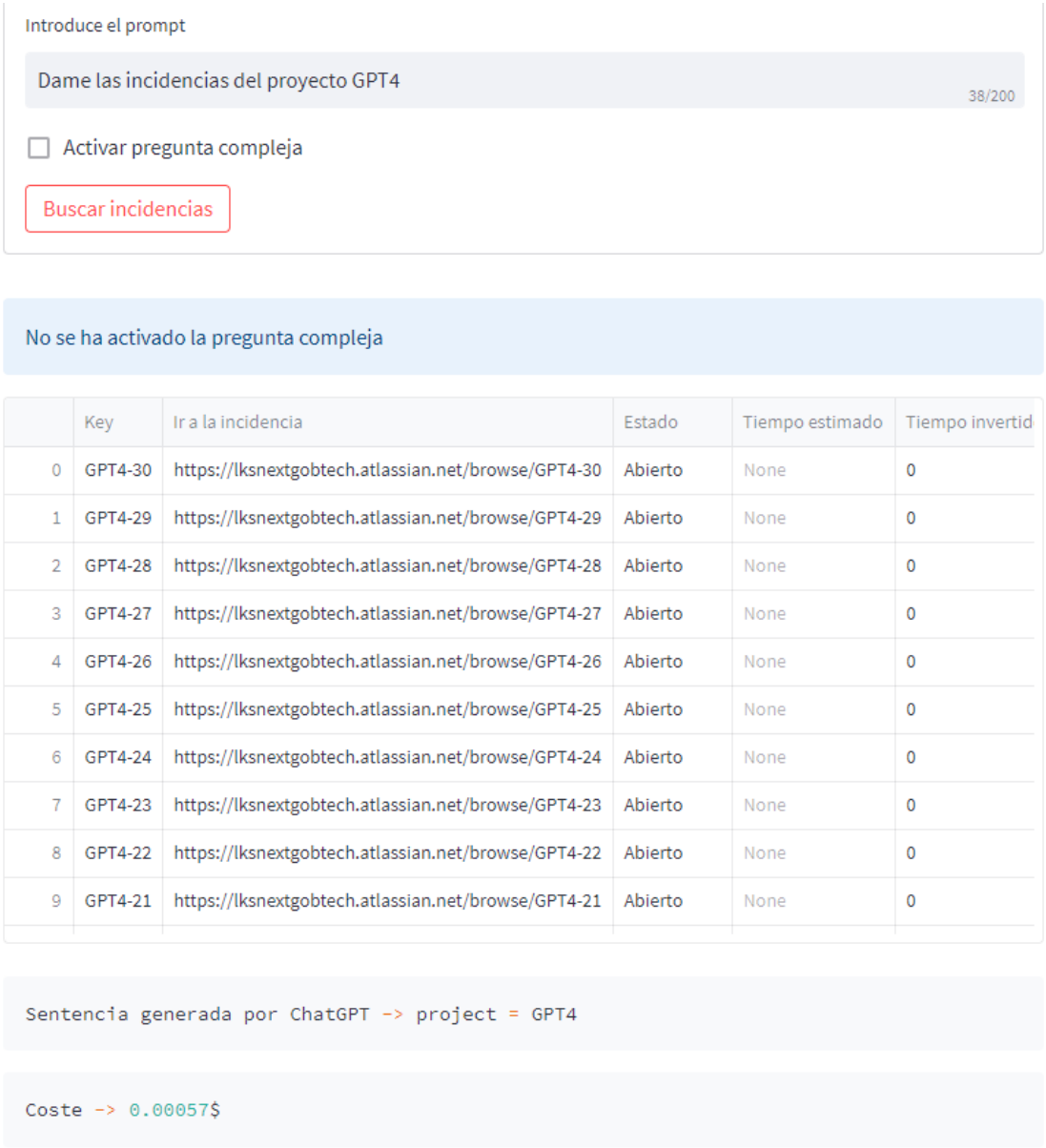

Figura 9: Captura de JiraGPT Next tras realizar una consulta en Modo Básico.

del programa y es mucho más complejo. A continuación se detalla su funcionamiento:

1. Se genera la JQL tal y como se hace en el Modo Básico.
2. Se ejecuta la JQL y se reciben las incidencias de Jira en un JSON.
3. Fase 2: Se eliminan muchos campos del JSON de las incidencias que no son útiles para ningún tipo de pregunta. Sólo se mantienen 21 campos por cada incidencia.
4. Se pregunta a la API de GPT por qué campos del JSON recibido de Jira son estrictamente necesarios para resolver la pregunta compleja en la siguiente fase. GPT responde con una lista de los campos y se crea un nuevo JSON con sólo esos campos por cada incidencia. Se reducen muchísimo el número de tokens que se utilizarán en la Fase 3.
5. Fase 3: Se manda el JSON con las incidencias ya comprimido al máximo y la

pregunta del usuario para que GPT responda con los datos de la empresa a la consulta realizada. GPT responde con lenguaje natural.

Realizar 3 llamadas a la API de GPT con tanta información supone un coste mayor, aunque dado que la experiencia del usuario mejora enormemente al poder recibir una respuesta en lenguaje natural a una pregunta en lenguaje natural, es el modo recomendado. Como se puede ver en la figura 8, una consulta que devuelve una incidencia no ha costado ni un céntimo, por lo que es previsiblemente más barato que el tiempo que gastaría un usuario corriente en escribir la JQL y dar respuesta a su pregunta por si mismo.

Como conclusión sobre los dos modos de JiraGPT Next, mientras que el Modo Básico es rápido y mucho más barato para consultas sencillas que no requieren interpretación del LLM, el Modo Avanzado es mucho más completo e ideal para la mayoría de los casos de uso.

5.3.3. Plantillas utilizadas en cada fase

Antes de mandar la consulta del usuario al LLM, se incluye una plantilla para acompañar la consulta. Esta plantilla aporta contexto al LLM sobre el resultado que se quiere obtener. A continuación se muestran las plantillas que se han usado en cada una de las 3 fases para comunicarse con el LLM. Estas plantillas se han diseñado mediante prueba y error con el objetivo de obtener resultados consistentes.

5.3.3.1. Plantilla de la Fase 1:

Se encarga de generar la sentencia JQL que se ejecutará en la instancia de Jira.

La plantilla usada es:

You are an AI assistant trained on JIRA Query Language. Your task is to translate user requests into precise JQL queries. Remember, the output should only be the JQL query itself, without any additional information or explanations.

Always use the following issue status names in Spanish and never in English: "Open" should be "Abierto", "In Progress" should be "En Progreso", "Resolved" should be "Resuelto", "Approved" should be "Aprobada", "Delivered" should be "Entregado", "Reopened" should be "Reabierto", "Closed" should be "Cerrado".

Please note: If a project name is not specified in the user's request, do not invent or assume a project name. Simply omit the project name from the generated JQL query. For example, given the user's query "Muestra las incidencias en progreso en GPT4", you should return "status = "En Progreso" AND project = GPT4"'

Another example: given the user's query "¿Cuáles son las incidencias de máxima prioridad asignadas a joel.garcia?" you should return "assignee = "joel.garcia" AND priority = "Highest"".

Se muestra un ejemplo de consulta del usuario y el JQL generado en la tabla 26.

Consulta usuario	JQL generado
¿Cuántas personas tienen asignadas tareas en el proyecto GPT4?	`assignee is not empty AND project = GPT4`

Tabla 26: Ejemplo de consulta de usuario y JQL generado

5.3.3.2. Plantilla de la Fase 2:

Pregunta al LLM sobre los campos JSON que se necesitan mandar al LLM en la siguiente llamada para darle la mínima información necesaria para ahorrar en costes y ajustarse al máximo de tokens. En la siguiente fase se mandarán todas las incidencias de Jira filtradas por el JQL y se mandará solo la información contenida en los campos JSON elegidos en esta fase.

La plantilla utilizada es:

Given the user's query, please respond ONLY with the specific fields from the JIRA API's JSON response, separated by commas, that are necessary to fulfill the user's request. Each field should be separated by a comma with no additional explanation or text. For example, if the user's request required the "assignee", "project", and "time" fields, you would respond: "assignee, project, time". Now, please analyze the following user query.

En la tabla 27 se muestra un ejemplo de consulta de usuario y la lista devuelta por el LLM con los campos JSON necesarios.

Consulta usuario	Campos JSON necesarios
¿Cuántas personas tienen asignadas tareas en el proyecto GPT4?	`[assignee]`

Tabla 27: Ejemplo de consulta de usuario y campos JSON necesarios

5.3.3.3. Plantilla de la Fase 3:

Se aportan todas las incidencias filtradas en la Fase 1 y solo se aporta la información de los campos JSON obtenidos en la Fase 2. Además de los datos de las incidencias se incluye la consulta original del usuario para que el LLM disponga de los datos de la instancia de Jira y de la pregunta sobre dichos datos.

La plantilla es:

You are part of an application that interfaces with JIRA. The program is as follows: A user's natural language query asking about issues in JIRA is translated into Jira Query Language using GPT API. The JQL is then executed, and the resulting issues are collected in a JSON. You are given a JSON of the issues and the user's query and you must answer with a response to the user's query.

Se muestra una consulta de usuario y la respuesta del LLM en lenguaje natural en la última fase en la tabla 28.

Consulta usuario	Respuesta en lenguaje natural
¿Cuántas personas tienen asignadas tareas en el proyecto GPT4?	`A partir del JSON de incidencias proporcionado, se puede determinar el número de personas diferentes a las que se les ha asignado una incidencia en GPT4. En este caso son 3 personas.`

Tabla 28: Ejemplo de consulta de usuario y respuesta en lenguaje natural

5.3.3.4. Diagrama de las fases

En la figura 10 se muestra el diagrama de las fases de JiraGPT Next. En el modo Básico sólo se ejecuta la primera fase mientras que en el modo Avanzado se ejecutan las 3 fases.

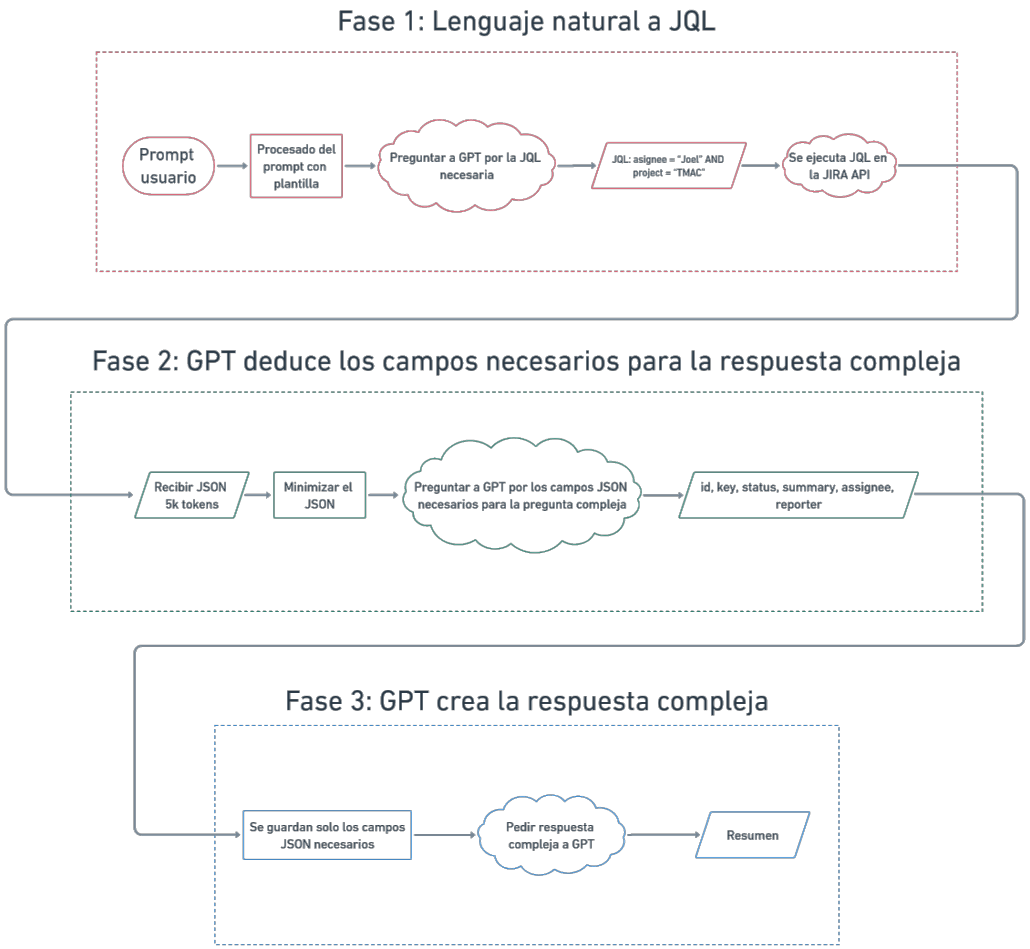

Figura 10: Diagrama de las fases de JiraGPT Next.

5.3.4. Ajuste de la temperatura

Dado que JiraGPT Next está basado en LLMs como GPT-3.5, cada respuesta tiene un considerable nivel de aleatoriedad, ya que los LLMs predicen en base a probabilidades y sus respuestas suelen variar con un mismo prompt. Para permitir que el usuario pueda manipular el nivel de creatividad del modelo para las preguntas en las que el programa no responde correctamente, se ha implementado un deslizador que va desde 0 a 1 y permite controlar el grado de determinismo de las respuestas tal y como permite la API de GPT. En resumen, modificar la temperatura modifica el grado de aleatoriedad de todas las respuestas de la API de GPT.

La temperatura por defecto es 0, ya que es la que mejor tasa de acierto ha dado en las pruebas realizadas, en las que se han probado 70 preguntas en español e inglés variando la temperatura con 10 valores de 0 a 1. Sin embargo, esto no quiere decir que sea la mejor temperatura; depende de la pregunta y del azar, ya que bajar la temperatura no garantiza que las respuestas sean deterministas [23].

A continuación se explica el impacto de modificar el parámetro de temperatura:

- Temperatura baja, cercana a 0
  - Se generarán respuestas más deterministas y menos aleatorias a la vez que creativas. Respuestas más consistentes con un menor grado de variabilidad. Es útil para consultas donde se busca una respuesta más directa y precisa. Para la mayoría de preguntas, suele ser mejor una temperatura muy baja, pero no garantiza que no haya respuestas erróneas. Una temperatura de 0 no elimina la aleatoriedad, aunque la reduzca bastante, ya que el LLM siempre va a contener un elemento de variabilidad [24].
- Una temperatura alta, cercana a 1
  - Las respuestas serán muy creativas y menos deterministas. Las respuestas serán mucho más diferentes utilizando la misma pregunta. Es de utilidad cuando se buscan respuestas muy complejas que no se han hecho correctamente con una temperatura baja.

5.4. Metodologías y tecnologías utilizadas

5.4.1. Git

Git es un sistema de control de versiones usado para facilitar el desarrollo de software y que además garantiza la integridad del código mediante repositorios. Permite ver un registro de las versiones de cada fichero de código y, como permite trabajar de forma descentralizada, cada desarrollador tiene una copia completa del repositorio, aumentando así la eficiencia e integridad del desarrollo.

En este Trabajo de Fin de Grado se utiliza Git en el repositorio privado de la empresa LKS Next en Gitlab por ser una práctica establecida por LKS Next para el desarrollo de software. Esto facilita enormemente el registro del progreso de la aplicación de Jira y además se puede considerar como experiencia profesional valiosa, ya que Git se utiliza en muchísimas empresas en desarrollo de software.

Además, Git es esencial para la utilización de Gitlab CI/CD que se ha llevado a cabo para la aplicación de Jira, ya que Git es la base sobre la que se ejecuta la automatización del despliegue en Gitlab CI/CD.

5.4.2. Streamlit

Se ha buscado implementar una interfaz sencilla e intuitiva para consultar información en Jira mediante lenguaje natural. Para ellos, se ha utilizado el Framework de Python Streamlit. Streamlit provee el Frontend de la aplicación Jira proporcionando una interfaz de usuario intuitiva y fácil de usar, donde los usuarios pueden realizar consultas mediante lenguaje natural y recibir las respuestas.

Streamlit fue creado para facilitar la aplicaciones web de forma rápida y sencilla para aplicaciones que trabajen con datos o Machine Learning. Apenas tiene curva de aprendizaje y simplifica el desarrollo web para científicos de datos que necesiten una aplicación web sin tener que aprender Frameworks como Django, por ejemplo. Dispone de la función Hot Reloading, con la cual no es necesario volver a iniciar la aplicación web para probar cambios en el código; Streamlit se actualiza automáticamente. Esto hace el desarrollo web más rápido aún.

Sin embargo, como el mayor objetivo de Streamlit es la facilidad del desarrollo, en el desarrollo de la aplicación Jira han habido dificultades para crear funcionalidades avanzadas para las que no está diseñado Streamlit. Por ejemplo, inicialmente se querían mostrar las incidencias encontradas en 3 columnas. Streamlit dispone de la funcionalidad para dividir contenido en columnas. Sin embargo, no se puede garantizar que los contenidos de las columnas se puedan alinear de forma horizontal cuando tienen distinto tamaño, por lo que cuando había varias incidencias, estas se descuadraban y provocaba un gran desajuste en la interfaz.

Finalmente se ha optado por visualizar las incidencias en una tabla con columnas. Utiliza menos espacio, es más simple y por lo tanto es más adecuado para el programa, ya que le da más importancia a la respuesta en lenguaje natural, que es lo más relevante de JiraGPT Next.

5.4.3. Python

Se ha utilizado el lenguaje Python para desarrollar la aplicación de Jira porque, además de sencillo, es un lenguaje de alto nivel con un gran ecosistema de bibliotecas, especialmente las relacionadas con LLMs. Es frecuente utilizar Python para proyectos que involucran LLMs, ya que las librerías de OpenAI y la librería LangChain se utilizan más en Python que en otros lenguajes como Javascript.

5.4.4. Github Copilot

La herramienta de programación con IA Github Copilot utiliza un LLM entrenado en miles de millones de líneas de código [25] y proporciona una gran ayuda a los desarrolladores en la escritura de código.

Analiza el contexto del archivo que se está editando y proporciona sugerencias de código, genera código y ayuda a corregir errores teniendo en cuenta todo el archivo. En el desarrollo de este TFG ha sido muy útil para tratar con librerías en Python, aunque

ha sido especialmente valioso para la manipulación de objetos JSON. Escribiendo una explicación de lo que se quiere conseguir con un JSON, Github Copilot ha sugerido múltiples soluciones que no son nada sencillas, y sin usarlo habría sido más difícil el desarrollo de JiraGPT Next.

También ha sido interesante utilizarlo para poder experimentar cómo los LLMs están siendo integrados en herramientas de generación de código. Otra razón de peso para usar Github Copilot ha sido que gracias a la licencia de estudiante, no ha supuesto ningún coste. Sin la licencia de estudiante tiene un coste de 10€ al mes.

5.5. Implementación técnica

5.5.1. Estructura de los archivos de la aplicación

La aplicación ha sido diseñada para poder ser actualizada en un futuro con un LLM mejor que GPT-3.5 como GPT-4. Tiene una estructura muy modular que garantiza un mantenimiento fácil en caso de errores y actualizaciones. Cada componente de la aplicación tiene un propósito específico y en caso de ocurrir una excepción es fácil encontrar la razón.

A continuación se explican los componentes o ficheros python de la aplicación, después se muestra el diagrama los archivos python explicados

5.5.1.1. Ejecución secuencial de las fases de la aplicación

- **streamlit.py**
  - Es la interfaz de usuario escrita en el framework Streamlit, un framework en Python orientado a ingenieros de datos. Es necesario ejecutar este archivo para desplegar la aplicación web.
  - Conforma la interfaz de usuario y se comunica con otros archivos mediante objetos JSON.
- **ejecutar_todas_las_fases.py**
  - Gestiona las llamadas a cada una de las 3 fases del programa de forma secuencial y ordenada.
  - Cuando no se ha activado la pregunta compleja solo se ejecuta la Fase 1. Cuando se activa la pregunta compleja se activan las 3 fases.
  - Al terminar manda toda la información a la aplicación web.

5.5.1.2. Fases 1, 2 y 3

- **fase1.py**
  - Recoge la pregunta escrita por el usuario, la incluye en la plantilla para la generar el JQL, le pide al LLM que genere el JQL, lo corrige cuando contiene fallos y finalmente ejecuta la JQL en la API de Jira.
- **fase2.py**

- Recibe el JSON de Jira con las incidencias, elimina muchos campos innecesarios de forma automática y le pregunta al LLM por los campos necesarios para poder optimizar mucho más el JSON. Procesa la respuesta del LLM y después de corregirla por si tiene fallos, la convierte a una lista de palabras.

- **fase3.py**

  - Comprime aún más el JSON con las incidencias de Jira manteniendo solo los campos JSON que ha dicho el LLM que necesita. Después le realiza la pregunta compleja al LLM aportando las incidencias.

- Finalmente devuelve la respuesta a la pregunta compleja.

5.5.1.3. Archivos funcionales

- **procesado_texto.py**

  - Este archivo tiene múltiples responsabilidades, desde gestionar las plantillas y la compresión programática del JSON de JIRA hasta corregir errores en el formato de la JQL generada.
  - Aplica la plantilla a la pregunta del usuario en cada una de las 3 fases. En cada fase se realiza una consulta diferente al LLM, por lo que es necesario utilizar un prompt diferente en cada fase, siempre incluyendo la pregunta del usuario.
  - Se encarga de realizar correcciones a los errores más comunes que ocurren cuando el LLM genera el JQL. Por ejemplo, GPT-3.5 es propenso a responder añadiendo "JQL: " antes de la instrucción JQL. Es necesario corregir el error pues a la API de JIRA se le tiene que enviar el JQL sin ningún carácter de más o devolverá error de JQL.
  - También aplica correcciones a errores que suelen ocurrir en la generación por parte del LLM de los campos necesarios del JSON para responder la pregunta compleja. A la vez que realiza las correcciones crea un nuevo JSON con solo los campos que el LLM ha decidido que son necesarios de las incidencias para responder la pregunta compleja. Esta optimización puede ahorrar hasta un 90 % de los costes totales del programa, ya que evita mandar muchísima información no necesaria que gastaría una gran cantidad de tokens en la Fase 3.
  - Comprime el JSON con las incidencias recibidas de JIRA eliminando campos que no sirven en ningún caso y crea un nuevo JSON que ocupa mucho menos y cuesta menos tokens para que la pregunta de la Fase 3 suponga un coste muchísimo menor.

- **llamadas_gpt.py**

  - Ejecuta las llamadas a la API de GPT, gestionando las interacciones con el LLM.
  - Llama a la API de GPT para obtener el JQL, para obtener los campos JSON necesarios para la pregunta compleja para optimizar el objeto JSON con las incidencias y para generar la respuesta a la pregunta compleja.
  - Además, tiene en cuenta el coste de cada llamada al LLM según su coste por token.
  - Gestiona las excepciones que ocurren cuando se sobrecarga la API de OpenAI.

- **llamadas_jira.py**

- Centrado en las interacciones con JIRA, este archivo gestiona todas las llamadas a la API de JIRA y la autenticación con el email y la clave de la API de Jira del usuario.
- Envía a Jira las consultas JQL generadas por GPT ya corregidas.
- Como la API de JIRA tiene un límite de incidencias devueltas por cada llamada, se realiza la paginación; se realizan varias llamadas secuenciales a la API de Jira para obtener todas las incidencias.

A continuación se muestra el diagrama con los ficheros Python del proyecto en la imagen 11.

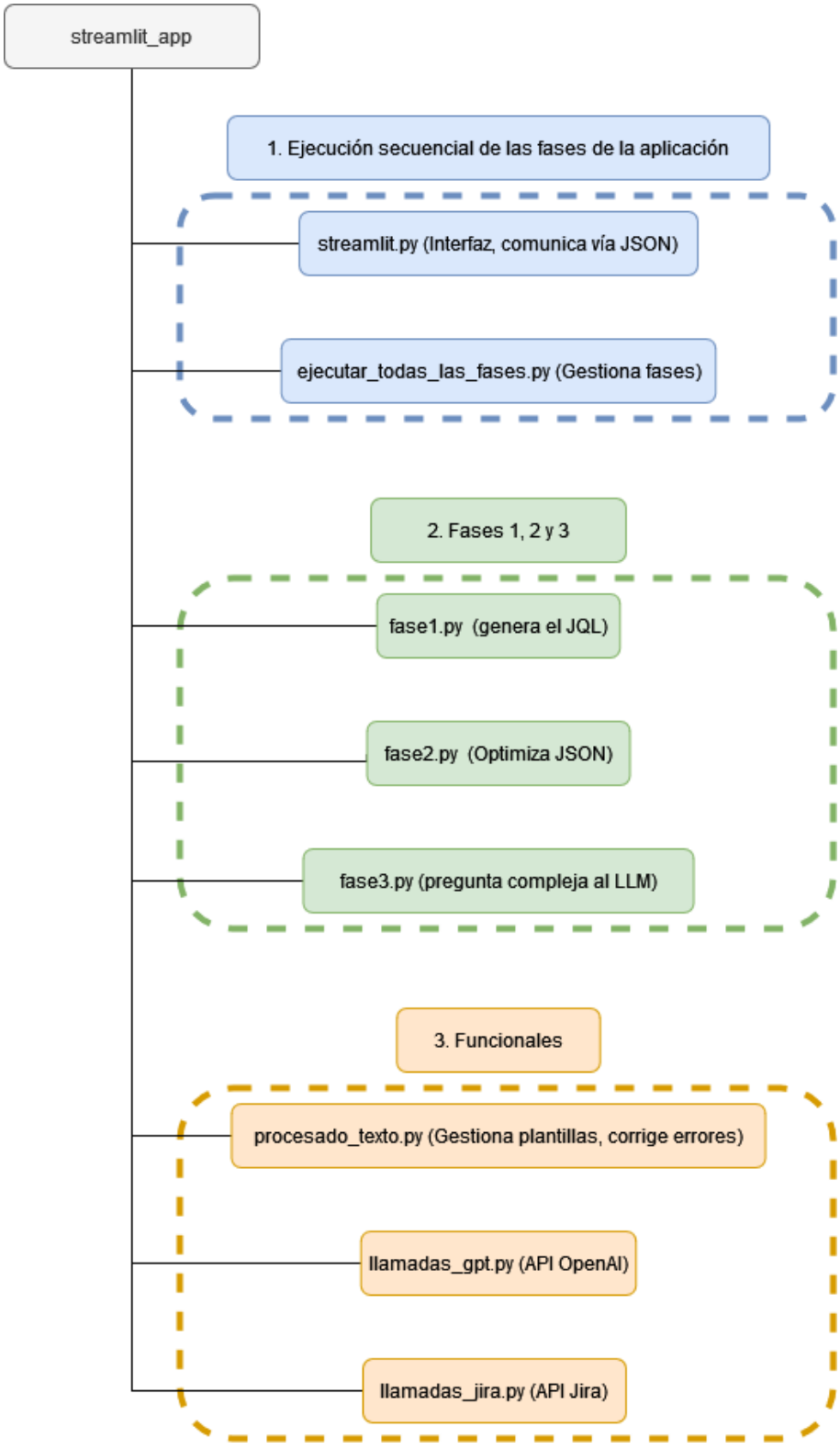

Figura 11: Diagrama con la estructura del proyecto.

5.6. Seguridad y privacidad

Al tratarse de una aplicación orientada a empresas, JiraGPT Next tiene en cuenta la seguridad y privacidad de los datos para garantizar su protección.

No se utiliza la información para ningún propósito que no sea enviarla a la API de la empresa OpenAI. OpenAI garantiza que no utiliza la información utilizada en su API para entrenar sus modelos, aunque sí que permanece en sus servidores durante un tiempo indeterminado.

Es importante mencionar que OpenAI gestiona la información que retiene con estrictas políticas. ya que posee certificaciones de seguridad y privacidad de información como GDPR, SOC2 y SOC3 [26].

5.7. Prompting vs Fine-tuning

Cuando se inició el desarrollo de JiraGPT Next se tuvieron en cuenta dos enfoques para el diseño de la interacción con el LLM: prompting y fine-tuning. Al comienzo se estudiaron ambas posibilidades para ver cuál era la más viable y con más puntos a favor que en contra. A continuación se explican ambos enfoques y por qué se eligió la opción del prompting frente al fine-tuning.

5.7.1. Prompting

Este enfoque consiste en añadir en cada prompt que se enviará al modelo toda una serie de instrucciones con detalles de cómo debe responder el LLM. Además, se pueden añadir ejemplos de pregunta-respuesta para mejorar las posibilidades de una buena respuesta [27].

Como se incluyen instrucciones y ejemplos, cada prompt tiene un alto número de tokens y por ende puede llegar a ser caro a largo plazo y tener mucho tiempo de generación de respuesta.

Sin embargo, es una forma más accesible de desarrollar aplicaciones con LLMs, ya que no hace falta entrenar un modelo con cientos de ejemplos.

5.7.2. Fine-tuning

Consiste en entrenar a un modelo ya entrenado como GPT-3 para que el formato de sus respuestas se adecúen a las necesidades del programa. Dicho de otra forma, se entrena al LLM con un conjunto de datos específico para especializar su comportamiento.

Esto es útil para no tener que incluir instrucciones en el prompt sobre cómo debe ser el formato de la respuesta y también para no tener que poner ejemplos para mejorar la precisión del LLM en un caso de uso específico. Tras realizar fine-tuning a un modelo con ejemplos de pregunta-respuesta, este obtiene tiene cientos de ejemplos de cómo responder y por ello no necesita más ejemplos ni instrucciones.

Permite ahorrar costes a largo plazo, ya que los prompts constan de muchísimos menos tokens al incluir sólo la consulta del usuario y ni instrucciones ni ejemplos.

La capacidad de realizar fine-tuning via API sólo estaba disponible para GPT-3, y este LLM ha dado muy malos resultados en las pruebas realizadas en la generación de sentencias JQL, por lo que se ha prescindido de realizar fine-tuning. GPT-3 muy rara vez respondía con una sentencia JQL; la mayoría de veces devolvía explicaciones o información inventada, por lo que se decidió no invertir más tiempo en adaptar el modelo para la generación de JQLs. En su lugar se optó por utilizar GPT-3.5.

Tiempo después de terminar todo el desarrollo, OpenAI habilitó el servicio de fine-tuning de GPT-3.5, que es el LLM que se ha utilizado en este programa mediante prompting.

5.7.3. Conclusión

- Se ha utilizado el prompting por ser la única opción viable en el momento de desarrollar JiraGPT Next. No estaba disponible el fine-tuning para GPT-3.5 y, aunque sí lo estaba para GPT-3, el modelo funciona muy mal en la generación de JQLs.
- Si hubiese estado disponible la capacidad de fine-tuning para GPT-3.5 al momento de desarrollar JiraGPT Next se habría evaluado el modelo con fine-tuning y en caso de tener muchas mejores tasas de acierto, se habría utilizado en vez de el prompting.
- Tal y como se explica en futuras mejoras, gracias al entorno de pruebas desarrollado para la evaluación de LLMs en esta aplicación, se podrían evaluar tanto LLMs a los que se les ha realizado un fine-tuning como a los que no para sacar conclusiones actualizadas sobre si merecería la pena usar prompting o fine-tuning en dichos modelos.

6. Evaluación de generación de JQL usando GPT-3.5

Todas las aplicaciones que utilizan LLMs se enfrentan a desafíos relacionados con la aleatoriedad y falta de determinismo de los modelos. No se puede garantizar que un LLM responda consistentemente de la misma manera ante consultas idénticas, y esto supone un problema para estas aplicaciones.

En este TFG se busca estudiar las capacidades de los LLMs actuales herramientas, y tras desarrollar la aplicación JiraGPT Next que utiliza LLMs para integrarlos con Jira, se ha decidido diseñar e implementar un entorno de pruebas para evaluar cualquier LLM en la aplicación. El entorno de pruebas se ha diseñado teniendo en cuenta los grandes avances que ocurren en el campo de los LLMs y su rápida evolución; constantemente se lanzan LLMs mejores que responden mejor a los prompts.

El entorno de pruebas puede utilizar cualquier LLM del que se disponga una API. En las pruebas presentes en el TFG solo se ha utilizado GPT-3.5, pero utilizar otro LLM como GPT-4 supone solo cambiar la dirección de la API del modelo.

Se ha diseñado un entorno de pruebas automatizado que contiene 70 preguntas bilingües en castellano e inglés y devuelve una métrica de acierto. Solo se evalúa la primera fase del programa que trata la generación de sentencias JQL para obtener las incidencias correctas, ya que es la que genera sentencias JQLs mediante lenguaje natural y la generación de JQLs correctos se puede comprobar de forma automatizada. Las otras dos fases no se pueden comprobar, ya que tienen infinidad de posibilidades.

Las pruebas consisten en probar que cada pregunta enviada al LLM contenida en una plantilla genere una consulta JQL que, una vez ejecutada en una instancia de Jira, acabe devolviendo exactamente ciertas incidencias. Si devuelve exactamente las incidencias que debe devolver, se marca la pregunta como correcta. Si se equivoca y no devuelve todas las incidencias o devuelve incidencias que no debería, se marca como incorrecta. En cada prueba se realizan cada una de las 70 preguntas en orden y sin repetición.

La plantilla es la información adicional que se envía al LLM además de la pregunta. Sirve para darle instrucciones y ejemplos al LLM. Se han realizado 4 pruebas, cada una cambiando la plantilla para experimentar con las técnicas de prompt-engineering y evaluar su rendimiento.

No se evalúa que el JQL coincida con uno ya establecido, ya que el LLM puede generar un gran número de sentencias JQL que sean correctas pero que no hayan sido contempladas. La mejor forma de evaluar las preguntas es tal y como se ha explicado; comprobando que se devuelven las incidencias correctas.

El porcentaje de acierto que pueden obtener las pruebas está determinado por

el modelo utilizado, por la plantilla que se utiliza para resolver la pregunta y por un parámetro que se explicará más adelante. Para obtener las plantillas que conforman el prompt final se ha utilizado mucho este entorno de pruebas, ya que permite saber qué plantillas funcionan mejor con el caso de uso de Jira solamente ejecutando el programa de pruebas y sin tener que realizar pruebas manualmente.

6.1. Estado del entorno de LKS Next

Dado que JiraGPT Next se ha creado con el objetivo de ser una herramienta útil para la gestión de proyectos en LKS Next, inicialmente se han tenido en cuenta las características de los proyectos que se están desarrollando actualmente en la empresa.

A continuación se describen las características de los proyectos importantes de Jira de LKS Next

- El máximo número de tareas en los proyectos es de 40 tareas, ya que los equipos de desarrollo no son excesivamente grandes.
- Generalmente las tareas están asignadas a una persona, aunque es posible que no estén asignadas a nadie.
- En las tareas se añaden descripciones y resúmenes, por lo que se han formulado preguntas que trabajan con estos dos campos de texto.
- Es común utilizar fechas límite para la realización de las tareas. Se han creado preguntas para comprobar fechas límite.
- Siempre se imputan las horas que ha hecho cada desarrollador para luego poder cobrárselas al cliente, por lo que se han realizado preguntas que tratan sobre carga de trabajo u horas imputadas por una persona.
- Se utilizan las prioridades de tareas de Jira, así que se han incluido en las preguntas.

6.2. Diseño de las preguntas

6.2.1. Por nivel

Cada pregunta se clasifica en un nivel y hay 3 niveles según la dificultad. Están balanceadas para que no haya demasiadas ni pocas preguntas de un solo nivel.

1. Preguntas que filtran incidencias según un sólo campo:
    - Tratan sobre obtener incidencias.
    - Son el 34 % de las preguntas.
    - "Lista todos los tickets creados en agosto de 2023.".
2. Preguntas que filtran incidencias según dos o más campos:
    - Tratan sobre obtener incidencias.
    - Son el 34 % de las preguntas.

- "Dame las incidencias aprobadas en agosto de 2023".

3. Preguntas complejas que requieren interpretación adicional y que se resuelven en la Fase 3:
    - No importa el número de campos que se filtran; la dificultad reside en que el LLM no se confunda e intente resolver directamente la pregunta del usuario. El LLM debe devolver una JQL para obtener las incidencias necesarias para resolver la pregunta compleja en la Fase 3.
    - Son el 32 % de las preguntas.
    - "¿Cuántas incidencias se han movido a "En progreso" en agosto de 2023?"
        - El LLM debe responder con un JQL que devuelva las incidencias que han sido actualizadas a un estado en un periodo de tiempo. No debe responder con un número ya que ni dispone de esa información ni se puede incluir un contador de incidencias en una sentencia JQL.

Las preguntas de los niveles 1 y 2 solo tratan de obtener ciertas incidencias. Las del nivel 3 implican que el LLM tenga que suponer que hay más pasos posteriores para resolver la pregunta compleja y que sólo debe responder con un JQL.

6.2.2. Por tipo

Antes de crear las preguntas, se investigó sobre las capacidades de filtrado de la API de Jira mediante sentencias JQL con el objetivo de diseñar unas pruebas que fuesen difíciles y a la vez pudiesen ser realizables por un LLM actual como GPT-3.5.

Las preguntas creadas se pueden clasificar en distintos tipos según el criterio de filtrado de las incidencias:

- Por estado de incidencia:
    - El filtrado por estado es útil para poder saber de forma rápida el progreso de una incidencia y del proyecto en conjunto.
    - Se prueba especificando el nombre del estado exacto en comillas y sin comillas. Esto se hace para ver cómo responde el LLM en las preguntas en inglés que involucran estados cuando los estados del proyecto están en castellano.
    - "Muestra las incidencias en progreso"
    - "Muestra las incidencias en el estado "En Progreso""
    - "Show me all issues in progress"
    - "Show me all issues in status "En Progreso""
    - Estas 4 preguntas tienen la misma solución como JQL.
- Por fecha
    - Poder filtrar incidencias según fechas límite o de creación ayuda a los equipos a planificar mejor.
    - "Muestra todas las incidencias que deberían acabar en agosto de 2023"
- Por persona asignada

    - Ser capaz de filtrar por asignaciones de tareas a personas permite ver la carga de trabajo de cada miembro del equipo.
    - "Muéstrame todas las incidencias asignadas a joel.garcia"
    - Una limitación de estas preguntas es que se necesita el nombre de usuario de Jira exacto, y si este tiene espacios el JQL generado tiene más posibilidades de ser incorrecto.
    - "Show all issues assigned to "Alex Novoa""

- Por prioridad
    - Poder identificar incidencias por su prioridad permite obtener las tareas que deben resolverse primero, por lo que es un filtro muy útil.
    - "Muestra las incidencias con alta prioridad"
    - El LLM puede confundirse y necesitar el nombre de la categoría de la prioridad, que es en inglés.
    - "Muestra las incidencias con prioridad "High"" tiene más posibilidades de ser acertada.

- Por contenido
    - En un proyecto es muy valioso poder filtrar incidencias según el contenido o el texto que contienen las mismas. Por ejemplo, un jefe de equipo puede querer obtener todas las incidencias relacionadas con el frontend. Para ello realizaría una consulta para filtrar las tareas que contengan la palabra frontend en su título o resumen.
    - "Encuentra todas las incidencias con "base de datos en el resumen""

- Por proyecto
    - Se han hecho bastantes variaciones que consisten en añadir el nombre de un proyecto en el filtrado ya que GPT-3.5 tiene muchos problemas a la hora de tenerlo en cuenta. Esto es algo que no ocurre con GPT-4, ya que se han hecho pruebas con ChatGPT con el modo de GPT-4 y no tiene problemas resolviendo estas consultas JQL.
    - "¿Cuántos tickets están resueltos en el proyecto GPT4?"

- Otros criterios
    - Hay preguntas que filtran según otros criterios.
    - Según un campo personalizado: "¿Cual es el estado de la tarea con campo personalizado Custom_id con valor GPT4-30? ¿A quién ha sido asignado?"
    - "Lista los tickets en los que joel.garcia ha votado"

6.3. Diseño del entorno de pruebas en Jira

A continuación se explica el diseño del entorno de pruebas que se ha creado en Jira.

Se trata de un proyecto con el nombre GPT4 y al que se tiene que tener acceso como usuario para ejecutar las pruebas. El usuario que vaya a ejecutar las pruebas no debe tener acceso a ningún otro proyecto, ya que algunas preguntas no filtran por proyecto y se obtendrían incidencias de otros proyectos ajenos al proyecto GPT4 cuando no se debe.

Se ha utilizado el usuario joel.garcia para que sea el usuario que interactúe en las pruebas. Este es el usuario al que se le asignan las incidencias y quien crea las incidencias. Si se quiere utilizar otro usuario, se puede cambiar en el diseño de las incidencias.

Cada pregunta tiene asignadas una o más incidencias que se han tenido que crear de forma muy estricta para que cada una de las 70 preguntas tenga exactamente las incidencias asignadas que debe y ninguna más por error. En la imagen 12 se muestra el diagrama con dichas incidencias, que en total son 20. Es importante distinguir entre incidencias que implican cambio de estado y las que no, ya que las instrucciones de las que necesitan un cambio de estado deben seguirse en el orden de estados descrito. Si no se hace en el orden escrito, las preguntas asignadas a esas incidencias siempre darán error.

Los números dentro de cada incidencia son los identificadores de las preguntas en las que se deben recibir dicha incidencia. Solo se muestra la información necesaria para la creación del entorno de pruebas en Jira. Por ejemplo, de la primera incidencia que se puede ver en el diagrama, GPT4-5, sólo pone que el tiempo imputado debe ser mayor que el estimado. No se menciona que deba estar en ningún estado de incidencia, por lo que se debe crear la incidencia en Jira y no cambiar su estado.

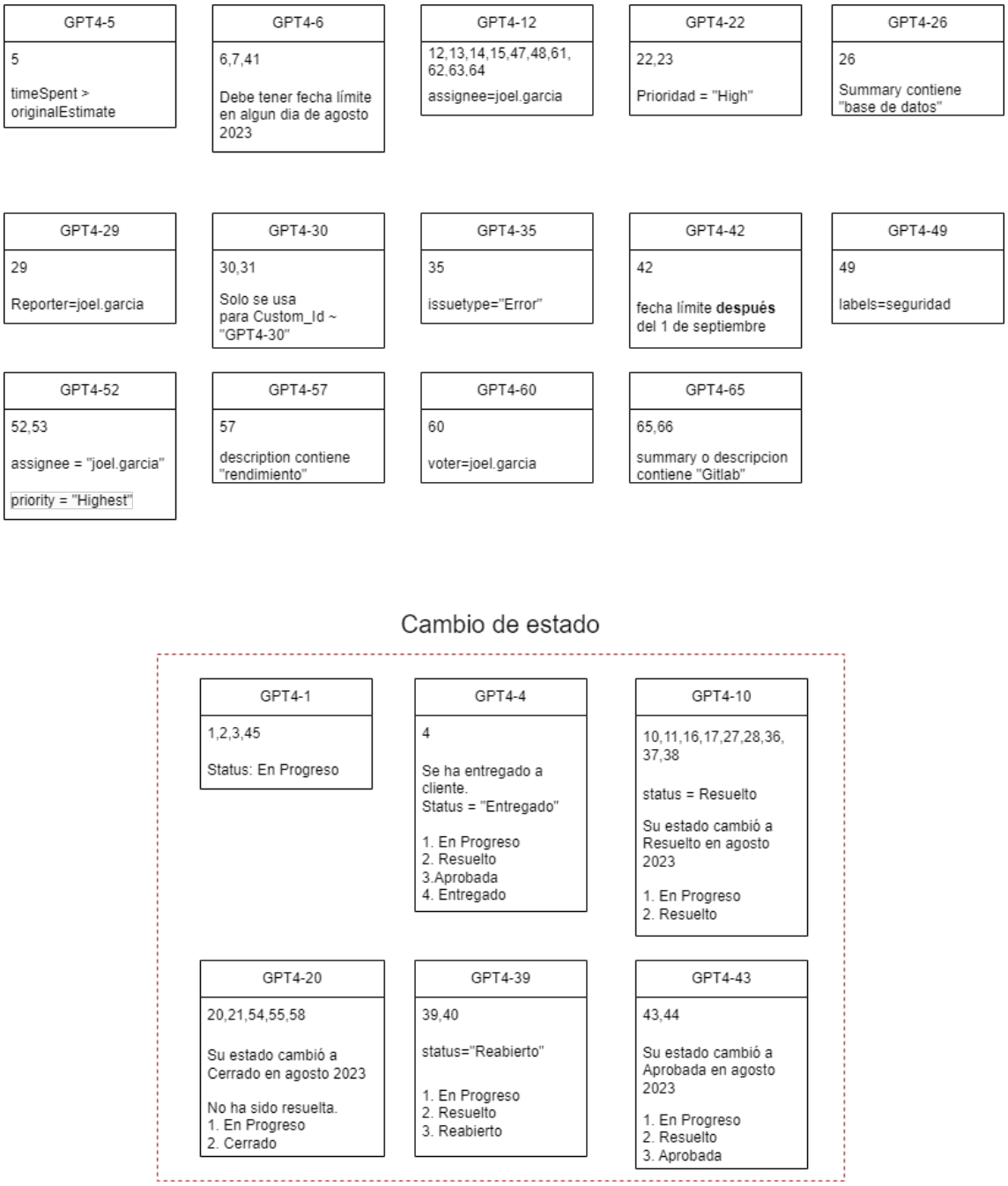

Figura 12: Conjunto de incidencias creadas en Jira. En la parte superior las incidencias que no dependen de cambios de estado. En la parte inferior las incidencias que sí dependen de cambios de estado

6.4. Realización de las pruebas

Se han realizado 4 pruebas en castellano e inglés. Cada prueba utiliza una plantilla diferente para evaluar el efecto que tiene cada plantilla en la tasa de acierto final. A la plantilla se le añade la pregunta y finalmente se obtiene una tasa de acierto de las 70 preguntas con la plantilla. Para la realización de las pruebas se ha utilizado la temperatura 0, ya que es la que mejores resultados ha dado cuando se ha probado JiraGPT Next de forma manual. Después de estas 4 pruebas, se explicará cómo se ha cogido la mejor plantilla y se han realizado pruebas variando la temperatura para ver cuál es la más ideal.

Para ejecutar las pruebas es necesario ejecutar el archivo benchmark.py mediante python. Se pueden modificar los parámetros necesarios para cambiar el tipo de prueba en el código de este archivo.

Es importante explicar los términos "zero-shot" y "few-shot" relacionados con las plantillas.

- Zero-shot:
  - Son las plantillas que no aportan ejemplos de pregunta-respuesta. El modelo tendrá que responder sin tener un ejemplo del formato de la respuesta que se desea. Se basará únicamente en la descripción en la plantilla.
  - Por ejemplo la plantilla
    - "Traduce la siguiente frase al inglés: {frase a traducir}"
    - La respuesta puede ser sólo la traducción, pero también puede contener una explicación adicional que no se ha pedido. En un programa que utilice LLMs como JiraGPT Next es necesario que el formato de la respuesta sea exacto y no se añadan explicaciones, ya que causaría errores.
- Few-shot:
  - Se refiere a las plantillas que sí proporcionan al menos un ejemplo de pregunta-respuesta. Aportan más información y le dan al modelo una idea más clara de lo que se espera de él, aumentando también la tasa de acierto de las respuestas.
  - Por ejemplo la plantilla
    - "Traduce la siguiente frase al inglés: {frase a traducir}. Hazlo en este formato: Gracias =›Thank you. Es miércoles =›It is wednesday. ".
    - La respuesta se espera que sea en el formato Español =›Inglés.
    - De esta forma se reducen las posibilidades de obtener una respuesta en un formato incorrecto.

A continuación se describen las 4 pruebas con su plantilla utilizada y la explicación de cada plantilla.

6.4.1. Plantilla de la Prueba 1: Zero-shot

6.4.1.1. Descripción

Se ha utilizado una plantilla de tipo zero-shot, ya que no proporciona ejemplos de pregunta-respuesta y simplemente se limita a pedir una sentencia JQL en la respuesta, sin

ninguna explicación adicional. Es una plantilla muy simple que funciona muy bien cuando la generación del JQL no requiere información específica sobre los proyectos de Jira de la empresa. Puede fallar en el formato deseado, pero JiraGPT Next corrige los errores de formato más comunes de la respuesta.

6.4.1.2. Plantilla

You are an AI assistant trained on JIRA Query Language. Your task is to translate user requests into precise JQL queries. Remember, the output should only be the JQL query itself, without any additional information or explanations

6.4.1.3. Razonamiento

El objetivo de esta prueba es determinar las capacidades del LLM, en este caso GPT-3.5, de traducir lenguaje humano a sentencias JQL sin contexto de las instancias de Jira de la empresa. La plantilla utilizada es zero-shot y por ello no se esperan los mejores resultados. Añadiendo más contexto es probable que mejore la tasa de acierto.

6.4.1.4. Resultados

La tasa de acierto de esta prueba fue del 17,14% . Aunque puede parecer un porcentaje bajo, hay que tener en cuenta que la plantilla utilizada contiene la mínima información sobre el formato en el que debe devolver la JQL el LLM. La plantilla apenas utiliza tokens y por ello es extremadamente barata de usar, por lo que se considera una buena tasa de acierto conseguida.

Estudiando las preguntas en las que falla la plantilla se ve que falla en todas las preguntas que involucran estados de las incidencias de Jira y también falla en las preguntas que no especifican el proyecto. Lo primero se debe a que tiende a generar sentencias JQL que filtran por estados con el nombre en inglés. Como LKS utiliza nombres de estados en castellano que tampoco son una traducción exacta de los estados por defecto en inglés, la JQL devuelve error, ya que no encuentra dicho tipo de estado. Los fallos cuando no se especifica el proyecto ocurren porque GPT-3.5 tiende a inventarse el nombre de un proyecto cuando el usuario no lo menciona. Probando con el modelo más avanzado GPT-4, este nunca provoca estos fallos ya que no se inventa el proyecto cuando no es mencionado. Cuando las pruebas se ejecuten con mejores modelos como GPT-4 la tasa de acierto de esta prueba mejorará notablemente a causa de ello.

6.4.2. Plantilla de Prueba 2: Few-shot con Estados en Castellano

6.4.2.1. Descripción

Es de tipo few-shot, ya que, aunque no da ejemplos de pregunta-respuesta, aporta información específica sobre los estados que se utilizan en los proyectos de Jira en la empresa.

6.4.2.2. Plantilla

El texto que se agrega en cada prueba se encuentra en **negrita**

You are an AI assistant trained on JIRA Query Language. Your task is to translate user requests into precise JQL queries. Remember, the output should only be the JQL query itself, without any additional information or explanations.

Always use the following issue status names in Spanish and never in English: "Open" should be "Abierto", "In Progress" should be "En Progreso", "Resolved" should be "Resuelto", "Approved" should be "Aprobada", "Delivered" should be "Entregado", "Reopened" should be "Reabierto", "Closed" should be "Cerrado".

6.4.2.3. Razonamiento

Se ha mejorado la plantilla para mejorar la tasa de acierto teniendo en cuenta que muchos fallos ocurren porque GPT-3.5 no sabe los estados que usa LKS en sus proyectos de Jira. Incluir información sobre los estados de esta forma hace mucho más probable que se genere la JQL correcta, ya que se informa al LLM de que debe reemplazar los estados en inglés por los estados en castellano. Se han incluido todos los estados que se utilizan en los proyectos en Jira de LKS.

6.4.2.4. Resultados

Se ha obtenido una tasa de acierto del 22,86 % frente al 17 % anterior. Es una ligera mejora frente a la plantilla anterior tras pasar de una plantilla zero-shot a una few-shot. Muchas de las preguntas falladas en la prueba anterior ahora se han resuelto correctamente. Sin embargo, siguen fallando preguntas que GPT-3.5 no ha sabido interpretar correctamente y todavía falla mucho cuando el usuario no especifica un proyecto específico. Por esto es evidente que se pueden realizar mejoras en la plantilla.

Hay que tener en cuenta que se ha logrado una mejora en la tasa de acierto usando una plantilla con más tokens, por lo que es más cara de utilizar. Se ha optimizado esta plantilla lo más posible para que tenga el menor número de tokens.

6.4.3. Plantilla de Prueba 3: Few-shot con especificación del nombre del proyecto

6.4.3.1. Descripción

Dado que la anterior plantilla sigue teniendo problemas en obtener respuestas correctas a las preguntas en las que no se especifica el proyecto, se ha intentado mejorar el prompt final añadiendo una nueva instrucción para que no se invente el nombre del proyecto además de un ejemplo de pregunta-respuesta.

6.4.3.2. Plantilla

You are an AI assistant trained on JIRA Query Language. Your task is to translate user requests into precise JQL queries. Remember, the output should only be the JQL query

itself, without any additional information or explanations.

Always use the following issue status names in Spanish and never in English: "Open" should be "Abierto", "In Progress" should be "En Progreso", "Resolved" should be "Resuelto", "Approved" should be "Aprobada", "Delivered" should be "Entregado", "Reopened" should be "Reabierto", "Closed" should be "Cerrado".

Please note: If a project name is not specified in the user's request, do not invent or assume a project name. Simply omit the project name from the generated JQL query. For example, given the user's query "Muestra las incidencias en progreso en GPT4", you should return "status = "En Progreso" AND project = GPT4"'

6.4.3.3. Razonamiento

Se añade una instrucción explícita para que el LLM no se invente el nombre del proyecto cuando el usuario no lo menciona, algo que suele hacer GPT-3.5. La instrucción está formada por dos peticiones, una negativa y otra positiva:

- Petición positiva:
  - *Simply omit the project name from the generated JQL query.*
  - Le dice al modelo qué debe hacer de una forma clara y proactiva. Elimina la ambigüedad.
- Petición negativa:
  - *If a project name is not specified in the user's request, do not invent or assume a project name.*
  - Dando una instrucción negativa clara se pretende evitar una respuesta no deseada, en este caso se pide que el LLM no asuma proyectos no mencionados por el usuario.

En todos los LLMs es recomendable utilizar una petición positiva y negativa en cada prompt para aclarar la respuesta que quiere el usuario y reducir la posibilidad de obtener interpretaciones erróneas en la respuesta. De esta forma se maximizan las posibilidades de obtener las mejores respuestas. Además, se refuerza con un ejemplo de pregunta-respuesta, con lo que el LLM se ajustará mejor al formato [28].

Esta buena práctica también se aplica a otros modelos de inteligencia artificial como Stable Diffusion, un modelo de generación de imágenes usando texto que recomienda utilizar peticiones negativas en los prompts para especificar elementos como el formato o el estilo de la imagen que no deben formar parte de la imagen generada [29].

6.4.3.4. Resultados

La tasa de acierto con esta nueva plantilla pasa de un 22 % a un 37,14 %. Por lo que se nota una gran mejora. Añadir un ejemplo de pregunta-respuesta suele dar buenos resultados con GPT-3.5 y otros LLMs. En este caso añadir un ejemplo ha supuesto una gran mejora frente a simplemente explicarle al modelo qué hacer.

Esta plantilla ha sido una de las más difíciles de lograr, ya que se han probado con una gran cantidad de plantillas que pedían al modelo no inventarse el nombre de un proyecto en la JQL cuando el usuario no menciona ninguno y ninguna de las plantillas funcionó. Algunas suponían una gran mejora de acierto en las preguntas en las que antes fallaba el prompt. Sin embargo y por desgracia, cuando el usuario sí que pedía las incidencias de un proyecto en específico, el modelo fallaba siempre y cometía el mismo error; añadía *status = "Incidencia"* a la sentencia JQL. Es extraño que con tantas plantillas diferentes haya dado ese mismo error, pero al fin y al cabo GPT-3.5 tiene muchos fallos y no es tan bueno como GPT-4. Lo que fallaba en todas las plantillas fallidas ha sido que no he probado a aportar un ejemplo de pregunta-respuesta que sea útil al modelo. El ejemplo proporcionado ha sido seleccionado porque es una pregunta en la que siempre fallaban las plantillas fallidas; cuando el usuario sí mencionaba el proyecto. De esta forma se ha obtenido un mejor resultado en las preguntas en las que el usuario menciona el proyecto y en las que no.

6.4.4. Plantilla de Prueba 4: Few-shot con 2 ejemplos

6.4.4.1. Descripción

Se ha realizado esta prueba mejorar aún más la anterior plantilla introduciendo un ejemplo más de pregunta-respuesta. La nueva plantilla pretende mejorar las respuestas que impliquen filtrar por prioridad de la incidencia y también pretende reforzar la probabilidad de que la respuesta sea devuelta en el formato JQL correcto al darle un ejemplo más de pregunta-respuesta.

6.4.4.2. Plantilla

You are an AI assistant trained on JIRA Query Language. Your task is to translate user requests into precise JQL queries. Remember, the output should only be the JQL query itself, without any additional information or explanations.

Always use the following issue status names in Spanish and never in English: "Open" should be "Abierto", "In Progress" should be "En Progreso", "Resolved" should be "Resuelto", "Approved" should be "Aprobada", "Delivered" should be "Entregado", "Reopened" should be "Reabierto", "Closed" should be "Cerrado".

Please note: If a project name is not specified in the user's request, do not invent or assume a project name. Simply omit the project name from the generated JQL query. For example, given the user's query "Muestra las incidencias en progreso en GPT4", you should return "status = "En Progreso" AND project = GPT4"'

Another example: given the user's query "¿Cuáles son las incidencias de máxima prioridad asignadas a joel.garcia?" you should return "assignee = "joel.garcia" AND priority = "Highest"".

6.4.4.3. Razonamiento

Disponiendo ya de un ejemplo de pregunta-respuesta, puede ser que la tasa de acierto de una plantilla con un ejemplo más aumente, ya que añadir ejemplos puede reforzar las posibilidades de obtener una respuesta correcta. En general es mejor añadir la

				Diferencia al anterior
Prueba	1	Correctas	12	-
		Incorrectas	58	
		Tasa de acierto (%)	**17,14%**	
	2	Correctas	16	+ 5,72%
		Incorrectas	54	
		Tasa de acierto (%)	**22,86%**	
	3	Correctas	26	+ 14,28%
		Incorrectas	44	
		Tasa de acierto (%)	**37,14%**	
	4	Correctas	34	+ 11,43%
		Incorrectas	36	
		Tasa de acierto (%)	**48.57%**	

Figura 13: Resultados de las 4 pruebas.

máxima información posible a todos los prompts, aunque en este TFG se ha tenido muy en cuenta el coste que puede suponer para la empresa que la aplicación sea utilizada por muchas personas, por lo que se intenta buscar un balance entre la cantidad de tokens en el prompt y la calidad del prompt. Además, como se muestra un ejemplo en el que el JQL debe filtrar por el tipo de prioridad de la incidencia, esta plantilla debería funcionar muy bien para las preguntas que filtren por prioridad.

6.4.4.4. Resultados

La aportación de este nuevo ejemplo ha resultado en una tasa de acierto del 48,57 %, bastante mayor que la anterior del 37 %. Han mejorado también las preguntas que filtran por prioridad gracias al ejemplo. Es evidente entonces que aportar un ejemplo más ha influido positivamente en los resultados finales.

Esta plantilla resultante es la definitiva. Se ha realizado un trabajo de experimentación exhaustivo en el que se han probado distintas formas de escribir la plantilla y esta es la que mejores resultados ha dado. Obviamente hay márgen de mejora; se podrían añadir más ejemplos, pero además de que eso podría llegar a tener un efecto negativo en la calidad de las respuestas, como TFG aplicado a la empresa, se ha buscado una relación coste/calidad y añadir más ejemplos haría más caro todo el programa cuando lo usen varias personas a la vez. En este caso, esta plantilla resulta ser una elección balanceada entre calidad y coste.

En la tabla 13 se puede ver un resumen de los resultados de las pruebas. Se detalla el número de respuestas correctas e incorrectas y la tasa de acierto de cada prueba. Además, se incluye como última columna la diferencia en la tasa de acierto que se ha ganado de una prueba a otra, de forma que se ve que el mayor salto en precisión ocurre en la prueba 3; cuando se añade la instrucción de no inventarse el nombre del proyecto.

6.4.5. Pruebas variando la temperatura

Una vez identificada la plantilla de prompt más efectiva en las pruebas y con el objetivo de encontrar el valor óptimo para el parámetro de temperatura del LLM, se han realizado 10 pruebas adicionales usando la plantilla de la prueba 4 y variando la temperatura usada en las pruebas.

El valor del parámetro temperatura en las 10 pruebas varía desde 0 hasta 1, incrementando su valor en pasos de 0.1.

Como se puede ver en la gráfica 14, hay un alto componente de aleatoriedad al variar el parámetro de temperatura. Se ve que con valores de temperatura menores a 0.5, los resultados son bastante estables. Cuando se utilizan valores de 0.5 o mayor, hay más inestabilidad y de media los resultados son peores que con temperaturas bajas.

Dada la naturaleza no determinista de los LLMs, en este caso del modelo GPT-3.5, es difícil sacar conclusiones acertadas sobre por qué se han conseguido resultados. Sin embargo, dado que las pruebas han sido sobre generar un código JQL, que es similar a las sentencias SQL, se trata de generación de código muy estructurado y ordenado que no deja lugar a interpretación subjetiva como si lo puede hacer el lenguaje natural.

Por esto, la conclusión sobre por qué se han obtenido mejores resultados con temperaturas menores a 0.5 es que en la generación de código muy estructurado y ordenado conviene utilizar parámetros del LLM que reduzcan la aleatoriedad e intenten obtener respuestas lo más deterministas posibles.

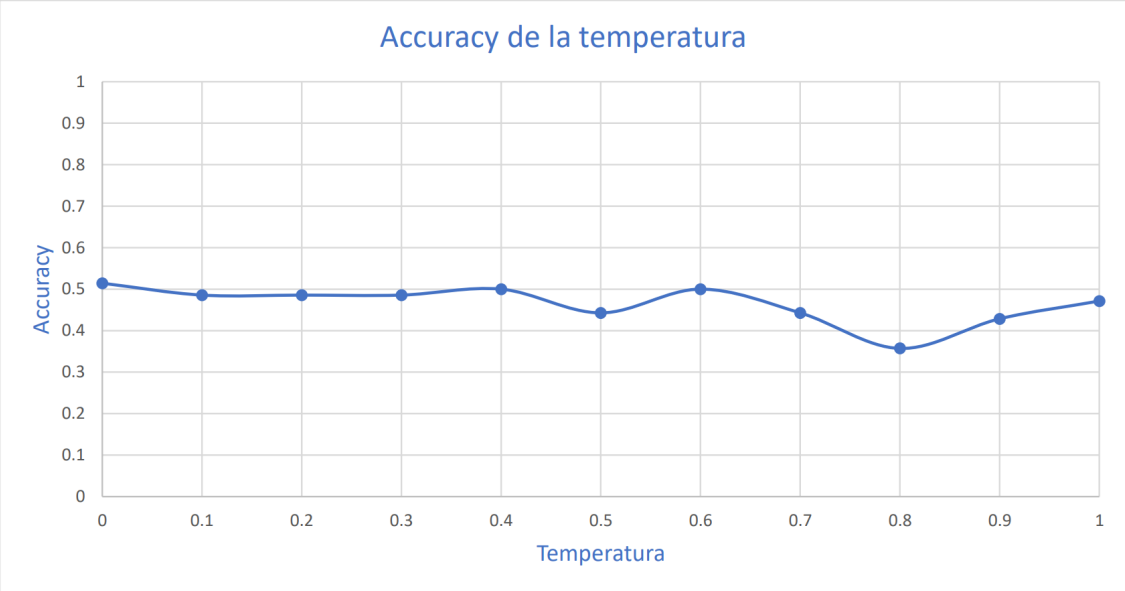

Figura 14: Gráfico con la accuracy obtenida según la temperatura

7. Análisis de alternativas

A continuación se detallan las 3 herramientas más relevantes que existen actualmente; Kubiya.ai en Slack, Copilot en Teams y Albus. Se explicarán los puntos positivos y negativos de cada una así como las razones por las que no se ha utilizado ninguna de las tres.

7.1. Kubiya.ai

Es un asistente virtual que implementa IA para ofrecer una interfaz como ChatGPT y permite interactuar con aplicaciones DevOps como Github o JIRA con lenguaje natural [30]. Está alojado en Slack y utiliza Actions y Workflows para controlar las integraciones con las aplicaciones como JIRA.

- **Actions**:
  - Son llamadas a una API de por ejemplo Github para obtener un dato así como para modificarlo, crearlo o borrarlo. Constan de un bloque de código en Python que hace una llamada con los datos necesarios como parámetros.
  - Se pueden crear y modificar manualmente para añadir nuevas llamadas a APIs.
- **Workflows**:
  - Son flujos de Actions a las que Kubiya.ai decide llamar según lo que haya escrito el usuario como prompt. Los Workflows se componen de Actions.
  - Al igual que las Actions, se pueden crear y modificar manualmente para tener Workflows personalizados.

Es un producto orientado para empresas que utilicen DevOps como LKS Next, por lo que era interesante contactar con ellos para que explicaran más a detalle la aplicación así como sus posibilidades con las herramientas DevOps que se utilizan en la empresa (JIRA, GitLab, Nexus...).

Se tuvo una reunión el 6 de junio con Shani Shoham, representante de Kubiya.ai. En la reunión resolvió nuestras dudas y realizó una demostración en directo de la versión actual de la aplicación y de la versión en Beta.

La versión actual simplemente utiliza un modelo de NLP (Natural Language Processing) que ha sido entrenado con lenguaje humano para entender el prompt del usuario y después ejecutar un Workflow. Le pregunta al usuario por los datos que sean necesarios

para ejecutarlo. Por ejemplo, en el caso de Jira, si el usuario pide las incidencias de una persona, el Workflow puede pedir el proyecto que debe analizar. Se mostrarían botones con los proyectos de la instancia de Jira y el usuario tendría que hacer click sobre uno de ellos.

- No utiliza un LLM y habría que implementar los Workflows y Actions para que se ajusten a nuestras herramientas, ya que hay sólo 10 Actions sobre JIRA y ningún Workflow actualmente.

La versión en Beta de Kubiya.ai implementa un LLM, por lo que se mejora el entendimiento del lenguaje y además permite la creación de Workflows mediante lenguaje natural. Ver Figura 15.

- La creación de Workflows mediante lenguaje natural crea un Workflow con todas las acciones necesarias. Este proceso utiliza técnicas avanzadas de Prompt Engineering para poder dividir los pasos a realizar y asignarles Actions. En la demostración, Shani Shoham intentó crear un Workflow con acciones, pero falló. Esto se debe a que es una funcionalidad todavía en progreso y todavía no está publicada en producción, por lo que no es de utilidad aún.

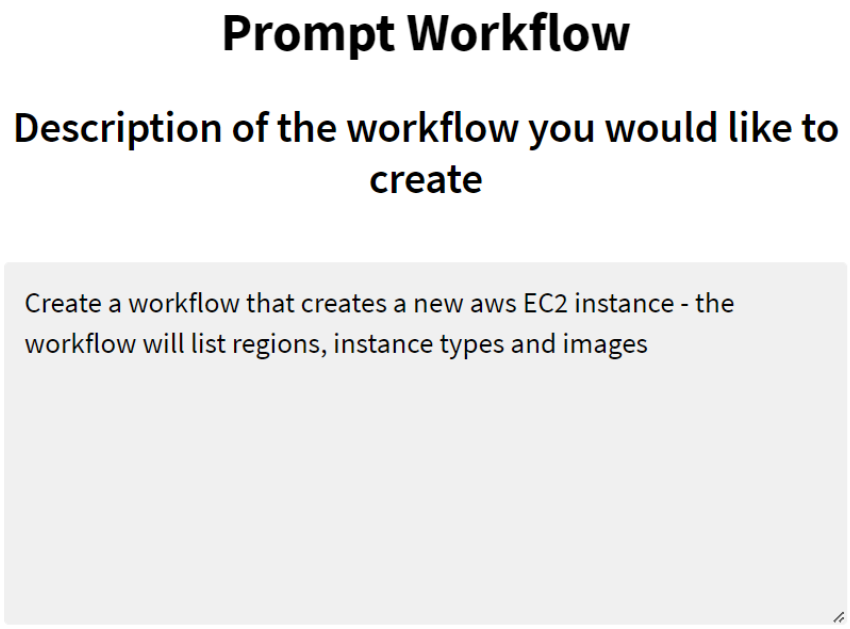

Figura 15: Imagen de creación de un Workflow en Kubiya.ai mediante lenguaje natural.

7.1.1. Seguridad de los datos

Para asegurar la gobernanza de los datos, Kubiya.ai dispone de un componente On-Premise[1] que se aloja en los servidores de las empresas contratantes. De esta forma, los datos del cliente no salen de sus instalaciones ni se alojan en una nube externa [31]. Este componente ejecuta las Actions en el entorno de la empresa para poder acceder a las herramientas internas.

[1] Se aloja en los servidores locales de la organización y no en una nube externa para garantizar la privacidad de los datos

Para el procesamiento del lenguaje, también dispone de un componente, esta vez en la nube de Kubiya.ai, que utiliza su LLM para procesas los prompts de los usuarios y generar las instrucciones necesarias. Para que funcione son necesarios ambos componentes.

7.1.2. Puntos positivos

- Implementar una solución como Kubiya.ai en la empresa puede agilizar procesos y mejorar la eficiencia de los empleados. Por ejemplo, si funcionase correctamente con JIRA, podría permitir a un Project Manager de Jira realizar consultas complejas sobre la instancia de Jira y así aumentar su productividad realizando un seguimiento del proyecto y sus tareas de una manera mucho más ágil.
- La versión Beta está programada para el último trimestre de 2023 e integraría las últimas funcionalidades en una versión mucho más estable. Entre otros, dispondría de más Actions y Workflows que podrían ser útiles para mejorar procesos en LKS Next, así como de la generación de Workflows con lenguaje natural.
- Utilizar Kubiya.ai es más rápido y sencillo que crear una aplicación web como he hecho yo. La interfaz es intuitiva y sencilla.
- Dadas las funcionalidades actuales de Kubiya.ai en Jira, tiene más posibilidades y complejidad que mi aplicación web. Mi aplicación web sólo permite leer incidencias, mientras que Kubiya.ai puede ejecutar acciones sobre incidencias además de crear nuevas incidencias.

7.1.3. Puntos negativos

- La integración con las herramientas DevOps que utiliza LKS Next tiene margen de mejora, ya que Kubiya.ai está muy orientado a utilizar Kubernetes y Jenkins entre otros. LKS Next utiliza Jira, GitLab, Nexus y Sonarqube en DevOps, y Kubiya.ai sólo dispone de 10 actions para JIRA, no tiene integración con ninguna de las demás herramientas.
    - Además, Kubiya.ai se trata de un bot de Slack y no tiene compatibilidad con Teams, que es la aplicación que utiliza LKS Next para las comunicaciones corporativas.
    - Utilizar Slack para Kubiya.ai a la vez que Teams podría repercutir en pérdidas de información y desde la empresa no se contempla migrar de las aplicaciones de Microsoft, con quien tienen partenariado.
- Tiene un coste muy elevado; 40$ cada mes por cada usuario. La aplicación JiraGPT Next desarrollada en este TFG supone un coste de uso muchísimo menor por cada usuario y ofrece una mejora en productividad notable. Además, para lo que puede ofrecer Kubiya.ai, no merece la pena contratar sus servicios.

Dado que la única integración con las herramientas que utiliza LKS Next es con la aplicación de gestión de proyectos Jira, a continuación se detallan las tareas que es capaz de realizar Kubiya.ai sobre Jira según la documentación oficial de Kubiya.ai [32].

7.1.3.1. Funcionalidades de Jira

- Asignar una incidencia a una persona.
- Dar por terminada una incidencia.
- Comprobar el estado de una incidencia.
- Crear una nueva incidencia.
- Dar acceso a una persona a un proyecto.

7.1.4. Conclusiones

Es una herramienta todavía por pulir pero que en un futuro puede ser ideal para las empresas como LKS Next, las cuales buscan una mejora continua en los procesos DevOps. Para este trabajo de fin de grado he decidido no utilizarla, ya que mi aplicación web obtiene resultados muy similares con Jira y no requiere contratar un servicio externo tan caro.

Como punto adicional, en la reunión con el representante de Kubiya.ai nos explicó el flujo de información que seguía la aplicación Beta (la que utiliza un LLM) para realizar las peticiones a las APIs. El flujo de información es el mismo que utilizo yo en mi aplicación, lo que valida mi enfoque, ya que otros desarrolladores lo han planteado de la misma forma.

7.2. Microsoft 365 Copilot

Microsoft 365 Copilot (no confundir con Github Copilot) es una herramienta de asistencia con IA que se integra con Teams. Su objetivo es proporcionar asistencia basada en IA en las aplicaciones incluidas en la suite de aplicaciones Microsoft 365.

Se ha incluido como herramienta DevOps dado que tiene la capacidad de integrarse con Jira para realizar consultas rápidas sobre todos los proyectos en tiempo real. Además, en un futuro seguramente se integre con Azure DevOps para agilizar mucho más la gestión de un entorno DevOps. Aunque Copilot no es una herramienta diseñada específicamente para el ámbito DevOps, sus capacidades de mejora de los procesos de desarrollo hacen que sea una oportunidad para los equipos DevOps.

A continuación se muestra una demostración de Copilot utilizando el plugin que lo integra con Jira. La demostración es de la conferencia Microsoft Build de 2023 [33].

7.2.1. Demo con plugin de Jira

- En la imagen 16 el usuario le realiza una pregunta a Copilot que necesita acceso a internet y Copilot responde citando las fuentes que ha utilizado para generar la respuesta.
- Después, tal y como se ve en la imagen 17, el usuario le pide a Copilot que cree un ticket en Jira utilizando la información que acaba de obtener. Copilot entonces decide utilizar el plugin de Jira para crear el ticket que le ha pedido el usuario y muestra al usuario que se ha creado el ticket correctamente.

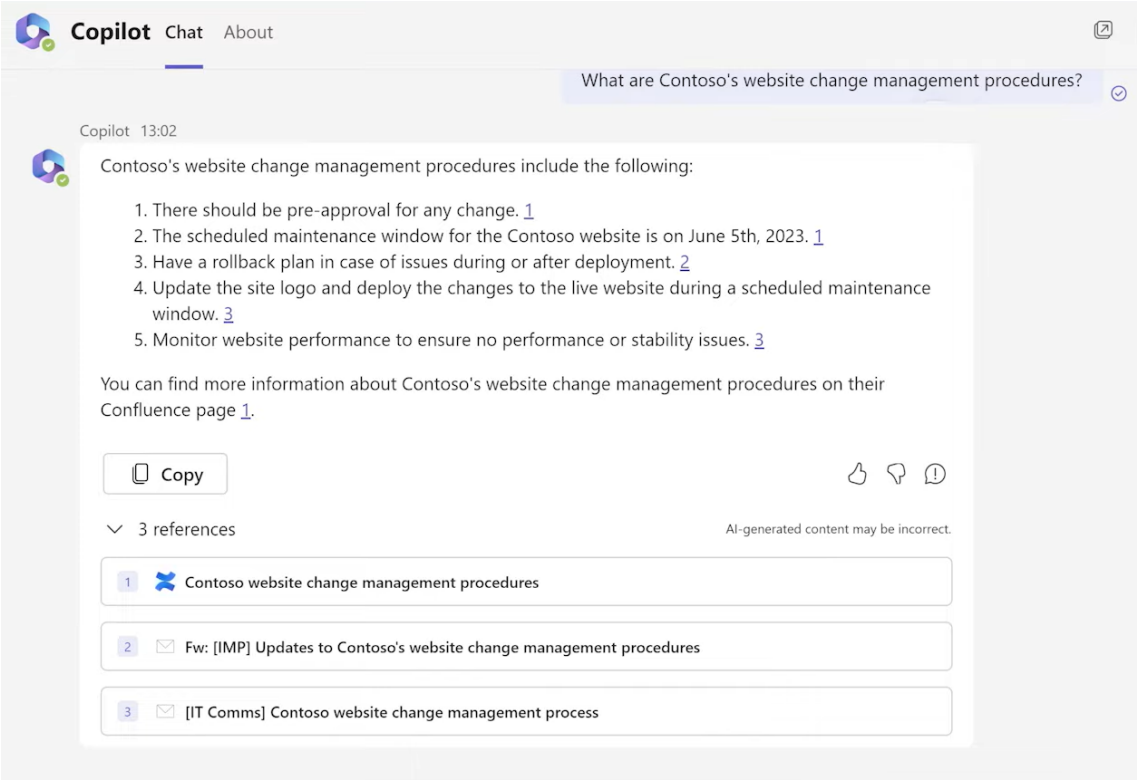

Figura 16: Pregunta a Copilot que necesita acceso a internet.

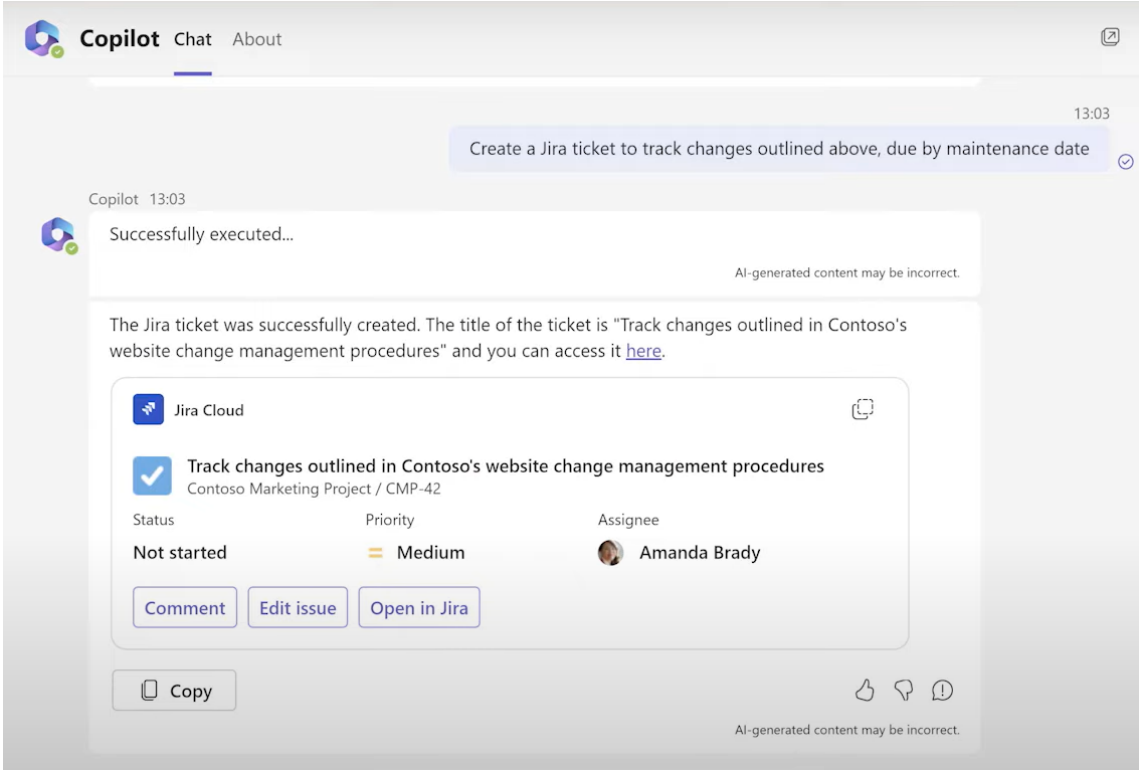

Figura 17: Creación de ticket en Jira mediante M365 Copilot.

- El plugin de Copilot para Jira utiliza su API para acceder a la información de la instancia de Jira y realizar cambios. La aplicación desarrollada en este TFG no incluye la creación de tickets porque iba a conllevar muchos más riesgos. Puede que el

programa crease tickets, pero habrían muchos campos que el modelo GPT-3.5 se inventaría y acabaría creando tickets erróneos que generarían desorden en la instancia de Jira de LKS Next.

7.2.2. Seguridad de los datos

Copilot solo permite que los usuarios obtengan información de fuentes de datos a las que tengan acceso [34], por lo que la confidencialidad de los datos dentro de la empresa es la misma que había antes con todos los servicios de Microsoft 365.

7.2.3. Puntos positivos

- Facilita una interfaz de lenguaje natural
  - Permite a los usuarios de la empresa interaccionar con una interfaz de lenguaje natural para consultar datos internos a los que tienen acceso. El potencial de la herramienta es poder realizar consultas muy complejas para generar información u obtener respuestas resumidas teniendo solo que escribir una consulta en lenguaje natural como si de ChatGPT se tratase.
  - Por ejemplo, es capaz de realizar consultas sobre proyectos e incidencias de Jira y más funcionalidades que superan a las de JiraGPT Next, como la creación de tickets utilizando lenguaje natural.
  - También es importante que es conversacional, por lo que tiene en cuenta todo el registro de conversación. JiraGPT Next no es conversacional porque la API de OpenAI actualmente necesita que se le mande la conversación entera en cada llamada y eso implica que sea inevitable llegar al límite de tokens en poco tiempo y además un coste muy elevado.

    Copilot es conversacional porque ya tendrá funcionalidades de la API de OpenAI que todavía no hayan sido lanzadas al público, como no tener que mandar toda la conversación en cada pregunta a la API de un modelo GPT.
- Copilot se actualizará con más plugins en el futuro
  - Dado que es un producto de Microsoft muy ambicioso dirigido a todo tipo de empresas que utilicen Microsoft 365, se espera que se desarrollen plugins para todo tipo de aplicaciones.
  - Por ejemplo, cabe esperar un plugin para gestionar repositorios de Github mediante lenguaje natural, ya que pertenece a Microsoft.
  - También, aunque no sean del ecosistema de aplicaciones de Microsoft, multitud de aplicaciones del ámbito DevOps podrían tener su propio plugin de Copilot para aumentar el número de usuarios. En LKS Next se utiliza Docker y Portainer para desplegar aplicaciones, Gitlab como respositorio de código, Nexus para las dependencias y SonarQube para la calidad del código.

    Viendo el interés creciente que tienen muchas empresas en sacar sus propios asistentes virtuales, es razonable pensar que muchas herramientas que usa LKS Next se lleguen a integrar en Copilot mediante plugins.

7.2.4. Puntos negativos

- El mayor punto negativo de implementar Copilot en LKS Next es su coste. Con un precio de 30€ al mes por usuario, la inversión sería muy costosa si se utilizase para todos los usuarios de la organización. Claro que se podría optar por adquirir la licencia de Copilot sólo para los puestos más importantes de la organización y no comprar una licencia para todas las personas, esta sería una buena opción.

  En el contexto de una empresa grande como LKS Next, habría que evaluar si el retorno de la inversión (ROI)[2] sería aceptable.

- Utilizar Copilot en LKS Next podría provocar que los empleados desarrollasen una gran dependencia de la herramienta. Los usuarios confiarían tanto en Copilot para realizar sus tareas que llegaría el punto en el que les resultaría muy difícil operar sin ella. Es posible que los empleados olviden cómo realizar algunas tareas sin utilizar Copilot y esto podría tener consecuencias muy negativas si el servicio de Copilot se cae o si la empresa busca una alternativa más económica.

  Además, viendo el potencial que puede llegar a tener Copilot, si en base a las métricas que consiga recabar Microsoft ven que pueden arriesgarse a subir el coste por empleado, seguramente lo harán. En caso de que un aumento del coste de Copilot fuese demasiado alto, LKS Next tendría que prescindir del servicio y si los empleados son demasiado dependientes de él esto puede ocasionar pérdidas en la productividad.

- Como Copilot utiliza LLMs, no está exento de errores o alucinaciones del LLM. Como los LLMs actuales en los que se basa Copilot tienden a inventarse información a veces, recibir respuestas erróneas de Copilot puede llevar a confusiones que lleguen a ocasionar pérdidas en la organización.

  Además, este potencial de error constata la necesidad de que los empleados estén adecuadamente formados en el uso de Copilot y también en sus capacidades y limitaciones. No se puede esperar que Copilot siempre dé respuestas correctas y es crucial que los usuarios puedan interpretar y verificar las respuestas del asistente. Esto implica una inversión adicional en formación para garantizar que los empleados sepan cuándo confiar en el asistente y cuándo es necesario verificar la información.

7.2.5. Conclusiones

A diferencia del programa desarrollado en este TFG JiraGPT Next, Copilot es mucho más potente ya que utiliza una combinación de LLMs, incluído GPT-4 [34]. Además, como Copilot es un asistente virtual base al que se le podrán añadir multitud de plugins para aplicaciones que usan las empresas, es un asistente con un enorme potencial de crecimiento y mejora.

La implementación del asistente Microsoft 365 Copilot debe ser una decisión bien meditada, ya que aunque puede tener un gran impacto positivo en la productividad de los empleados, también existe el riesgo de que los empleados desarrollen una excesiva dependencia del asistente. Su integración en el entorno de trabajo de LKS Next podría simplificar muchos procesos y ahorrar tiempo, pero también es importante que los empleados estén formados en las nuevas herramientas basadas en IAs Generativas como LLMs para saber sus capacidades y sus limitaciones.

[2] Métrica financiera utilizada para medir la probabilidad de obtener el rendimiento financiero de una inversión.

7.3. Albus

Es una plataforma de inteligencia artificial que permite integrar varios servicios como JIRA, Drive y Dropbox junto con un asistente virtual alojado en Slack. Propone eliminar la necesidad de buscar información de forma manual en documentos internos de la empresa mediante un asistente que, dada una pregunta en lenguaje natural, utiliza distintas fuentes de datos de la compañía para responder al usuario con el mismo lenguaje. Se muestra como una herramienta útil para empresas que quieren mejorar su eficiencia y productividad.

Para estar informados sobre las posibilidades que ofrece Albus para la empresa, se tuvo una reunión con el representante de Albus Kaneesha Parikh. En esta reunión Kaneesha demostró las capacidades actuales de Albus y sus características. A continuación se explican los aspectos positivos y negativos de Albus y más información relevante para la empresa.

7.3.1. Seguridad de los datos

Hay dos roles en Albus; administradores y usuarios. Según el soporte de Albus[35], cuando un administrador añade una fuente de datos, esta también es visible para el resto de administradores. La fuente de datos solo es visible para los usuarios cuando estos lanzan una pregunta a Albus y tiene que referenciar a dichas fuentes. De esta forma, solo los administradores saben todas las fuentes de datos de las que dispone Albus y se evita dar esa información a los usuarios a no ser que la necesiten.

Dada una pregunta de un usuario, Albus consulta con una API externa, por ejemplo la API de Jira, y procesa los datos con la API de OpenAI. En todas las fases de este proceso en tránsito y en reposo, los datos del cliente son cifrados. Además, se menciona que el contenido de las fuentes de datos nunca se comparte con otros clientes de Albus. Para corroborar esto, Albus tiene el certificado para la seguridad de la información ISO 27001.

Todo el acceso se controla y queda registrado; utilizan inteligencia artificial para detectar cualquier actividad sospechosa de los usuarios.

7.3.2. Puntos positivos

- Puede aumentar la eficiencia y la productividad de la empresa al proporcionar respuestas rápidas y precisas a las preguntas de los empleados.
- Tiene integración con Jira y es muy sencilla, aunque no parece más útil que mi programa de Jira. En la figura 18 el usuario pide el estado de sus tickets de Jira. Albus responde con una presentación de las incidencias que deja mucho que desear; no ofrece ningún link para poder acceder a las incidencias y tiene mucho margen de mejora en cuanto a visualización.
- Destaca al resolver preguntas de los empleados que de otra forma irían al equipo de soporte de la empresa. Si por ejemplo un empleado tiene dudas sobre los días de vacaciones que le corresponden según su situación específica, Albus devuelve una respuesta en lenguaje natural muy precisa y además referencia los documentos o las fuentes de datos como APIs de donde ha sacado la información. Cuando referencia un documento especifica el número de página concreto.

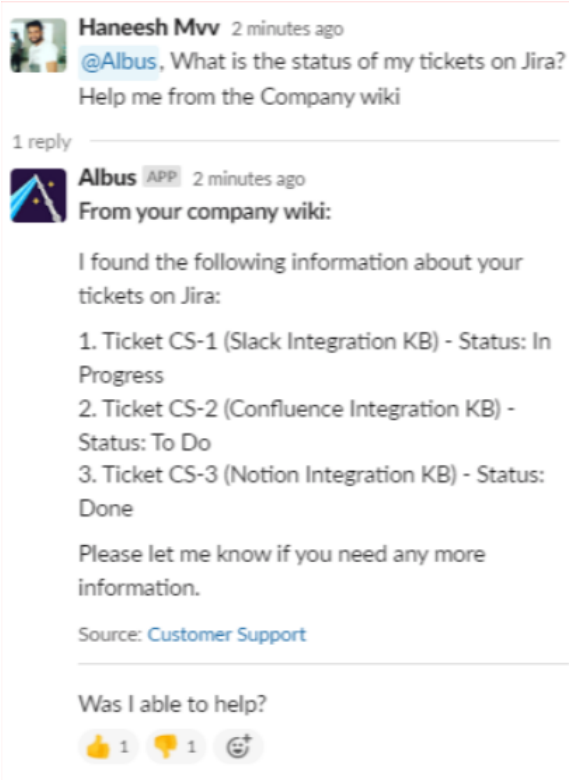

Figura 18: Ejemplo de consulta a Jira utilizando Albus.

- Proporcionar las fuentes de datos correspondientes junto con la respuesta ayuda a verificar que la información sea la correcta, ya que si el usuario tiene dudas, puede consultar la fuente o el documento para asegurarse de ello. Esto es algo que por ejemplo Kubiya.ai no tiene implementado, ya que es más complejo de realizar.
- Es muy sencillo añadir fuentes de datos como carpetas de Drive o documentos sueltos para luego poder responder teniéndolos en cuenta.
- Utiliza el LLM GPT-4, el cual es el más potente en la actualidad. Está entrenado con muchísimas APIs, es capaz de entender muchos idiomas, entre los cuales está el español, y es capaz de procesar preguntas muy complejas.
- Albus afirma que puede aprender de las consultas de los usuarios para mejorar las respuestas. Actualmente la API de GPT-4 no permite entrenar al modelo con nuevos datos. Para que sea correcto, deberían de disponer de algún otro modelo de inteligencia artificial que tome parte en las consultas además de GPT-4. Este modelo sí se podría alimentar con las consultas de los usuarios y podría preprocesar o postprocesar la consulta que se le envía a GPT-4.

7.3.3. Puntos negativos

- El precio es muy caro para lo que ofrece. Cada usuario supone un coste de 10 euros al mes y dispone de hasta 100 preguntas al mes. Si un usuario agota estas 100 preguntas, puede recargar su cuenta con otras 100 preguntas adicionales por un coste de 8 euros.
- Aunque GPT-4 sea el mejor LLM actualmente, no está exento de errores. Un error en la respuesta de Albus como puede ser una confusión o una invención de un dato, puede provocar que un trabajador tome una decisión incorrecta que acabe conllevando costos adicionales y pérdidas financieras para la empresa en el peor caso.
- Utilizar Albus en LKS Next implicaría utilizar un servicio externo para consultar información sobre datos internos de la empresa. Si Albus tuviese una brecha de seguridad, un agente malicioso podría modificar su sistema de transferencia de los datos de los clientes y los datos de LKS Next podrían quedar expuestos. Depender de terceros para tratar los datos de la empresa supone un riesgo que puede no ser necesario.

7.3.4. Conclusiones

Albus es una herramienta para la productividad que en un futuro podría ser buena opción implantarla en LKS Next. Sin embargo, las pocas integraciones que tiene con las herramientas DevOps de la empresa como Gitlab, Sonatype Nexus o Sharepoint junto con el muy elevado precio de 10 euros al mes por cada usuario la hacen demasiado cara para lo que puede ofrecer a la empresa.

8. Conclusiones

Este trabajo se enfoca en tres áreas principales: la aplicación JiraGPT Next con LLMs, el programa de pruebas para evaluar el rendimiento de diferentes LLMs en JiraGPT Next, y soluciones alternativas que emplean LLMs para mejorar la productividad. Se presentan conclusiones específicas para cada uno de estos aspectos. Después se analizan y comparan los costes previstos con los totales y finalmente se describen las posibles mejoras futuras para JiraGPT Next.

8.1. Conclusiones sobre JiraGPT Next

El desarrollo del programa JiraGPT Next ha sido útil para LKS Next pues es un desarrollo de un programa experimental que combina los LLMs con una aplicación muy importante utilizada en LKS Next como Jira. En el momento en el que se inició este trabajo de fin de grado apenas había desarrollos de aplicaciones que integrasen LLMs, ya que es una tecnología emergente muy difícil de integrar en aplicaciones en etapa de producción debido a lo impredecibles que son los LLMs.

JiraGPT Next se ha desarrollado utilizando el modelo GPT-3.5 de OpenAI. A pesar de sus capacidades, este modelo puede ser mejorado en infinidad de aspectos como su capacidad de tokens y su capacidad de entendimiento. JiraGPT Next no da resultados excelentes debido a los fallos que dependen totalmente del LLM GPT-3.5 Sin embargo, si en el futuro se integra JiraGPT Next con un LLM más potente como GPT-4 se obtendrán mejores resultados, ya que el LLM es el gran limitante del programa. Además, las plantillas diseñadas que se han utilizado en el programa sirven como base para futuros desarrollos con LLMs en LKS Next, ya que el desarrollo de aplicaciones con LLMs es muy prometedor para crear aplicaciones con interfaces de lenguaje natural.

Aumenta en gran medida la eficiencia de los empleados a la hora de consultar incidencias de Jira mediante lenguaje natural y de forma intuitiva, eliminando la necesidad de aprender y utilizar JQL. Minimizando el tiempo que tardan los empleados como los jefes de proyecto de Jira, se puede aprovechar ese tiempo en tareas que aporten valor a la empresa. Tener más tiempo para enfocarse en tareas estratégicas supone una gran mejora en el rendimiento de los jefes de proyecto.

En el caso de LKS Next, el jefe de proyecto no necesita utilizar JQL para acceder a las incidencias que le interesan. Además puede preguntarle al programa como si de una persona se tratase y formularle preguntas complejas que no solo utilizan JQL, sino un procesamiento del lenguaje del que solo son capaces los LLMs actuales como GPT-3.5. El tutor de la empresa de LKS Next Arkaitz Carbajo gestiona varios proyectos Jira y ha valorado positivamente JiraGPT Next, ya que le permite realizar consultas a Jira sin preocuparse por escribir sentencias JQL.

A continuación se desglosan los beneficios que suponen cada llamada al LLM GPT-3.5 que se realiza en el programa. La primera para interpretar las consultas de los usuarios, la segunda para identificar los campos relevantes de las incidencias, y la tercera para generar respuestas en lenguaje natural. Posteriormente, una vez explicado el impacto positivo de cada una de las llamadas, se detallarán los beneficios generales de la aplicación.

8.1.1. Beneficios de cada llamada a GPT-3.5

Inicialmente el programa solamente iba a servir para consultar incidencias mediante lenguaje natural usando GPT-3.5, lo cual ya es bastante novedoso y complejo. Apenas hay soluciones que permitan esto, y las que existen son muy caras además de que implican compartir datos internos de la empresa con terceras partes.

Para aumentar la complejidad del programa e implementar nuevas funcionalidades como responder a preguntas más complejas que no sólo piden consultar ciertas instancias, se tuvo la idea de añadir una nueva llamada a GPT-3.5 que tuviese en cuenta las incidencias recuperadas para poder hacer preguntas sobre ellas. Algunas preguntas complejas que podrían realizar los jefes de equipo de Jira son "¿De todas las incidencias, qué persona es la que tiene asignadas menos tareas?" o "¿Qué tareas tienen poco progreso?".

Añadir esta llamada mejora la experiencia general del usuario, al recibir una respuesta en lenguaje natural fácil de entender que se adapta a sus consultas. Los usuarios pueden así no gastar tiempo valioso creando informes de forma manual en Jira para obtener esta información.

La llamada a GPT-3.5 para resolver la pregunta compleja requería mandar todos los datos de las incidencias, los cuales gastan muchísimos tokens. Para no sobrepasar los límites de tokens de GPT-3.5, que actualmente son 4k tokens, y para ahorrar costes, se decidió implementar una nueva llamada a GPT-3.5 con el objetivo de minimizar al máximo los datos de las incidencias que se mandan a GPT-3.5.

Esta llamada consiste en describirle el programa de Jira a GPT-3.5 y pedirle que, dada la pregunta del usuario, devuelva los campos JSON que se deberían mandar a GPT-3.5 en la siguiente llamada para poder resolver la pregunta.

De esta forma se ha conseguido utilizar hasta un 90 % menos de tokens, ya que en la mayoría de casos sólo son necesarios 3 o 4 campos JSON de las incidencias en vez de todos los campos. Esto repercute en un ahorro de hasta 90 % en el uso de tokens de la API de GPT-3.5 en la llamada más cara. De esta forma se optimizan al máximo los fondos necesarios para utilizar el programa y se aumenta la probabilidad de obtener respuestas más precisas y contextualmente relevantes al no mandar tanta información.

JiraGPT Next supone una base para el desarrollo de futuras aplicaciones con LLMs por las fases del programa y las plantillas que se han diseñado y actualmente proporciona una mejora en la productividad. Por todo esto se considera que los objetivos del desarrollo para LKS Next están cumplidos.

8.2. Conclusiones sobre el programa de pruebas y la investigación

El programa de pruebas es muy útil para evaluar como van a funcionar los diferentes LLMs en el programa JiraGPT Next. Ofrece una métrica de precisión que evalúa de forma objetiva tras muchos intentos. Evaluar un modelo integrándolo directamente en JiraGPT Next sin antes probarlo en el programa de pruebas no va a ofrecer una visión real de la calidad del modelo para esta aplicación, ya que esta evaluación no sería nada consistente y no estaría registrada ni evaluada correctamente con métricas. El programa de pruebas funciona correctamente, permite modificar las plantillas y evaluar diferentes LLMs, por lo que sido un desarrollo bien logrado y con gran potencial que ha cumplido con los objetivos formulados.

La investigación registrada en este TFG sobre alternativas trata diferentes programas que integran LLMs para mejorar la productividad. Ha sido una investigación útil para LKS Next ya que ha ofrecido una amplia perspectiva sobre las soluciones más relevantes en el campo de las aplicaciones con LLMs, aportando ventajas y desventajas de cada una y desarrollando una conclusión para cada solución. Especialmente la parte sobre Copilot 365 ha sido muy útil para LKS Next, ya que LKS Next cuenta con la Suite completa de Microsoft y tiene facilidades para adquirir el Copilot 365 e integrarlo en sus sistemas. Proveer a LKS Next de información de actualidad sobre los LLMs cumple el objetivo inicial de esta investigación.

8.3. Costes económicos

Los costes finales de software, hardware, mano de obra e indirectos han sido los mismos que los planificados anteriormente. El único coste que ha diferido de la planificación ha sido el coste de la API de GPT-3.5 de OpenAI. A continuación se muestra la diferencia entre el coste resultante y el planificado y se explica el por qué.

Tal y como se ha explicado en capítulo de planificación 4.3, se ha dispuesto de un presupuesto de 100€ para poder utilizar la API de GPT-3.5 de OpenAI, ya que es de pago. Se empezó a utilizar el 15 de mayo. Se utilizó la API de GPT-3.5 con 4000 tokens de contexto, que tiene el precio de alrededor de 0.002€ por cada 1000 tokens recibidos y generados.

A continuación se detallan los costes por cada mes

- Del 15 al 31 de mayo: 0.02
- Del 1 al 30 de junio: 0.2
- Del 1 al 31 de julio: 0.12
- Del 1 al 31 de agosto: 0.76
- Del 1 al 30 de septiembre: 0.04

El mes de agosto fue cuando más presupuesto se gastó porque fue cuando se realizaron las 8 pruebas con 70 preguntas. Teniendo en cuenta el presupuesto dado por LKS Next de 100€ y las estimaciones iniciales hechas de alrededor de 20€, se han tenido

unos gastos irrisorios en comparación a los esperados. El gasto total suma 1.14€ en 7 meses frente a los 20€ planificados.

La implementación de la Fase 2 en JiraGPT Next para optimizar al máximo la cantidad de información que se envía a la API de GPT puede suponer hasta un 90 % de ahorro en todas las llamadas a la API. Si no se hubiera integrado esta fase en el programa, los costes habrían sido mucho más altos.

En la imagen 19 se puede observar que los días 17 y 18 de agosto ha sido cuando más se ha utilizado la API. Estos dos días ya se había desarrollado el programa en python para ejecutar las pruebas y también se habían creado las incidencias en Jira, por lo que se iniciaron las pruebas.

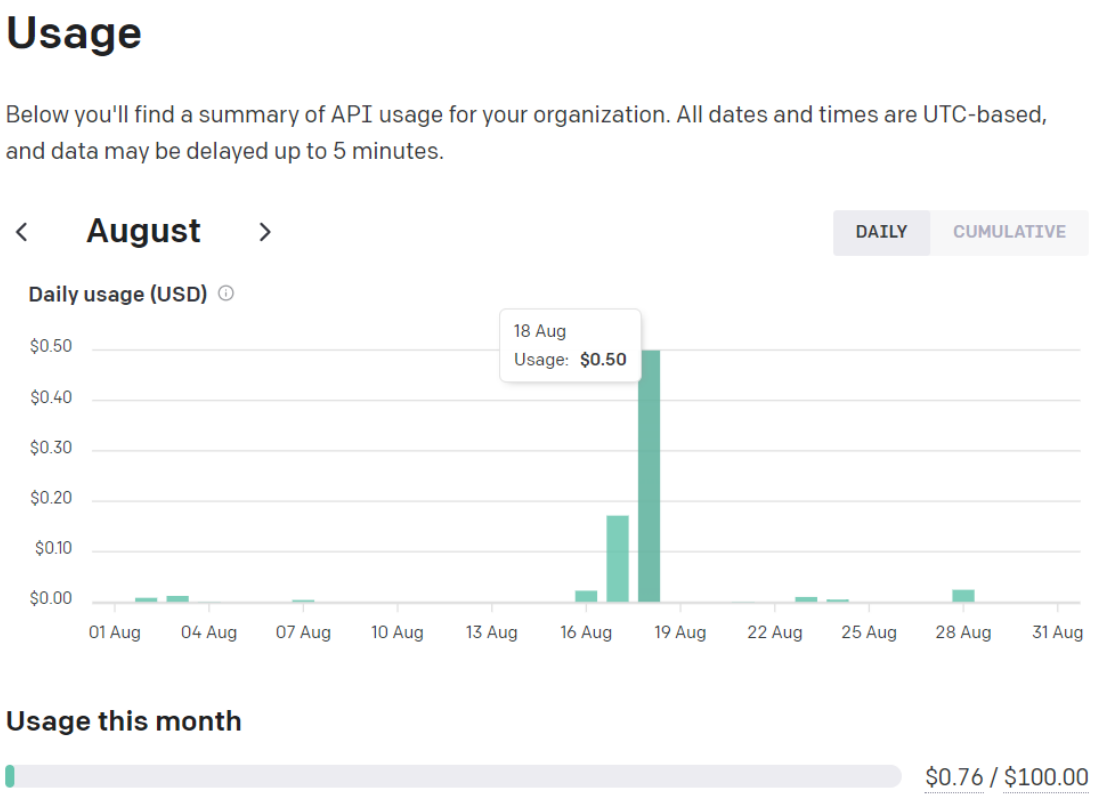

Figura 19: Captura de pantalla de los gastos en Agosto en la API de OpenAI

Los costes anteriormente mencionados se han obtenido mediante el registro de gastos de API que aporta OpenAI en su página.

8.4. Mejoras futuras

8.4.1. Fine-tuning de GPT-3.5

El 23 de agosto de 2023 OpenAI anunció una nueva funcionalidad para GPT-3.5: la capacidad de fine-tuning. En este TFG no se explorarán las capacidades del fine-tuning de GPT-3.5 puesto que cuando se habilitó, ya se había completado todo el desarrollo del programa de Jira así como del entorno de pruebas.

El fine-tuning es una técnica para ajustar LLMs ya entrenados. Frente a lo que se pueda

pensar de forma intuitiva, el servicio de fine-tuning no está diseñado para entrenar al modelo con nuevos datos para que el modelo responda después utilizando estos datos sino más bien para adaptar los LLMs a un formato de respuesta específico. El fine-tuning que ofrece OpenAI está orientado a entrenar a un LLM para que devuelva respuestas en un formato específico, no para entrenar al modelo con nuevos datos.

Para el programa de Jira inicialmente se intentó utilizar un modelo GPT-3 entrenado mediante fine-tuning de GPT-3, pero dado que GPT-3 fallaba muchísimo en la lógica para generar JQLs y que nunca conseguía generar las respuestas en formato adecuado, se decidió utilizar la API de GPT-3.5 con plantillas en cada prompt.

Si el fine-tuning de GPT-3.5 hubiese estado disponible al empezar el desarrollo del TFG, sin duda se habría utilizado, ya que tiene un gran potencial para mejorar la consistencia de las respuestas generadas. Las plantillas que se utilizan cada vez que el usuario realiza una petición no serían necesarias, ya que ya se habría entrenado a un modelo para cada fase para que diese una respuesta en un formato específico. Se habrían utilizado 3 modelos de GPT-3.5 con fine-tuning diferentes; uno para cada fase. De esta forma solo con mandar la petición del usuario en cada fase del programa de Jira, la respuesta estaría probablemente en el formato que le corresponde.

9. Agradecimientos

A Arkaitz Carbajo, mi tutor de las prácticas en LKS Next, por su gran ayuda y guía durante estos 7 meses de elaboración del TFG en su equipo. Su amabilidad y disposición han ayudado a sacar adelante este trabajo. También agradecer a mis compañeros de LKS Next como Alex Novoa, Iker y Oscar por resolver tantas dudas sobre cómo orientar el desarrollo de la aplicación e implementar distintas funcionalidades.

A Ander Barrena, por su valiosa contribución en la toma de decisiones sobre la implantación de LLMs en mi aplicación así como por su gran ayuda para diseñar el entorno de pruebas para lograr un mejor trabajo final. Su valoración sobre el alcance adecuado del TFG ha sido crucial.

10. Anexo

10.1. Diseño de la instancia Jira para el entorno de pruebas

A la hora de crear las incidencias en Jira, solo se deben cambiar el estado de las incidencias dentro del recuadro *Cambio de estado*. Hay que cambiar el estado en el orden exacto en el que se indica, ya que algunas preguntas tienen en cuenta las incidencias que han sido de un estado en algún momento. Mover el estado de cualquier incidencia puede hacer que esas preguntas dejen de funcionar, puesto que van a recibir una incidencia de más y eso dará error.

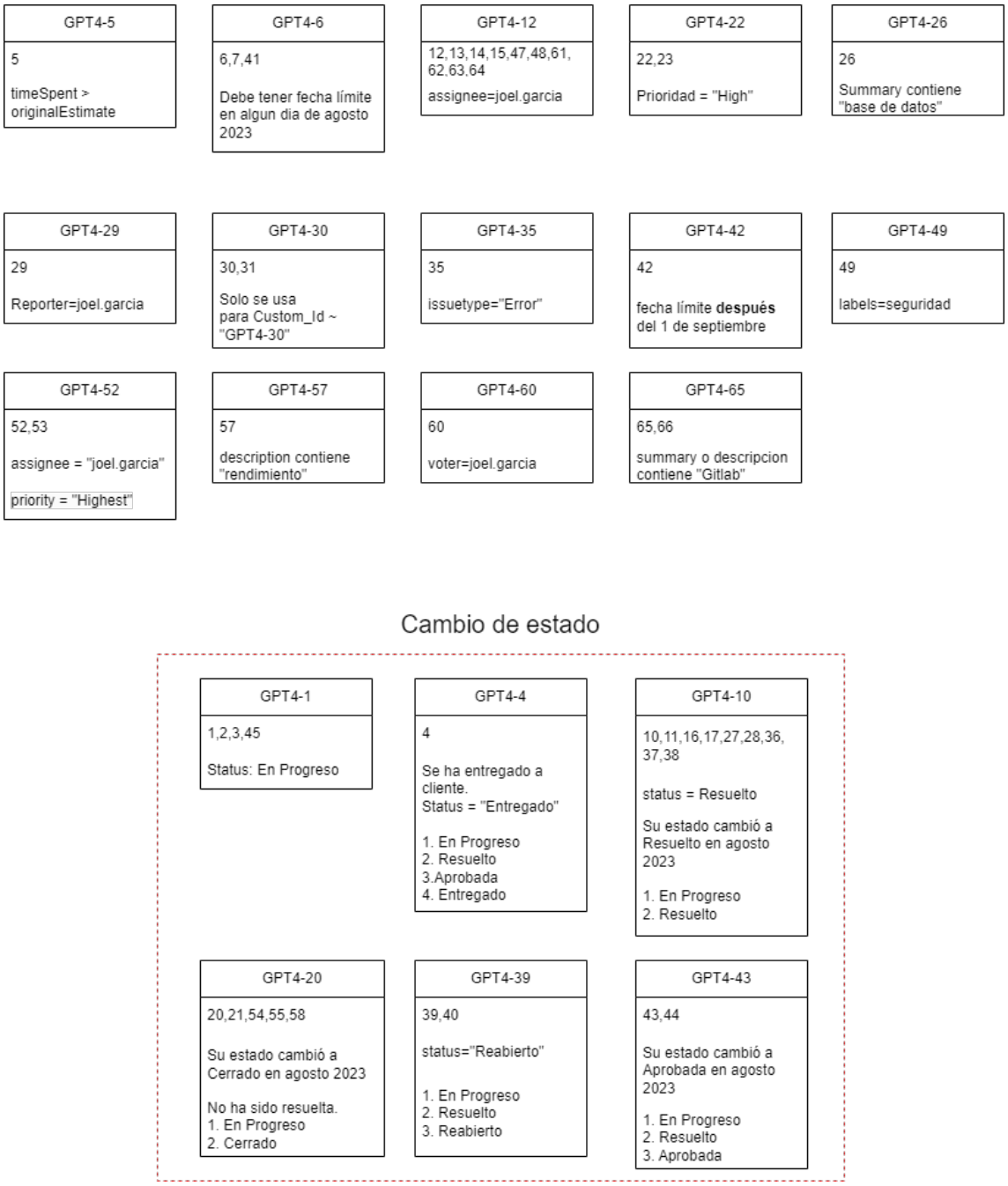

Figura 20: Diagrama del entorno de pruebas en Jira

10.2. 70 Preguntas del entorno de pruebas

Las 70 preguntas utilizadas en el entorno de pruebas en castellano e inglés y clasificadas en uno de los 3 posibles niveles según su dificultad de generación. En la sección 6.2 se explican los 3 niveles en los que se dividen las preguntas.

Tabla 29: Listado de preguntas utilizadas en el entorno de pruebas en castellano e inglés

ID	Pregunta (castellano)	Pregunta (inglés)	Nivel
1	Muestra las incidencias en progreso	Show me all issues in progress	1
2	Muestra las incidencias en progreso en GPT4	Show me all issues in progress in GPT4	2
3	Muestra las incidencias en progreso y sus horas acumuladas	Show me all issues in progress and their accumulated hours	1
4	Muestra las incidencias que se han entregado a cliente y sus horas acumuladas	Show me all issues sent to the client and their accumulated hours	1
5	Muestra las incidencias que se han pasado de horas y por cuánto	Show me all issues overdue and by how much	1
6	Muestra las incidencias que deberían acabar en agosto de 2023	Show me all issues that should be resolved by August 2023	1
7	Muestra las incidencias en el proyecto GPT4 que deberían acabar en agosto de 2023	Show me all issues from GPT4 that should be resolved by August 2023	2
8	Muestra quién tiene menos carga de trabajo	Show me who has less work hours	3
9	Muestra quién tiene menos carga de trabajo en GPT4	Show me who has less work hours in project GPT4	3
10	¿Cuántos tickets se han resuelto en agosto de 2023?	How many tickets have been resolved in August 2023?	3
11	¿Cuántos tickets se han resuelto en agosto de 2023 en GPT4?	How many tickets have been resolved in August 2023 in GPT4?	3
12	Muéstrame todas las incidencias abiertas asignadas a joel.garcia.	Show me all the open issues assigned to joel.garcia.	2
13	Muéstrame todas las incidencias abiertas asignadas a joel.garcia en GPT4.	Show me all the open issues assigned to joel.garcia in GPT4.	2

Continúa en la siguiente página

Tabla 29 – continuado desde la anterior página

ID	Pregunta (castellano)	Pregunta (inglés)	Nivel
14	Muéstrame todas las incidencias con estado "Abierto" asignadas a joel.garcia en GPT4.	Show me all the issues with status "Abierto" assigned to joel.garcia in GPT4.	2
15	Muéstrame todas las incidencias con estado "Abierto" asignadas a joel.garcia.	Show me all the issues with status "Abierto" assigned to joel.garcia.	2
16	Lista todas las incidencias resueltas en agosto de 2023	List all the issues resolved in August 2023	2
17	Lista todas las incidencias resueltas en agosto de 2023 en GPT4.	List all the issues resolved in August 2023 in GPT4.	2
18	¿Cuántos tickets sin resolver hay en el proyecto GPT4?	What's the number of unresolved tickets in project GPT4?	3
19	Encuentra las incidencias asignadas a joel.garcia en el proyecto GPT4.	Find issues assigned to joel.garcia in project GPT4.	2
20	¿Cuántas incidencias se cerraron en agosto de 2023?	How many issues were closed in August 2023?	3
21	¿Cuántas incidencias se cerraron en agosto de 2023 en el proyecto GPT4?	How many issues were closed in August 2023 in project GPT4?	3
22	Muestra todas las incidencias con alta prioridad.	Display all the issues with high priority.	1
23	Muestra todas las incidencias con prioridad "High".	Display all the issues with "High" priority.	1
24	Dame una lista de las incidencias actualizadas en agosto de 2023.	Give me a list of issues updated in August 2023.	1
25	Enumera de menor a mayor prioridad las incidencias actualizadas en agosto de 2023.	List from lowest to highest priority all issues updated in August 2023.	1
26	Encuentra todos los tickets con "base de datos" en el resumen.	Find all the tickets with "database" in the summary.	1
27	¿Cuántos tickets están resueltos?	How many tickets are resolved?	3
28	¿Cuántos tickets están resueltos en el proyecto GPT4?	How many tickets are resolved in project GPT4?	3

Continúa en la siguiente página

Tabla 29 – continuado desde la anterior página

ID	Pregunta (castellano)	Pregunta (inglés)	Nivel
29	Muestra todas las incidencias reportadas por joel.garcia.	Display all the issues reported by joel.garcia.	1
30	¿Cuál es el estado del ticket con el campo personalizado Custom_Id con valor GPT4-30? ¿A quien ha sido asignado?	What is the status of the ticket with the custom field Custom_Id having the value GPT4-30? Who has it been assigned to?	3
31	Muestra el progreso de la incidencia con el campo personalizado Custom_Id con valor GPT4-30.	Show me the progress of the issue with the custom field Custom_Id having the value GPT4-30.	1
32	¿Cuáles son las incidencias creadas en agosto de 2023?	What are the newly created issues in August 2023?	1
33	¿Cuáles son las incidencias creadas en agosto de 2023 en el proyecto GPT4?	What are the newly created issues in August 2023 in project GPT4?	2
34	¿Cuáles son las incidencias creadas en agosto de 2023 en el proyecto GPT4? Ordenalas de mayor a menor prioridad	What are the newly created issues in August 2023 in project GPT4? List them from higher to lower priority	2
35	¿Cuántos bugs se han reportado en el proyecto GPT4?	How many bugs have been reported in project GPT4?	3
36	Dame todas las incidencias resueltas	Give me all resolved issues	1
37	Dame todas las incidencias en la fase de “Resuelto”.	Give me all issues in the “Resuelto” phase.	1
38	Dame todas las incidencias en la fase de “Resuelto” del proyecto GPT4.	Give me all issues in the “Resuelto” phase from project GPT4.	2
39	Muéstrame todas las incidencias reabiertas.	Show me all reopened issues.	1
40	Muéstrame todas las incidencias con estado “Reabierto”.	Show me all issues with “Reabierto” status.	1
41	Muestra todos los tickets que vencen en agosto de 2023.	Display all the tickets due in August 2023.	1
42	¿Qué tickets terminan después del 1 de septiembre de 2023?	What tickets are due after September 1, 2023?	1

Continúa en la siguiente página

Tabla 29 – continuado desde la anterior página

ID	Pregunta (castellano)	Pregunta (inglés)	Nivel
43	Encuentra todas las incidencias movidas a "Aprobada" en agosto de 2023.	Find all the issues moved to "Aprobada" in August 2023.	2
44	Dame las incidencias aprobadas en agosto de 2023	Give me all approved issues in August 2023.	2
45	¿Cuántos tickets están en fase de "En Progreso"?	How many tickets are in "En Progreso" phase?	3
46	Lista todos los tickets creados en agosto de 2023.	List all tickets created in August 2023.	1
47	Muéstrame las incidencias abiertas más antiguas.	Show me the oldest open issues.	2
48	Muéstrame las incidencias con estado "Abierto" más antiguas.	Show me the oldest issues with "Abierto" status.	2
49	Lista todos los tickets con la etiqueta "seguridad" y dame un resumen.	List all tickets with "seguridad" label and give me a summary.	3
50	¿Cuántas incidencias se han movido a "En progreso" en agosto de 2023?	How many issues have been moved to "En progreso" in August 2023?	3
51	¿Cuántas incidencias se han movido a "En progreso" en agosto de 2023 en el proyecto GPT4?	How many issues have been moved to "En progreso" in August 2023 in project GPT4?	3
52	¿Cuáles son las incidencias de máxima prioridad asignadas a joel.garcia?	What are the top priority issues assigned to joel.garcia?	2
53	¿Cuáles son las incidencias de máxima prioridad asignadas a joel.garcia en el proyecto GPT4?	What are the top priority issues assigned to joel.garcia en el proyecto GPT4?	2
54	Lista todas las incidencias cerradas en agosto de 2023.	List all the issues closed in August 2023.	2
55	Lista todas las incidencias cerradas en agosto de 2023 en el proyecto GPT4.	List all the issues closed in August 2023 in GPT4.	2

Continúa en la siguiente página

Tabla 29 – continuado desde la anterior página

ID	Pregunta (castellano)	Pregunta (inglés)	Nivel
56	¿Cuántas incidencias nuevas se han agregado al proyecto GPT4 en agosto de 2023?	How many new issues have been added to project GPT4 in August 2023?	3
57	Encuentra todas las incidencias con "rendimiento" en su descripción.	Find all the issues with "rendimiento" in their description.	1
58	Encuentra todos los tickets movidos a "Cerrado" en agosto de 2023.	Find all the tickets moved to "Cerrado" in August 2023.	2
59	Muestra todas las incidencias que están vencidas.	Show all the issues that are past due.	1
60	Lista todos los tickets en los que joel.garcia ha votado.	List all the tickets joel.garcia has voted for.	1
61	¿Cuántas incidencias están en la fase "Abierto" en el proyecto GPT4?	How many issues are in the "Abierto" phase for project GPT4?	3
62	¿Cuántas incidencias están en la fase "Abierto"?	How many issues are in the "Abierto" phase?	3
63	¿Cuántas incidencias están abiertas?	How many issues are opened?	3
64	¿Cuántas incidencias están abiertas en el proyecto GPT4?	How many issues are opened in project GPT4?	3
65	Recupera las incidencias que tengan que ver con Gitlab	Show me issues that are related to Gitlab	2
66	Recupera las incidencias en el proyecto GPT4 que tengan que ver con Gitlab	Show me issues in GPT4 project that are related to Gitlab	2
67	Muestra las incidencias sin asignar	Show me the unassigned issues	1
68	Muestra las incidencias sin asignar en el proyecto GPT4	Show me the unassigned issues in GPT4 project	2
69	¿Cuantas incidencias sin asignar hay?	How many unassigned issues are there?	3
70	¿Cuantas incidencias sin asignar hay en el proyecto GPT4?	How many unassigned issues are there in project GPT4?	3

Bibliografía

[1] Forbes Staff. *Chatgpt Bate El Récord de crecimiento más rápido de usuarios en la historia*. Feb. de 2023. URL: https://www.forbes.com.mx/chatgpt-bate-el-record-de-crecimiento-mas-rapido-de-usuarios-en-la-historia/#:~:text=El%20informe%2C%20que%20cita%20datos,Foto%3A%20Archivo%20Reuters.

[2] Baba Tamim. *Google, Microsoft, Amazon, amp; Meta put AI on steroids while cutting jobs*. Abr. de 2023. URL: https://interestingengineering.com/innovation/google-microsoft-amazon-meta-ai#:~:text=The%20US%20tech%20giants%20like,the%20power%20of%20artificial.

[3] Diana Enríquez. *Como Usar Chatgpt Para Ser Más productivo en el trabajo*. Oct. de 2023. URL: https://www.iebschool.com/blog/como-usar-chatgpt-para-ser-mas-productivo-en-el-trabajo-tecnologia/.

[4] Jacob Johnson. *What is chatgpt amp; why should programmers care about it?* Jun. de 2023. URL: https://www.codecademy.com/resources/blog/what-is-chatgpt/.

[5] OpenAI. URL: https://openai.com/gpt-4.

[6] Maite Martamp;iacute;nez. *Elena Zárraga (LKS Next): "Lks next ampliará su presencia en madrid en legal Y consultoría en 2022"*. Dic. de 2021. URL: https://www.eleconomista.es/pais_vasco/noticias/11529318/12/21/Elena-Zarraga-LKS-Next-LKS-Next-ampliara-su-presencia-en-Madrid-en-legal-y-consultoria-en-2022.html.

[7] Venny Turner. *The art and evolution of prompt engineering: A comprehensive guide to mastering AI queries*. Sep. de 2023. URL: https://pub.aimind.so/the-art-and-evolution-of-prompt-engineering-a-comprehensive-guide-to-mastering-ai-queries-40d825b49814.

[8] Anastasia Stsepanets agosto 2, Anastasia Stsepanets y agosto 2. *Ventajas y desventajas, Características de Jira Software*. Ago. de 2023. URL: https://blog.ganttpro.com/es/caracteristicas-de-jira-software/#:~:text=Jira%20es%20una%20herramienta%20de,las%20caracter%C3%ADsticas%2C%20ventajas%20y%20d.

[9] Alexandre. *Gitlab: Saber Todo Sobre el Repositorio Git para devops*. Dic. de 2022. URL: https://datascientest.com/es/gitlab-todo-lo-que-hay-que-saber.

[10] Exceltic. *Devops en Menos de 3 minutos*. Nov. de 2017. URL: https://www.youtube.com/watch?v=p-bOnV8FRMQ.

[11] Wendy Cornell. *Modelos Matemáticos Para El Diseño de fármacos*. Oct. de 2020. URL: https://elpais.com/ciencia/2020-10-20/modelos-matematicos-para-el-diseno-de-farmacos.html.

[12] Ángel Canal-Alonso y col. *Ia Generativa en ensayos Clínicos: Revolucionando El Diseño, análisis y predicción en Investigación Médica*. Ene. de 2020. URL: https://gredos.usal.es/handle/10366/153109.

[13] Jul. de 2023. URL: https://es.futuroprossimo.it/2023/07/deepmind-presenta-rt-2-robot-che-vedono-apprendono-e-agiscono/.

[14] Ashish Vaswani y col. «Attention is All You Need». En: 2017. URL: https://arxiv.org/pdf/1706.03762.pdf.

[15] *A beginner's Guide to Large Language models*. URL: https://resources.nvidia.com/en-us-large-language-model-ebooks/llm-ebook-part1.

[16] 1308031437975826437. *Open-sourced training datasets for large language models (llms)*. URL: https://kili-technology.com/large-language-models-llms/9-open-sourced-datasets-for-training-large-language-models.

[17] *GPT3 new LLM from OpenAI*. Jun. de 2023. URL: https://developer.nvidia.com/blog/openai-presents-gpt-3-a-175-billion-parameters-language-model/.

[18] Miguel amp;Aacute;ngel Trabado. *Prompt engineering: El Secreto de la Comunicación Humano/IA*. URL: https://es.linkedin.com/pulse/prompt-engineering-el-secreto-de-la-comunicaci%C3%B3n-humanoia-trabado.

[19] DimensionIA. *Desentrañando el Enigma de la temperatura en los modelos de lenguaje*. Jun. de 2023. URL: https://www.dimensionia.com/temperatura-en-modelos-de-lenguaje.

[20] Borja Colomer. *Gizmodo España Cierra, despide a su plantilla, La Sustituye por una ia y esta la lía*. Sep. de 2023. URL: https://elchapuzasinformatico.com/2023/09/gizmodo-espana-ia-despidos-trabajadores/.

[21] *Salario de ingeniero informatico*. 2023. URL: https://www.jobted.es/salario/ingeniero-inform%C3%A1tico.

[22] *Salario de profesor*. URL: https://www.ehu.eus/es/web/gardentasun-ataria/ordainketak.

[23] 152334H. *Non-determinism in GPT-4 is caused by sparse MOE*. Ago. de 2023. URL: https://152334h.github.io/blog/non-determinism-in-gpt-4/.

[24] Samuel Humeau. *Observing discrepancy in completions with temperature = 0*. Feb. de 2023. URL: https://community.openai.com/t/observing-discrepancy-in-completions-with-temperature-0/73380/2.

[25] Github. *GitHub copilot · your AI pair programmer*. URL: https://github.com/features/copilot/.

[26] OpenAI. *Security portal*. Ago. de 2023. URL: https://trust.openai.com/.

[27] Chip Huyen. *Building LLM applications for production*. Abr. de 2023. URL: https://huyenchip.com/2023/04/11/llm-engineering.html.

[28] Sunil Ramlochan. *Master prompting concepts: Zero-shot and few-shot prompting*. Ago. de 2023. URL: https://www.promptengineering.org/master-prompting-concepts-zero-shot-and-few-shot-prompting/.

[29] AUTOMATIC1111 AUTOMATIC1111. *stable-diffusion-webui-feature-showcase: Feature showcase for stable-diffusion-webui*. URL: https://github.com/AUTOMATIC1111/stable-diffusion-webui-feature-showcase#negative-prompt.

[30] Kubiya. URL: https://www.kubiya.ai/product/overview.

[31] URL: https://www.kubiya.ai/product/secure-devops-automation-tool.

[32] Kubiya. *Jira Kubiya Documentation*. Mayo de 2023. URL: https://docs.kubiya.ai/user-guide/integrations/official-action-stores/jira.

[33] Microsoft, mayo de 2023. URL: https://youtu.be/FaVOtIaWWEg?si=nSJ2_6c7s_e95EEp&t=929.

[34] Dhb-Msft. URL: https://learn.microsoft.com/es-es/DeployOffice/privacy/microsoft-365-copilot.

[35] Springworks. *Adding Internal Documents to the Company Wiki Dashboard: Is it Safe?* 2023. URL: https://support.springworks.in/portal/en/kb/articles/adding-internal-documents-to-the-company-wiki-dashboard-is-it-safe (visitado 18-07-2023).