eman ta zabal zazu

Universidad del País Vasco
Euskal Herriko Unibertsitatea

INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Trabajo de Fin de Grado

Grado en Ingeniería Informática

Ingeniería de Software

Título del trabajo

Martín López de Ipiña Muñoz

Dirección
Galdos Otermin, Aritz
Pereira Varela, Juanan
Azanza Sesé, Maider

1 de mayo de 2025

Resumen

Índice de contenidos

Índice de figuras

Índice de tablas

CAPÍTULO 1

Introducción

CAPÍTULO 2

Antecedentes

Una vez establecido el objetivo de este proyecto en la introducción, en este capítulo se describirán los conceptos generales necesarios para la comprensión de este documento. Para ello, en primer lugar se detalla el proceso de onboarding en proyectos software y las dificultades que presenta, junto con el trabajo previo realizado en este ámbito.

Por otro lado, se explicará a grandes rasgos qué son los agentes basados en grandes modelos de lenguaje (conocidos también por el anglicismo Large Language Models o por sus siglas LLM), su arquitectura, funcionamiento e interacción con herramientas externas. Se introduce además el Model Context Protocol como estándar de comunicación entre estos componentes.

Finalmente, se aborda el estado del arte en arquitecturas de agentes, sus aplicaciones en proyectos software y las técnicas de ajuste de modelos.

2.1. Onboarding en proyectos software

El proceso de incorporación (onboarding) de nuevos desarrolladores de software constituye un desafío persistente para las organizaciones tecnológicas, donde los recién incorporados enfrentan una sobrecarga informativa mientras los desarrolladores senior ven afectada su productividad al destinar tiempo considerable a actividades de formación y mentoría[1].

Si bien prácticas como la designación de mentores han demostrado ser efectivas para facilitar la integración de nuevos miembros, estas incrementan significativamente la carga de trabajo sobre los profesionales experimentados, generando potenciales retrasos en los proyectos[2].

En este contexto, los modelos de lenguaje de gran escala emergen como una alternativa prometedora para transformar el proceso de onboarding, ofreciendo orientación personalizada e

instantánea que podría reducir la dependencia de los desarrolladores senior, preservar la productividad global de los equipos y facilitar una incorporación más eficiente y menos disruptiva[3].

2.1.1. Trabajo previo

La Universidad del País Vasco, en colaboración con LKS NEXT[1], desarrolló el prototipo denominado I Need a Hero (INAH), diseñado para aprovechar el potencial de los LLM en la localización de expertos dentro de la organización[4]. INAH opera en dos fases: primero crea perfiles de "héroes" extrayendo información de currículos de empleados dispuestos a asistir; luego, ante una consulta, utiliza GPT-3.5 para identificar las competencias requeridas y localizar a los profesionales que las poseen.

En una línea de estudio complementaria, un trabajo reciente ha presentado el sistema "Onboarding Buddy", el cual implementa una arquitectura multi-agente que organiza diversos componentes especializados para proporcionar asistencia contextualizada durante la incorporación de nuevos desarrolladores[5].

El sistema fundamenta su funcionamiento en la generación dinámica de planes mediante cadena de pensamiento 2.4.2, evaluados posteriormente para determinar su posible descomposición en sub-tareas procesadas en paralelo por otros agentes.

2.2. Agentes de Grandes Modelos de Lenguaje (LLM)

Los agentes de Inteligencia Artificial son programas informáticos que implementan modelos computacionales para ejecutar diversas funciones específicas del contexto en el que se aplican. Tras siete décadas y media de investigación, los esfuerzos en el campo se han focalizado en agentes basados en grandes modelos de lenguaje.

2.2.1. Modelos LLM

Los Grandes Modelos de Lenguaje son redes neuronales especializadas en el procesamiento del lenguaje natural que funcionan mediante un mecanismo de entrada-salida de tokens 2.2.1. Estos modelos reciben secuencias de tokens como entrada, denominada comúnmente "prompt", y generan secuencias de tokens como salida, aplicando durante este proceso las representaciones y relaciones semánticas aprendidas durante su fase de entrenamiento con extensos corpus textuales [6].

Para comprender el funcionamiento de estos agentes, resulta imprescindible asimilar previamente conceptos como la tokenización, las representaciones vectoriales del lenguaje y el ajuste de dichos modelos.

[1]LKS NEXT: https://www.lksnext.com/es/

Tokens Los tokens constituyen la unidad mínima de texto que el modelo puede procesar. Dado que dichos modelos operan sobre estructuras matemáticas, requieren transformar el lenguaje natural en representaciones matriciales. Para lograr esta conversión, el texto se segmenta en dichas unidades mínimas, que pueden corresponder a caracteres individuales, fragmentos de texto o palabras completas. El conjunto íntegro de estas unidades reconocibles por el modelo configura su vocabulario.

Representaciones vectoriales Constituyen vectores numéricos de dimensionalidad fija que codifican la semántica inherente a cada token. Por ejemplo, una dimensión específica podría especializarse en representar conceptos abstractos. En este contexto, la representación vectorial del token "animal" contendría un valor más elevado en dicha dimensión que la correspondiente al término "gato", reflejando su mayor grado de abstracción conceptual.

Ajuste de modelos instruct El entrenamiento A.2 de los LLM se estructura en dos fases diferenciadas. La primera corresponde al preentrenamiento, donde el modelo procesa extensos conjunto de datos textuales con operaciones como intentar predecir el siguiente token en la secuencia. Esta fase permite al modelo captar las complejas estructuras sintácticas y relaciones semánticas inherentes al lenguaje natural. La segunda fase consiste en el ajuste fino, donde el modelo previamente entrenado se especializa mediante conjuntos de datos específicos y etiquetados A.1, optimizando su capacidad para ejecutar tareas concretas de clasificación o generación de texto.

Los agentes basados en LLM implementan en su mayoría modelos *instruct*, variantes especialmente ajustadas para responder a consultas e instrucciones de usuarios. Dentro de esta categoría se encuentran GPT (base de ChatGPT[2]) de OpenAI[3], Claude Sonnet de Anthropic[4], y LLama-Instruct de Meta[5].

2.2.2. Interacción con herramientas externas

Los agentes LLM poseen la capacidad de interactuar con diversas herramientas como búsquedas web, bases de datos o interfaces de usuario. Fundamentalmente, este tipo de modelos solo genera tokens de texto, por lo que la integración de herramientas se implementa mediante palabras clave o tokens especiales que este puede incluir en su salida. Para ello, en el texto de entrada se especifica el esquema de la función a utilizar y, si decide emplearla, el modelo generará el texto correspondiente. Posteriormente, se procesa la respuesta para extraer llamadas a funciones si las hubiese.

La interacción con herramientas es típicamente alternante. Tras realizar la llamada a la herramienta, la salida de esta se utilizará como entrada para el siguiente mensaje del modelo. La figura 2.1 ilustra el esquema de un agente con acceso a una API del clima. Como el modelo carece

[2]ChatGPT:https://chatgpt.com
[3]OpenAI:https://openai.com/
[4]Anthropic:https://www.anthropic.com/
[5]Meta: https://about.meta.com/es/

de información climática en tiempo real, se le indica en el prompt la posibilidad de invocar esta función. Al incluir la llamada en su texto de salida, se ejecuta la función y su respuesta se transmite al modelo para generar el resultado final.

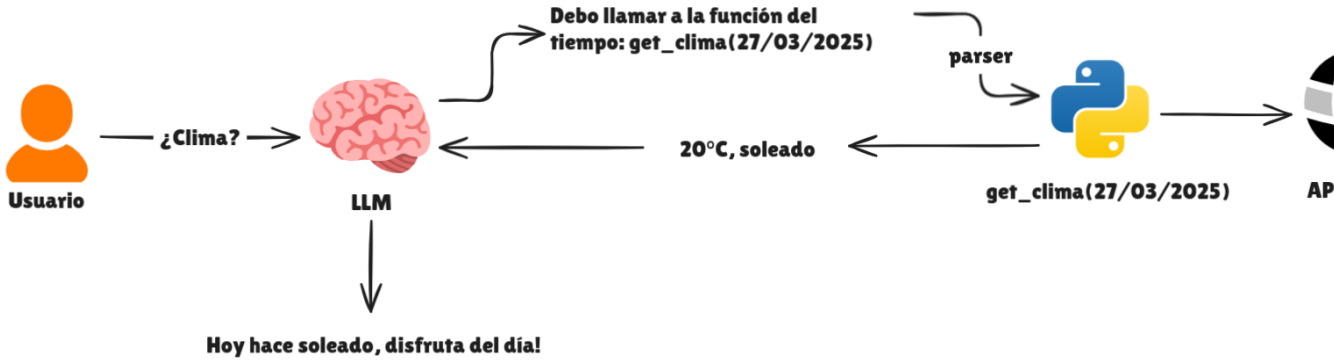

Figura 2.1: Ejemplo de interacción de un modelo LLM con una herramienta externa.

2.2.2.1. Patrón ReAct

El agente Reasoning and Act (ReAct) constituye uno de los patrones más utilizados[7]. Se basa en un ciclo de tres pasos fundamentales: el razonamiento como generación de pensamiento sobre posibles acciones, la acción como ejecución de la herramienta seleccionada y la observación como procesamiento del resultado obtenido. Este ciclo se repite iterativamente hasta que durante la fase de razonamiento se determina que la tarea ha sido completada.

2.2.3. Abstracciones en frameworks

En pleno auge de los agentes LLM, surgen cada vez más bibliotecas y frameworks que estandarizan su implementación. Estos marcos de trabajo ofrecen abstracciones de alto nivel para reutilizar funcionalidades comunes presentes en la mayoría de sistemas de agentes. Las principales funcionalidades proporcionadas son las siguientes:

- **Gestión de modelos:** La ejecución de modelos de lenguaje requiere del dominio de estos, ya que cada uno posee tokenizadores específicos A.4 y esquemas propios de entrada y salida. Los frameworks ofrecen interfaces unificadas, facilitando el uso de diversos modelos sin necesidad de conocimientos técnicos excesivamente detallados.
- **Interacción conversacional:** La comunicación con los agentes se efectúa mediante un esquema conversacional, donde el modelo recibe un texto de entrada y genera una respuesta correspondiente. Las respuestas y entradas se concatenan secuencialmente para preservar el contexto de la conversación, cada consulta subsiguiente incorpora todos los intercambios precedentes.

- **Uso de herramientas externas:** Toda la complejidad de la interacción se abstrae en el framework, por lo que el desarrollador únicamente debe especificar la función que desea incorporar.

- **Interacción entre agentes:** Los agentes pueden establecer comunicación entre sí, permitiendo la construcción de sistemas con mayor complejidad. Algunos frameworks establecen protocolos que definen las modalidades de comunicación entre los distintos agentes.

Para este trabajo utilizaremos fundamentalmente LangChain[6] para la gestión de llamadas a APIs de modelos y prompting, LangGraph[7] para la orquestación de flujos agénticos, y LangSmith[8] para el seguimiento de trazas de llamadas a modelos y herramientas. Para la gestión de los conjuntos de datos y entrenamiento de modelos utilizaremos HuggingFace.[9]

2.3. Model Context Protocol

El Model Context Protocol (MCP), desarrollado por Anthropic, estandariza la comunicación entre agentes LLM y herramientas. Permite que aplicaciones diversas ofrezcan herramientas a agentes externos sin exponer detalles de implementación[8].

La figura 2.2 ilustra el esquema operativo del protocolo. Los desarrolladores de Jira[10] y GitHub[11] han implementado un servidor MCP que traduce las interacciones con sus APIs y proporciona herramientas al cliente MCP. Esto permite que el desarrollador del agente se enfoque exclusivamente en conectar las herramientas del cliente con el agente, sin necesidad de interactuar con APIs externas. Asimismo, el cliente MCP gestiona la comunicación entre servidores, facilitando al agente el acceso directo a múltiples herramientas.

[6] LangChain: https://www.langchain.com/
[7] LangGraph: https://www.langchain.com/langgraph
[8] LangSmith: https://www.langchain.com/langsmith
[9] HuggingFace: https://huggingface.co/
[10] Jira: https://www.atlassian.com/es/software/jira
[11] GitHub: https://github.com/

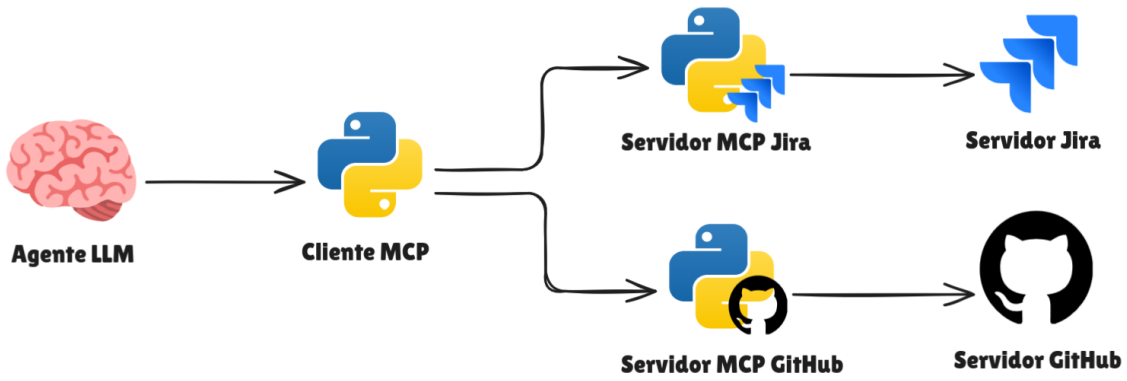

Figura 2.2: Esquema de funcionamiento del Model Context Protocol.

El protocolo ofrece dos modos de operación para establecer la comunicación entre cliente y servidor:

- **Comunicación SSE:** El protocolo Server-Sent Events (SSE) establece un canal de comunicación unidireccional sobre HTTP desde el servidor hacia el cliente. Proporciona actualizaciones en tiempo real con capacidad de streaming. En el protocolo MCP, el cliente efectúa solicitudes para la ejecución de herramientas en el servidor mediante HTTP, a lo que el servidor puede responder mediante eventos SSE.

- **Comunicación STDIO:** El protocolo de entrada y salida estándar (STDIO) facilita la comunicación bidireccional entre cliente y servidor a nivel de proceso en el sistema operativo. Este mecanismo permite el intercambio de información en formato JSON a través de los canales estándar del sistema. Su diseño, orientado principalmente a entornos locales, restringe la conexión a un único cliente por servidor al limitarse a la comunicación entre dos procesos.

La aplicación de escritorio Claude Desktop[12] de Anthropic constituye un reflejo del potencial del protocolo. Esta plataforma ofrece la posibilidad de interactuar con servidores mediante una configuración mínima. Implementando el protocolo STDIO, la aplicación ejecuta los servidores distribuidos por terceros a través de gestores de paquetes como uv[13], npx[14] o Docker[15]. Al incorporar un cliente MCP en la aplicación, consigue integrar las herramientas disponibles en la interfaz de chat con los modelos de Anthropic.

[12] Claude Desktop: https://claude.ai/download
[13] uv: https://astral.sh/blog/uv
[14] npx: https://docs.npmjs.com/cli/v8/commands/npx
[15] Docker: https://www.docker.com/

2.4. Estado del arte en arquitecturas de agentes LLM

La comunidad científica ha diseñado diversas arquitecturas de agentes para optimizar el rendimiento de los modelos disponibles. La arquitectura RAG se distingue por complementar la entrada del modelo con información recuperada de documentos relevantes. Otras propuestas se centran en mejorar la comunicación y coordinación entre agentes.

2.4.1. Arquitectura RAG

Los modelos LLM poseen un conocimiento restringido a los datos con los que fueron entrenados. Para superar esta limitación, la arquitectura RAG (Retrieval-Augmented Generation) complementa la generación del LLM mediante la recuperación de información relevante desde repositorios de conocimiento externos. La figura 2.3 ilustra un ejemplo de su funcionamiento.

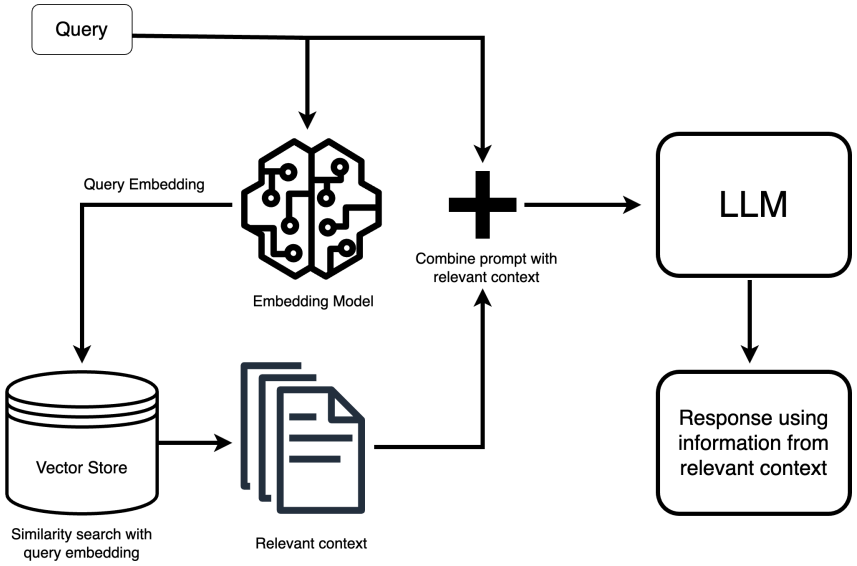

Figura 2.3: Esquema de funcionamiento de la arquitectura RAG en un LLM Fuente.

La recuperación de documentos relevantes se puede implementar mediate recuperadores dispersos: expresiones regulares, búsqueda de n-gramas, palabras clave, entre otras. No obstante, el enfoque predominante consiste en el uso de recuperadores densos, conocidos como indexación vectorial. En este método, los documentos se transforman en vectores, generalmente mediante LLMs especializados en codificación, denominados Embedders. Al representar los documentos en un espacio vectorial, es posible recuperar aquellos semánticamente más pertinentes mediante la comparación del vector de consulta con los vectores de los documentos indexados, utilizando métricas como la distancia coseno A.3.

2.4.1.1. Estrategias RAG avanzadas

La optimización del rendimiento en arquitecturas RAG ha sido ampliamente estudiada [9][10], enfocándose en tres áreas principales: el procesado de documentos, los sistemas de recuperación y la mejora del flujo de generación.

- **Procesado de documentos:** La calidad de la indexación documental determina la eficacia del sistema. Entre las estrategias destacadas figuran la eliminación de ruido textual, el ajuste del tamaño óptimo de los segmentos indexados, y la técnica de ventana solapada. Esta última superpone fragmentos para preservar los datos situados en las intersecciones de los segmentos.

- **Sistemas de recuperación:** Los recuperadores densos basados en representaciones, ilustrados en la figura 2.3, precalculan las representaciones vectoriales de los documentos mediante una indexación independiente de las consultas. Por otro lado, los recuperadores basados en interacción, proponen calcular el vector para cada documento junto al vector de consulta durante el tiempo de ejecución, obteniendo así una representación más rica que captura las relaciones contextuales específicas entre ambos elementos[11][12].

  Para contrarrestar el sobrecoste computacional de este enfoque, se han desarrollado técnicas de indexación híbridas que combinan ambos métodos. Estas técnicas permiten realizar una búsqueda inicial utilizando representaciones vectoriales y posteriormente refinar los resultados mediante la extracción interactiva[13].

- **Optimización del flujo de generación:** Para tareas que requieren reflexión documentada, se han desarrollado sistemas de salto múltiple que alternan entre recuperación y generación[14][15][16][17][18]. Estos flujos permiten al sistema examinar críticamente la información inicialmente recuperada, identificar lagunas informativas, y formular las consultas correspondientes para subsanar dichas carencias.

2.4.2. Arquitecturas de interacción entre agentes

La interacción entre agentes LLM constituye un campo de investigación activo, distinguiéndose diversos avances en módulos de memoria, planificación e interacción multiagente[19].

- **Módulos de memoria**: En una interacción conversacional, el modelo procesa todos los mensajes previos, pudiendo generar un contexto excesivamente amplio. Para mitigar este problema, se han desarrollados módulos de memoria que almacenan información relevante de interacciones pasadas de forma resumida[20][21][22]. Esta memoria puede consultarse posteriormente mediante RAG, permitiendo recuperar los elementos más relevantes según el contexto[23].

Algunos módulos se inspiran en la estructura de la memoria humana[24], incorporando mecanismos que almacenan información con distintas temporalidades y niveles de relevancia[19][25].

- **Planificación**: Los mecanismos de planificación potencian el razonamiento de los agentes sobre sus acciones futuras.

  Entre estos mecanismos destaca el prompting de cadena de pensamiento (Chain of Thought)[26], que instruye al modelo para elaborar un razonamiento secuencial previo a su decisión final, permitiendo así descomponer problemas complejos paso a paso. Partiendo de este enfoque, estrategias avanzadas como la autoconsciencia[22] lo amplían mediante la generación de múltiples cadenas de razonamiento independientes y la posterior selección de la respuesta óptima entre ellas[27][28].

  De manera complementaria, las técnicas de reflexión[29][30][31] implementan un proceso iterativo donde el propio modelo evalúa y refina sus respuestas.

  Por otro lado, la estructuración de acciones constituye una metodología ampliamente adoptada en la planificación[32][33][34]. Esta técnica permite definir planes de alto nivel que posteriormente se desglosan en acciones específicas ejecutables por los agentes[35][36][37][38]. Adicionalmente, la definición de interdependencias entre estas acciones permite verificar la validez de los planes generados[39][40][41].

  Los modelos razonadores como o1 de OpenAI o DeepSeek-R1 incorporan estas técnicas de forma nativa[42]. Estos han sido entrenados con datos que incluye ejemplos de razonamiento y planificación, lo que les permite generar respuestas que incluyen dichas estructuras.

- **Interacción entre agentes:** Los sistemas multi-agente implementan una arquitectura donde diversos agentes especializados son coordinados por un componente central denominado orquestador[43][44]. En este paradigma, cada agente se especializa en una función particular, como la búsqueda de información, la ejecución de herramientas o la generación de texto. El orquestador evalúa las consultas entrantes y las dirige hacia el agente más competente para resolverlas.

  Enfoques complementarios proponen la interacción directa entre agentes especializados como mecanismo de retroalimentación[45][46]. Por ejemplo, ChatDev[47] establece un sistema de colaboración entre agentes programadores, testers y gestores para abordar problemas de ingeniería de software. MetaGPT[48] refina esta propuesta al implementar un protocolo de comunicación basado en el patrón publicador/suscriptor entre los agentes, permitiéndoles difundir información de forma selectiva.

2.5. Agentes LLM en proyectos software

La integración de agentes en proyectos software ha sido objeto de estudio enfocándose principalmente en el ámbito de la generación automática de código. Se pueden destacar tres niveles

de autonomía en los productos desarrollados: sugerencias de autocompletado, asistentes de programación y agentes autónomos.

- **Sugerencias de autocompletado:** GitHub Copilot[16] o Cursor[17] ofrecen la integración de un modelo LLM ajustado para generación de código directamente al entorno del desarrollador. De esta forma, el desarrollador recibe sugerencias de autocompletado en tiempo real mientras escribe el código.

  Este tipo de herramientas han demostrado ser eficaces para aumentar la productividad de los desarrolladores en tareas de programación repetitivas[49]. Sin embargo, debido a su enfoque en la rapidez de respuesta, presentan limitaciones en la aplicación de razonamiento profundo.

- **Asistentes de programación:** Principalmente en aplicaciones en forma de chat, herramientas como GitHub Copilot Chat, Cursor o Cody[18] de SourceGraph[19] ofrecen un sistema de chat interactivo que permite al desarrollador realizar preguntas específicas sobre el código[50].

  Cabe destacar el proyecto SourceGraph, el cual al ser inicialmente de código abierto ha publicado una explicación de su funcionamiento. Proporciona un sistema avanzado de navegación de código que permite la integración del agente LLM denominado Cody.

  La arquitectura de este sistema opera en dos etapas: en primera instancia, los servidores de lenguaje específicos para cada lenguaje de programación analizan la estructura del código fuente, generando un grafo de dependencias que representa las relaciones jerárquicas entre los distintos elementos del código (funciones, clases, métodos, interfaces, entre otros)[51]. Posteriormente, el sistema implementa un mecanismo RAG basado en trigramas sobre el grafo del proyecto, cuyos fragmentos son suministrados al agente Cody, que procesa esta información estructurada y elabora respuestas contextualizadas a las consultas del desarrollador[52].

- **Agentes autónomos:** Con un enfoque más ambicioso, soluciones como Aider[20] o DevinAI[21] persiguen la automatización del ciclo completo de desarrollo de software. Debido a la complejidad inherente a dicha tarea, estas herramientas permanecen en una fase incipiente[53].

  El funcionamiento de Aider presenta similitudes con SourceGraph[54]. En primera instancia, analiza el código fuente mediante la biblioteca Tree-sitter[22], generando un grafo de dependencias que representa las definiciones (funciones, clases, interfaces, etc.) y las referencias

[16] GitHub Copilot: https://github.com/features/copilot
[17] Cursor: https://www.cursor.com/
[18] Cody: https://sourcegraph.com/cody
[19] SourceGraph: https://sourcegraph.com/
[20] Aider: https://aider.chat/
[21] DeviAI: https://devin.ai/
[22] Tree-sitter: https://tree-sitter.github.io/tree-sitter/

entre estas. Posteriormente, implementa un algoritmo de clasificación de grafos para determinar la relevancia de cada nodo respecto al contexto actual. De este modo, incorpora dicho contexto junto con un mapa de los ficheros del proyecto en la entrada del agente LLM.

2.6. Ajuste de modelos para agentes LLM

2.6.1. Limitaciones de los modelos actuales

Los modelos *instruct* del estado del arte poseen la capacidad de resolver tareas que requieren comportamiento agéntico gracias a su amplio conocimiento general. No obstante, su eficiencia es cuestionable, puesto que para tareas específicas pueden carecer de ciertos conocimientos particulares, mientras que disponen de una cantidad considerable de información superflua. Se estima que los modelos más avanzados en la actualidad, como GPT-4o o Claude 3.7 Sonnet, contienen cientos de miles de millones de parámetros[55], lo que los convierte en prohibitivamente costosos para su ejecución en entornos locales.

Debido a estas limitaciones y por tratarse de modelos propietarios, estos están disponibles únicamente a través de API. Este modelo de acceso requiere enviar los datos a servidores externos, planteando así preocupaciones sobre la privacidad y la seguridad de la información. Esta restricción es especialmente relevante en el ámbito del desarrollo de software, donde los datos pueden incluir información sensible o confidencial.

2.6.2. Enfoques de ajuste fino para agentes específicos

Como consecuencia de estas limitación, diversos proyectos de investigación han propuesto el ajuste fino de modelos de menor dimensión para tareas agénticas específicas. Mediante esta aproximación, es posible obtener modelos con mayor precisión en las tareas requeridas, pero con una fracción del coste.

Enfoques como FireAct[56] y AgentTuning[57] proponen el entrenamiento de dichos modelos mediante una estrategia de destilación. Esto consiste en previamente utilizar un modelo instruct de gran tamaño para generar un conjunto de datos sintéticos para las respuestas del agente. Posteriormente, se ajusta un modelo de menor tamaño utilizando este conjunto de datos, de forma que el conocimiento del modelo grande se destila en el modelo pequeño.

Adicionalmente, Agent-FLAN[58] propone la división de dichos datos de entrenamiento en varios conjuntos de aprendizaje. Debido a que los modelos aprenden antes a interpretar el formato de entrada que a razonar sobre la acción a realizar, se propone modificar parte de las trazas de entrenamiento para que estas tengan una estructura conversacional. De esta forma, al ajustar el modelo con las trazas originales, este aprende la estructura de entrada y llamada de herramientas, mientras que en las trazas conversacionales, el modelo aprende a razonar sobre la acción a realizar.

2.6.3. Técnicas de entrenamiento eficiente

El entrenamiento de estos modelos se realiza mediante el ajuste de los pesos de la red neuronal. Debido a que el ajuste fino del modelo completo es costoso, en contextos de bajo coste se propone el uso de la técnica de entrenamiento Low Rank Adaptation (LoRA)[59]. Esta técnica consiste en ajustar únicamente un subconjunto de los pesos de la red neuronal, lo que permite reducir significativamente el coste computacional del ajuste fino.

CAPÍTULO 3

Planificación

En este capítulo se presenta la planificación del proyecto, abordando el alcance definido, los periodos de realización de tareas y los diversos ámbitos de gestión: temporal, de riesgos, de comunicaciones e información, y de partes interesadas. El objetivo de esta planificación es establecer una hoja de ruta estructurada que permita el cumplimiento de todos los objetivos del proyecto dentro de los plazos establecidos.

3.1. Alcance

Este proyecto aborda el desarrollo de un sistema basado en agentes LLM implementado sobre una solución software propia de LKS NEXT, con el propósito de investigar el comportamiento de diversas arquitecturas de agentes definidas en el capítulo 2 y evaluar su viabilidad para su futura incorporación en los procesos productivos de la empresa.

En dicho sistema, se implementarán diversas modalidades de interacción entre agentes especializados en fuentes de datos específicas, evaluando la eficacia de los distintos patrones de comunicación entre dichos componentes.

Cabe destacar que el alcance del proyecto no está definido en su totalidad, dada la complejidad de estimación inherente a su naturaleza exploratoria y al uso de tecnologías emergentes. Para mitigar riesgos en el desarrollo, se ha establecido un protocolo preventivo que contempla reuniones quincenales de seguimiento y control.

3.1.1. Objetivos concretos del proyecto

Para facilitar el logro del objetivo principal, se han identificado y definido los siguientes objetivos específicos que estructuran el avance progresivo del proyecto:

- **Estudio de arquitecturas agénticas:** Realizar un análisis de las diversas arquitecturas de agentes, considerando distintas estrategias de interacción y mecanismos de acceso a fuentes de información.
- **Desarrollo de sistema de Onboarding:** Implementar las arquitecturas propuestas en un proyecto software corporativo, con el propósito de analizar su eficacia en la asistencia a nuevos integrantes durante su proceso de incorporación a la empresa.
- **Integración del Model Context Protocol:** Exploración de las características y beneficios que aporta la implementación del protocolo MCP, con el objetivo de realizar una valoración objetiva para su posible integración en el entorno profesional de la empresa.
- **Evaluación de agentes:** Desarrollar un sistema de evaluación para proporcionar métricas comparativas cuantificables sobre el rendimiento de los diferentes enfoques de agentes.
- **Valoración de ajuste de agentes:** Analizar la relación coste-beneficio asociada al proceso de ajuste fino de modelos LLM para su aplicación en agentes concretos.

Adicionalmente, el proyecto contempla desarrollar el sistema de onboarding incorporando en la medida de lo posible en su base de conocimiento la metodología de trabajo implementada en la empresa. Mediante la adhesión a los estándares definidos en un entorno profesional real, se pretende garantizar que los resultados obtenidos constituyan un reflejo de la viabilidad de implementación y eficacia de proyectos similares en un sistema de explotación.

3.1.2. Requisitos

Los requisitos del proyecto se detallan en profundidad en la sección ref:**siguiente capítulo**

3.1.3. Fases del proyecto

Tal y como se ha mencionado anteriormente, el alcance del proyecto no está completamente definido debido a su naturaleza exploratoria. Consecuentemente, se propone un ciclo de vida iterativo-incremental con iteraciones de aproximadamente dos semanas de duración, permitiendo una adaptación progresiva a los requisitos emergentes.

La primera iteración se centra en la captura de requisitos del proyecto, donde se explorará y definirá el alcance del sistema de agentes a desarrollar. Tras establecer estas bases, la segunda iteración abordará la implementación de un sistema de agentes mínimo que contenga la estructura general del sistema, proporcionando un marco operativo inicial.

Con este sistema mínimo implementado, la tercera iteración corresponderá al desarrollo de un mecanismo de evaluación, con el objetivo de establecer métricas que permitan mejorar el sistema en la cuarta iteración, donde se aplicarán las optimizaciones identificadas. La quinta iteración se

dedicará a la implementación de arquitecturas de agentes exploratorias, evaluando su rendimiento mediante las métricas previamente establecidas.

Finalmente, la sexta iteración contemplará el ajuste fino de un modelo LLM para su integración en un agente específico del sistema, así como su evaluación contextualizada dentro del entorno funcional del proyecto.

Gracias a la implementación del ciclo de vida iterativo, se logra la consecución de los objetivos del proyecto de forma progresiva y estructurada. Este enfoque permite obtener retroalimentación constante por parte de los directores del proyecto durante cada iteración, facilitando así la posibilidad de reorientar la dirección del trabajo ante la aparición de posibles contratiempos.

3.1.4. Descomposición de tareas

La Estructura de Descomposición de Trabajo (EDT) del proyecto se ha creado considerando el ciclo de vida iterativo-incremental del proyecto. La figura 3.1 ilustra un diagrama de esta.

Se divide en los siguientes paquetes:

- **1ª iteración:**
  - **Captura de requisitos (CR):** Actividades orientadas a la definición de los requisitos necesarios para el desarrollo del proyecto.
    - **Definición de requisitos principales:** Identificación y documentación de los requisitos del proyecto, validados con los directores al inicio del mismo.
    - **Elicitación de requisitos:** Recopilación de preguntas potenciales para el sistema desarrollado mediante cuestionario electrónico y análisis posterior de las respuestas.
  - **Recopilación de recursos disponibles (RR):** Actividades centradas en la selección y generación de recursos esenciales para el desarrollo.
    - **Selección de proyecto software:** Identificación y documentación del proyecto software sobre el que se desarrollará el sistema.
    - **Generación de recursos:** Elaboración de documentación complementaria para la implementación del sistema de agentes.
  - **Definición del alcance del sistema de agentes (DA):** Delimitación del alcance considerando implementaciones previas y trabajo académico existente.
    - **Investigación de arquitecturas del estado del arte:** Análisis de las arquitecturas relevantes documentadas en la literatura académica.
    - **Exploración de proyectos similares:** Estudio de implementaciones similares en proyectos software y procesos de Onboarding.
- **2ª iteración:**

- **Sistema de agentes (SA):** Actividades centradas en el diseño e implementación de un sistema de agentes LLM para asistencia en proyectos software.
  - **Diseño del sistema:** Conceptualización e implementación básica de los módulos fundamentales del sistema.
  - **Implementación de agentes especializados:** Desarrollo de agentes adaptados a las diversas fuentes de información disponibles.
  - **Sistema de comunicación mínima:** Creación de un mecanismo básico de orquestación para los agentes implementados.
- **Pruebas y evaluación (P):** Actividades destinadas a verificar y analizar el rendimiento de los módulos implementados.
  - **Pruebas automatizadas:** Desarrollo de pruebas unitarias para algoritmos críticos de los agentes especializados.
  - **Integración continua:** Implementación de un flujo de trabajo automatizado para la ejecución de pruebas unitarias.

- **3ª iteración:**

  - **Sistema de agentes (SA):** Actividades centradas en el diseño e implementación de un sistema de agentes LLM para asistencia en proyectos software.
    - **Sistema de evaluación:** Implementación de mecanismos de evaluación automática sobre el sistema mínimo.
    - **Captura de datos de evaluación:** Anotación manual de ejemplos representativos para evaluar el rendimiento del sistema.
  - **Pruebas y evaluación (P):** Actividades destinadas a verificar y analizar el rendimiento de los módulos implementados.
    - **Evaluación del sistema mínimo:** Ejecución de evaluaciones automatizadas e identificación de elementos clave para mejoras posteriores.

- **4ª iteración:**

  - **Sistema de agentes (SA):** Actividades centradas en el diseño, implementación y evaluación de un sistema de agentes LLM para asistencia en proyectos software.
    - **Mejora de agentes:** Refinamiento de los agentes implementados para optimizar su rendimiento según las métricas establecidas.
    - **Variaciones de orquestación:** Análisis e implementación de estrategias alternativas de orquestación.
  - **Pruebas y evaluación (P):** Actividades destinadas a verificar y analizar el rendimiento de los módulos implementados.

    - **Evaluación comparativa:** Ejecución de evaluaciones automatizadas y análisis comparativo respecto a la iteración anterior.

- **5ª iteración:**

  - **Sistema de agentes (SA):** Actividades centradas en el diseño e implementación de un sistema de agentes LLM para asistencia en proyectos software.
    - **Arquitecturas de interacción alternativas:** Implementación y evaluación de mecanismos adicionales de interacción entre agentes.
    - **Módulos de memoria:** Integración de sistemas de memoria y evaluación de su impacto en el rendimiento global.
    - **Agentes avanzados:** Desarrollo de un agente con un proceso de ejecución extenso para el análisis del coste-beneficio.
  - **Pruebas y evaluación (P):** Actividades destinadas a verificar y analizar el rendimiento de los módulos implementados.
    - **Evaluación comparativa:** Análisis del rendimiento de las nuevas características respecto al sistema precedente.

- **6ª iteración:**

  - **Ajuste de modelo (AM):** Actividades orientadas al entrenamiento especializado de un modelo para un agente específico.
    - **Selección del agente:** Identificación justificada del agente candidato para el ajuste del modelo.
    - **Extracción de datos:** Recopilación automatizada de datos de entrenamiento mediante un modelo de alto rendimiento.
    - **Entrenamiento del modelo:** Diseño y ejecución del ciclo de entrenamiento del modelo LLM seleccionado.
  - **Sistema de agentes (SA):** Actividades centradas en el diseño e implementación de un sistema de agentes LLM para asistencia en proyectos software.
    - **Incorporación del modelo ajustado:** Desarrollo de adaptadores e integración del modelo en el sistema existente.
  - **Pruebas y evaluación (P):** Actividades destinadas a verificar y analizar el rendimiento de los módulos implementados.
    - **Evaluación comparativa:** Análisis del rendimiento del sistema con el modelo ajustado frente a versiones anteriores.

- **Gestión (G):**

  - **Planificación (PL):** Establecimiento de directrices, objetivos y actividades para el desarrollo óptimo del proyecto.
  - **Seguimiento y control (SC):** Monitorización periódica mediante reuniones bisemanales con los directores del proyecto.

- **Trabajo académico (TA):**

  - **Memoria (M):** Elaboración del documento académico que recoge todo el trabajo realizado.
  - **Defensa (D):** Preparación de la presentación y defensa del proyecto ante el tribunal evaluador.

3.2. Periodos de realización de tareas e hitos

En este apartado se detallan las dependencias entre las diferentes tareas, así como la estimación de duración y fechas de cada una de ellas.

3.2.1. Dependencias entre tareas

Las dependencias de los paquetes de trabajo del proyecto requieren una ejecución secuencial planificada. La figura 3.2 ilustra dichas dependencias.

El proyecto se inicia con una primera iteración centrada en la captura de requisitos, invertiendo un tiempo significativo en alinear los objetivos con las necesidades empresariales. Esta fase comprende la captura de requisitos (CR), la recopilación de recursos disponibles (RR) y la definición del alcance y requisitos del sistema de agentes (DA).

Las iteraciones intermedias desarrollan progresivamente el paquete Sistema de agentes (SA) mediante fases complementarias. La segunda iteración implementa su diseño conceptual, agentes especializados y un sistema de comunicación mínima. Posteriormente, en la tercera iteración, el sistema evoluciona incorporando mecanismos de evaluación y captura de datos. La cuarta iteración optimiza los agentes existentes y explora variaciones en la orquestación, mientras que la quinta implementa arquitecturas de interacción alternativas, módulos de memoria y agentes de procesamiento avanzado.

La sexta iteración introduce el paquete Ajuste de modelo (AM), que comprende la selección del agente candidato, extracción de datos y entrenamiento especializado. Simultáneamente, el paquete SA aborda actividades de integración del modelo ajustado en la arquitectura existente.

Transversalmente, se establecen los paquetes de Gestión (G), que abarca actividades de planificación y seguimiento mediante reuniones bisemanales, y Trabajo académico (TA), orientado a la elaboración de la memoria y preparación de la defensa.

Cabe destacar que a partir de la segunda iteración, cada fase incorpora el paquete de Pruebas y evaluación (P), destinado a verificar y analizar el rendimiento de los módulos implementados. Este paquete evoluciona progresivamente desde pruebas automatizadas básicas hasta evaluaciones comparativas que contrastan el rendimiento entre iteraciones sucesivas.

3.2.2. Diagrama de Gantt

La figura 3.3 muestra el diagrama de Gantt, donde se visualiza de manera aproximada la distribución temporal a cada paquete de trabajo durante el transcurso del proyecto. Los rombos negros señalan los hitos clave que se detallan en la sección 3.2.3.

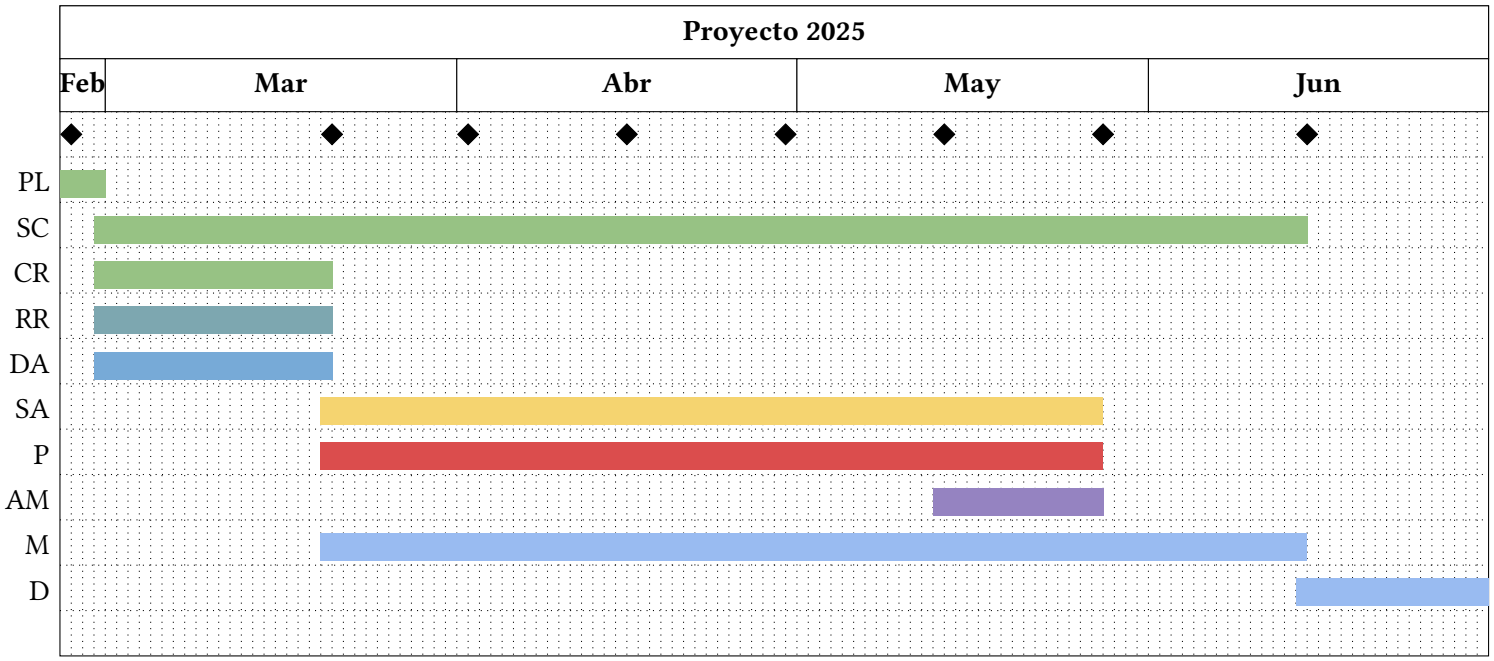

Figura 3.3: Diagrama de Gantt del proyecto

3.2.3. Hitos

En la tabla 3.1 se detallan los hitos establecidos para el desarrollo del proyecto. La finalización de la fase de implementación ha sido programada para el 31 de mayo, tras lo cual se destinarán dos semanas íntegramente a la elaboración de la memoria hasta el 14 de junio, proporcionando así un margen de 8 días previos a la fecha límite de entrega. Este período de contingencia está planificado para abordar posibles contratiempos que pudieran surgir durante el transcurso del proyecto.

Hito	Fecha
Inicio del proyecto	25/02/2025
Fin Iteración 1	20/03/2025
Fin Iteración 2	01/04/2025
Fin Iteración 3	15/04/2025
Fin Iteración 4	29/04/2025
Fin Iteración 5	13/05/2025
Fin Iteración 6	27/05/2025
Fin memoria	14/06/2025
Defensa del proyecto	Por determinar

Tabla 3.1: Cronograma de Hitos del Proyecto

3.3. Gestión del tiempo

Se ha gestionado el tiempo disponible para el proyecto considerando el alcance definido para cumplir todos los objetivos del proyecto.

3.3.1. Estimación de cada tarea

La estimación específica de cada tarea se puede ver en la tabla 3.4

3.4. Gestión de riesgos

Debido al enfoque exploratorio del proyecto, la gestión de riesgos cobra especial importancia. Se han identificado los siguientes riesgos con su correspondiente plan de contingencia:

- **R1- Concurrencia exploratoria:** Dado que el proyecto está enfocado en tecnologías emergentes, existe la posibilidad de que durante el período de desarrollo emerjan iniciativas paralelas que aborden la misma problemática.

  Para abordar este riesgo, se realizará una previa investigación de soluciones existentes antes de implementar cada módulo del sistema, así como un análisis periódico que permita identificar mejoras inspiradas en avances externos.

- **R2- Variabilidad del alcance:** Al ser un proyecto exploratorio, con un alcance inicialmente ambiguo, podría sufrir alteraciones no planificadas. Estos contratiempos podrían suponer un sobrecoste horario, lo que obligaría a replanificar el alcance para no sobrepasar ampliamente los recursos disponibles.

  Para mitigar este riesgo, se implementará un seguimiento y control mediante reuniones quincenales que permitirán evaluar la correcta evolución de las actividades y, en caso

necesario, adoptar medidas correctivas para garantizar la viabilidad del trabajo dentro de los plazos establecidos.

- **R3- Dependencia de sistemas externos:** La implementación del proyecto depende significativamente de sistemas externos, tanto por los modelos de lenguaje accedidos a través de APIs como por los servicios proporcionados por los servidores MCP. Cualquier alteración o interrupción en estos servicios podría comprometer la funcionalidad del sistema implementado.

  Para prevenir este riesgo, se diseñará un sistema con un manejo de excepciones robusto que garantice la continuidad operativa incluso cuando alguno de los módulos experimente fallos. Complementariamente, se adoptará una arquitectura de desarrollo modular que facilite la integración o desacoplamiento de componentes según las necesidades evolutivas del proyecto.

- **R4- Filtrado de credenciales:** El acceso a recursos externos sobre el que se desarrolla el proyecto requiere de claves secretas. Es necesario considerar que existen algoritmos de rastreo automatizados que analizan plataformas públicas en busca de credenciales expuestas para su uso fraudulento. La publicación inadvertida de estas claves en entornos públicos podría ocasionar pérdidas económicas considerables o comprometer la seguridad del sistema corporativo.

  Para contrarrestar este riesgo, se implementará una política de gestión de credenciales mediante variables de entorno, evitando su inclusión directa en el código fuente o su transferencia a plataformas en la nube. Adicionalmente, se mantendrán todos los repositorios y sistemas posibles en modo de visibilidad privada.

- **R5- Pérdida de recursos:** El desarrollo del proyecto se fundamenta en múltiples recursos esenciales, incluyendo el código fuente del sistema, la documentación de requisitos, los diversos artefactos generados durante el proceso de desarrollo y la propia memoria académica. La pérdida de cualquiera de estos elementos debido a fallos técnicos o incidentes fortuitos podría ocasionar un retraso significativo.

  Para mitigar este riesgo, se han implementado los sistemas de información descritos en la sección 3.5.1 que garantizan el mantenimiento de copias de seguridad actualizadas diariamente en la nube. Mediante esta estrategia preventiva, cualquier eventualidad que afecte al dispositivo principal de desarrollo supondría, como máximo, la pérdida del trabajo correspondiente a una jornada.

3.5. Gestión de Comunicaciones e Información

3.5.1. Sistema de información

La gestión de la información del proyecto se ha estructurado mediante los siguientes sistemas tecnológicos:

- **Repositorio GitHub para código fuente:** El código desarrollado será alojado en un repositorio privado de GitHub.
- **Repositorio GitHub para documentación:** La memoria del proyecto, elaborada utilizando LaTeX en el entorno local del alumno, será sincronizada con un repositorio dedicado en GitHub.
- **Almacenamiento en Google Drive:** Los diversos recursos y materiales auxiliares recopilados durante las fases de desarrollo serán almacenados en esta plataforma.

Esta estructura de gestión de la información aporta diversos beneficios al proyecto. Por un lado, facilita la supervisión continua por parte de los directores e implementa un control de versiones organizado que documenta la evolución del trabajo. Por otro lado, garantiza copias de seguridad actualizadas que protegen la integridad de los datos. Adicionalmente, asegura la accesibilidad para todos los interesados.

3.5.2. Sistema de comunicación

La comunicación eficaz entre alumno, director empresarial y directores académicos resulta imprescindible para el correcto seguimiento y control del proyecto. Las herramientas a utilizar son:

- **Correo electrónico:** Canal principal para consultas con los directores del proyecto y comunicación formal con otros miembros de la empresa.
- **Google Meet:** Plataforma destinada a la realización de reuniones telemáticas entre los participantes.
- **Google Chat:** Herramienta complementaria para resolución de consultas rápidas con el director empresarial.

3.6. Herramientas disponibles

Para el desarrollo del proyecto se dispone de varias herramientas que facilitan su implementación y gestión eficiente:

- **IDE PyCharm:** Se usará la versión Professional del entorno de desarrollo PyCharm[1], obtenida a través del paquete educativo de GitHub, que proporciona herramientas especializadas para el desarrollo en Python.
- **Asistencia de herramientas de inteligencia artificial:** Se dispone de herramientas de asistencia basadas en inteligencia artificial como GitHub Copilot y una suscripción al modelo Claude, que optimizan las tareas de programación, el procesamiento documental y la redacción de la presente memoria.
- **Claves de acceso a modelos:** La empresa LKS NEXT ha facilitado las credenciales de acceso necesarias para la integración y ejecución de los diferentes modelos LLM utilizados en el proyecto.
- **Plataformas de computación en la nube:** Se dispone de acceso a infraestructuras de computación en la nube con unidades de procesamiento gráfico (GPU) dedicadas para el entrenamiento e inferencia del modelo ajustado.

[1] PyCharm: https://www.jetbrains.com/es-es/pycharm/

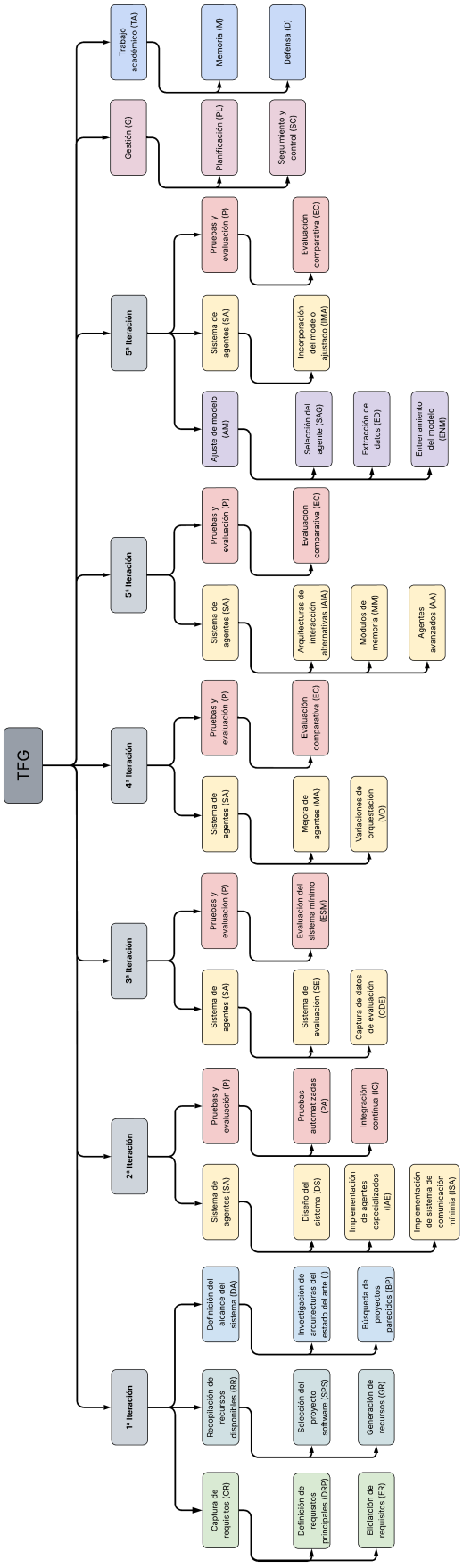

Figura 3.1: Estructura de Descomposición de Trabajo (EDT) del proyecto

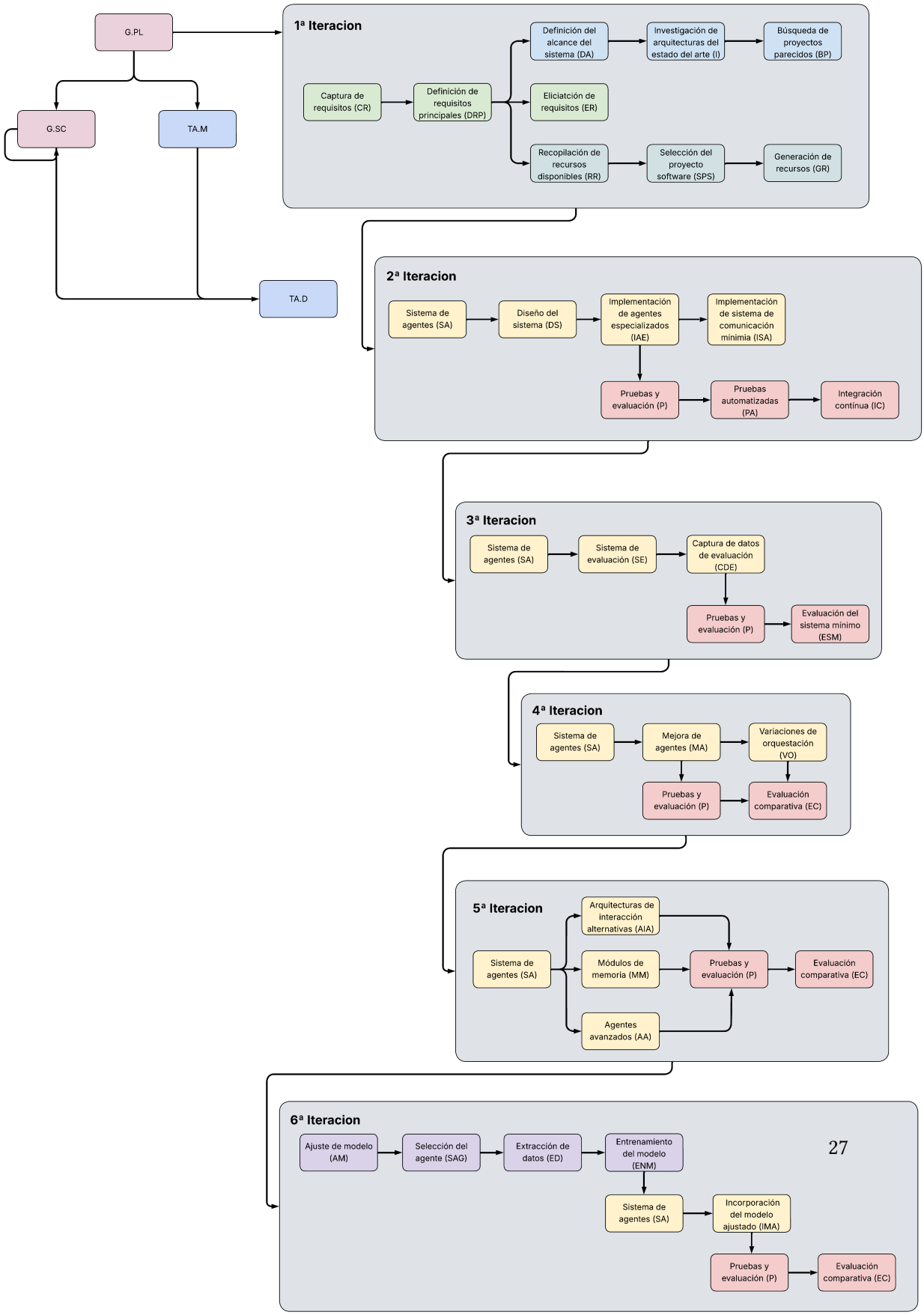

Figura 3.2: Dependencias entre tareas del proyecto

Iteración	Paquete de trabajo	Tarea	Dedicación estimada (H)	Horas totales
1ª Iteración	Captura de requisitos(CR)	Definición de requisitos principales	7	**361**
		Elicitación de requisitos	3	
	Recopilación de recursos disponibles (RR)	Selección del proyecto software	5	
		Generación de recursos	5	
	Definición del alcance del sistema de agentes (DA)	Investigación de arquitecturas del estado del arte	10	
		Búsqueda de proyectos parecidos	5	
		SUBTOTAL	35	
2ª Iteración	Sistema de agentes (SA)	Diseño del sistema	10	
		Implementación de agentes especializados	35	
		Implementación de sistema de comunicación mínimia	10	
	Pruebas y evaluación (P)	Pruebas automatizadas	5	
		Integración contínua	3	
		SUBTOTAL	63	
3ª Iteración	Sistema de agentes (SA)	Sistema de evaluación	20	
		Captura de datos de evaluación	5	
	Pruebas y evaluación (P)	Evaluación del sistema mínimo	5	
		SUBTOTAL	30	
4ª Iteración	Sistema de agentes (SA)	Mejora de agentes	15	
		Variaciones de orquestación	15	
	Pruebas y evaluación (P)	Evaluación comparativa	5	
		SUBTOTAL	35	
5ª Iteración	Sistema de agentes (SA)	Arquitecturas de interacción alternativas	15	
		Módulos de memoria	15	
		Agentes avanzados	15	
	Pruebas y evaluación (P)	Evaluación comparativa	10	
		SUBTOTAL	55	
6ª Iteración	Ajuste de modelo (AM)	Selección del agente	3	
		Extracción de datos	10	
		Entrenamiento del modelo	20	
	Sistema de agentes (SA)	Incorporación del modelo ajustado	5	
	Pruebas y evaluación (P)	Evaluación comparativa	5	
		SUBTOTAL	43	
	Gestión (G)	Planificación (PL)	15	
		Seguimiento y control (SC)	15	
		SUBTOTAL	30	
	Trabajo acdémico	Memoria (M)	60	
		Defensa (D)	10	
		SUBTOTAL	70	

Figura 3.4: Estimación horaria de cada tarea

CAPÍTULO 4

Captura de requisitos

Una vez establecido el objetivo general del proyecto en el capítulo 1 y asentados los conceptos con los que se trabajará en el capítulo 2, en el presente capítulo se detallarán los requisitos establecidos para el desarrollo del sistema.

La captura de requisitos se ha estructurado en dos fases: en primera instancia, se acordaron los requisitos principales y el alcance general con los directores del proyecto; posteriormente, se realizó una captura exhaustiva para determinar los dominios de conocimiento que el sistema debe implementar.

4.1. Requisitos principales

El sistema agéntico implementado debe tener las siguientes características para cumplir con las necesidades del director empresarial:

4.1.1. Requisitos funcionales

- **Agentes especializados:** El sistema contemplará al menos 4 agentes cuya labor esté especializada en responder preguntas sobre fuentes de datos específicas.
- **Fuentes de información:** Los agentes del sistema contarán con acceso al menos al repositorio de código del proyecto software, a una documentación del proyecto externa y a un sistema de gestión de tareas. Al menos una de dichas fuentes debe ser accedida por un sistema RAG.
- **Orquestación:** Se requiere un agente coordinador cuya función sea decidir a qué agente especializado delegar una acción a realizar.

- **Protocolo MCP:** La implementación priorizará el uso del protocolo MCP: Al menos uno de los agentes especializados empleará dicho protocolo para obtener las herramientas a utilizar.
- **Evaluación de agentes:** Se implementará un sistema de evaluación objetivo para determinar si la mejora de agentes incrementa la precisión del sistema. Dado un agente y su mejora, será posible determinar cuál de los dos responde mejor a las preguntas realizadas, utilizando al menos una métrica cuantificable.
- **Enfoque del sistema:** La finalidad principal del sistema consistirá en responder preguntas sobre un proyecto software, logrando una tasa de precisión mínima del 75 % sobre un conjunto predefinido de al menos 25 preguntas. Esta precisión se medirá utilizando una métrica cuantificable previamente establecida.
- **Librerías utilizadas:** El desarrollo se apoyará en un conjunto de librerías especializadas: LangChain para el control de modelos y prompting (empleándose en al menos el 75 % de las ejecuciones de modelos y prompts), LangGraph para el flujo lógico requiriendo que al menos el 75 % de los agentes del sistema se ejecuten sobre un grafo compilado de dicha librería, y LangSmith como herramienta de monitorización, garantizando que toda llamada a los modelos sea visualizable en dicha plataforma.

4.1.2. Requisitos no funcionales

- **Control de excepciones:** El sistema incorporará un control de excepciones que garantice la ejecución aún si un agente o una herramienta genera una excepción. Se contempla la captura de excepciones de todos los agentes y todas las herramientas utilizadas.
- **Control del tiempo:** La ejecución completa del sistema para un caso de uso no excederá los 5 minutos.
- **Control del presupuesto:** Se establece un límite máximo de 25 céntimos por ejecución de un caso de uso para el coste de API de modelos.
- **Independencia de modelos:** La arquitectura garantizará independencia respecto a un único proveedor de modelos LLM. La sustitución de los modelos utilizados será posible en menos de 5 minutos.
- **Modularidad del sistema:** El orquestador funcionará de manera desacoplada respecto a los agentes especializados. La adición o eliminación de un agente especializado requerirá menos de 5 minutos de desarrollo.

4.2. Dominios de conocimiento

De acuerdo con los objetivos previamente establecidos en la sección 3.1.1, es necesario alinear el sistema con la metodología de desarrollo implementada en la organización. Con este propósito,

se ha efectuado una recopilación de las posibles interrogantes dirigidas al sistema, procediendo posteriormente a realizar un análisis de la metodología actualmente vigente en el entorno empresarial.

4.2.1. Anotación de preguntas

Para dicho cometido, se ha llevado a cabo una elicitación de requisitos mediante la implementación un cuestionario electrónico dirigido al personal técnico de la sede de Zuatzu en LKS Next. En dicho cuestionario, se solicitó a los profesionales que, acorde a su ámbito de especialización, anotasen las interrogantes que formularían a un sistema de onboarding hipotético.

Se recabaron un total de 63 interrogantes aportadas por 8 profesionales del ámbito del desarrollo software. Posteriormente, dichas cuestiones fueron ampliadas y categorizadas mediante la utilización del modelo Claude, tras lo cual se procedió a una anotación manual para eliminar elementos redundantes. La compilación clasificada final puede consultarse en el listado B.1, donde se han identificado los siguientes dominios de conocimiento:

- **Información general:** Objetivo, finalidad y contexto del proyecto.
- **Entorno y despliegue:** Entornos de desarrollo y sistemas de despliegue disponibles.
- **Gestión del proyecto:** Comunicación y coordinación del equipo, metodología de contribución y gestión de tareas y requisitos.
- **Estándares y prácticas:** Estándares de codificación, visuales, de integración y aspectos legales.
- **Documentación:** Las fuentes de documentación disponibles para el proyecto y su ubicación.
- **Recursos adicionales:** Recursos externos al proyecto como formaciones y documentación de librerías.
- **Arquitectura del sistema:** Cuestiones de diseño del código fuente sobre todos los niveles de abstracción del modelo C4[60]. Este modelo define los niveles de abstracción de un proyecto software en 4 niveles:
  - Actores y sistemas externos con los que interactúa el sistema
  - Contenedores y aplicaciones que componen el sistema
  - Componentes que definen cada contenedor
  - Diagrama de clases e interfaces de cada componente

4.2.2. Metodología empresarial

La metodología implementada fue consultada directamente con los dearrolladores de la empresa. Al ser LKS Next una consultoría, la metodología implementada depende en gran medida del cliente específico con el que se está trabajando. Esta metodología cumple en su mayoría las siguientes pautas comunes:

- **Gestión del tiempo:** Los integrantes del proyecto utilizan una aplicación propietaria para computar las horas dedicadas al proyecto.
- **Gestión de tareas:** La asignación y monitorización de tareas se realiza mediante aplicaciones especializadas como Jira o Trello.
- **Documentación:** La documentación de los proyectos está dispersa en varias ubicaciones como páginas web dedicadas o dentro del propio repositorio del proyecto.
- **Respositorio del proyecto:** El código fuente del proyecto se gestiona de forma centralizada en un repositorio común, mayoritariamente en repositorios del GitLab propio de LKS Next.

Adicionalmente, el flujo de trabajo entre diseñadores y desarrolladores sigue una metodología especializada. En una reunión con un integrante del equipo de diseño **referencia al acta** se capturó el proceso utilizado:

En primera instancia, se documentan los requisitos visuales a utilizar en un repositorio de Confluence. De esta forma, el equipo de diseño realiza un prototipo visual con herramientas especializadas como Figma. Tras esto, se exportan dichos diseños a maquetas html, los cuales se comparten mediante Google Drive con los desarrolladores del proyecto.

4.3. Recursos a utilizar

Tras definir los requisitos principales y los dominios de conocimeiento necesarios, se ha de definir el contexto específico en el que incorporar el sistema. Para ello, no es posible utilizar un proyecto de un cliente en producción, dadas las restricciones legales tanto académicas como de inteligencia artificial.

4.3.1. Proyecto software

Se ha seleccionado el proyecto propietario IA-core-tools. Este proyecto constituye herramientas de desarrollo de agentes LLM con acceso a fuentes de información mediante RAG. La aplicación está desarrollada en Flask y proporciona una api para desarrollar agentes de forma interactiva con la interfaz visual, de forma que los desarrolladores de LKS Next la utilicen para crear soluciones para los proyectos en producción.

Se dispone de acceso al repositorio de GitLab del proyecto, con información detallada de todos los usuarios, contribuciones, gestión de incidencias y gestión de tareas.

La metodología de gestión dista ligeramente de los proyectos orientados a clientes. El desarrollador líder es Aritz Galdos (director empresarial del presente proyecto), y realiza reuniones tanto presenciales como virtuales con los contribuyentes del proyecto. Dadas las restricciones legales, no se dispone de acceso a la aplicación de gestión horaria.

4.3.2. Recursos generados

Para la correcta evaluación del sistema, se ha generado documentación lo más fidedigna posible al proyecto. Para ello, se ha utilizado el modelo Claude con toda la información de gestión mencionada como entrada. También se han incluido ficheros clave del repositorio del código fuente. Adicionalmente, se ha refinado manualmente la documentaión generada para eliminar información incorrecta.

En primera instancia, se han generado 3 documentos relacionadas con la sección visual del proyecto: *funcionamiento_y_diseño_interfaz*, *guia_de_estilos_visual* y *limitaciones_y_mejoras_pendientes*. Estos documentos se han añadido a un repositorio de Confluence para simular la documentación mencionada por los diseñadores visuales.

A continuación, se han generado 13 documentos que simulan la "documentación oficial" del proyecto. Estos capturan lo necesario para entender la gestión, diseño y metodología de contribución del proyecto a cualquier nueva incorporación: *arquitectura-software.md*, *despliegue.md*, *equipo-y-comunicacion.md*, *estandares-codigo.md*, *flujos-trabajo.md*, *guia-contribucion.md*, *informacion-cliente.md*, *metodologia.md*, *modelo-negocio.md*, *onboarding.md*, *README.md*, *referencias-tecnicas.md* y *sistema-gestion-tareas.md*.

Finalmente, se ha generado la "Documentación API" del proyecto utilizando el agente RepoAgent[61]. Esta documentación contiene la explicación detallada de todas las clases y métodos del proyecto en Python.

APÉNDICE A

Definición de términos técnicos

A.1. Conjunto de datos etiquetados

Un conjunto de datos etiquetado es una colección estructurada de información donde cada elemento o instancia está asociado a una o más categorías, clases o valores objetivo, denominados etiquetas. Estas etiquetas representan la información que se desea predecir o clasificar mediante un modelo de aprendizaje automático.

En el contexto del aprendizaje supervisado, estos conjuntos constituyen la base para el entrenamiento de modelos, ya que proporcionan ejemplos concretos de la relación entrada-salida que el algoritmo debe aprender a generalizar.

A.2. Entrenamiento de redes neuronales

El entrenamiento de una red neuronal consiste en un proceso iterativo de modificación de los pesos de las conexiones entre neuronas artificiales. Estos ajustes permiten que la red aprenda a generalizar a partir de los datos de entrenamiento, extrayendo patrones subyacentes que podrá aplicar posteriormente a datos no observados.

En el aprendizaje supervisado, específicamente durante el ajuste fino, los pesos se modifican comparando las predicciones del modelo con los datos de referencia. Esta comparación se cuantifica mediante una función de pérdida, cuyos gradientes, calculados mediante la regla de la cadena, indican cómo deben ajustarse los pesos para minimizar el error. Este mecanismo de retropropagación permite que la red optimice progresivamente su capacidad predictiva.

A.3. Distancia coseno

La distancia coseno es una medida que cuantifica la similitud entre dos vectores basándose en el coseno del ángulo que forman, independientemente de sus magnitudes. Matemáticamente se expresa como:

$$Similitud_coseno(x, y) = \frac{x \cdot y}{|x||y|}$$

El valor 1 indica vectores perfectamente alineados (máxima similitud), 0 representa vectores perpendiculares (sin similitud) y -1 señala vectores en direcciones opuestas (máxima disimilitud.

A.4. Tokenizador

Un tokenizador es el componente algorítmico encargado de segmentar el texto en unidades mínimas procesables (tokens), implementando reglas específicas de división basadas en espacios, puntuación, subpalabras o patrones predefinidos según el modelo de lenguaje.

APÉNDICE B

Elicitación de preguntas

Los índices numéricos han sido establecidos en base a una captura de datos que fue inicialmente ampliada y posteriormente sometida a un proceso de depuración manual, con el objetivo de incorporar exclusivamente preguntas no redundantes.

La presencia del carácter “e” en el índice denota preguntas de carácter específico vinculadas a ejemplos concretos, esto es, aquellas que precisan de un contexto adicional para su adecuada interpretación. Las preguntas identificadas con el carácter “a” en su índice corresponden a interrogantes anotadas directamente del cuestionario electrónico, preservando su formulación original sin ningún tipo de alteración.

Listing B.1: Listado de elicitación de preguntas procesadas y clasificadas

```
1
2  {
3    "informacion_general": {
4      "intencion_del_proyecto": {
5        "1": "¿Cuál es el objetivo principal y la finalidad del
             proyecto?",
6        "2": "¿Qué problema específico o necesidad resuelve este
             proyecto?"
7      },
8      "funcionalidades_del_proyecto": {
9        "3": "¿Cuáles son las funcionalidades principales que
             incluye el proyecto?",
10       "4": "¿Qué funcionalidades están explícitamente fuera del
             alcance del proyecto?"
11     },
```

```
12      "estructura_del_proyecto": {
13        "5": "¿Cuál es la estructura organizativa general del
              proyecto a nivel de repositorios o subproyectos?",
14        "79a": "¿Cómo están organizados los módulos, componentes y
              paquetes dentro del proyecto?"
15      },
16      "contexto_de_negocio": {
17        "6": "¿Cuáles son los requisitos funcionales detallados del
               proyecto?",
18        "7": "¿Cuáles son los requisitos no funcionales (
              rendimiento, seguridad, escalabilidad, etc.) del
              proyecto?",
19        "8": "¿Existe documentación formal del modelo de negocio o
              dominio? ¿Dónde se encuentra?",
20        "80ae": "¿Dónde puedo encontrar los requerimientos
              funcionales documentados para entender el problema a
              resolver?"
21      },
22      "repositorio_codigo": {
23        "140a": "¿Cuál es la URL completa del repositorio de código
               y cómo puedo acceder a él?"
24      }
25    },
26    "entorno_y_despliegue": {
27      "guias_existentes": {
28        "9": "¿Existen guías o manuales de despliegue para el
              proyecto? ¿Dónde puedo encontrarlas?"
29      },
30      "entornos_disponibles": {
31        "10a": "¿Qué entornos están configurados o están
              disponibles para el proyecto (desarrollo, pruebas,
              preproducción, producción, etc.)?",
32        "11a": "¿Qué credenciales o permisos necesito para acceder
              a cada entorno (VPN, usuarios, certificados, etc.)?"
33      },
34      "configuracion_entorno_desarrollo": {
35        "12": "¿Cuál es el proceso paso a paso para configurar mi
              entorno de desarrollo local (IDE, herramientas, plugins)
              ?",
36        "13a": "¿Cómo compilo y ejecuto el proyecto en mi entorno
              local? ¿Qué comandos debo utilizar?",
```

```
      "84a": "¿Qué IDE o editor es recomendado para este proyecto y qué configuraciones específicas requiere?",
      "85a": "¿Cómo configuro mi entorno de desarrollo para integrarlo con los sistemas corporativos?",
      "86a": "¿Cuál es el proceso completo para compilar el proyecto y verificar que funciona correctamente?"
    },
    "despliegue": {
      "14": "¿Qué sistema o plataforma se utiliza para el despliegue de aplicaciones?",
      "15": "¿Cuál es el proceso detallado de despliegue, incluyendo configuraciones, tecnologías y herramientas utilizadas?",
      "87a": "¿Existen pipelines DevOps implementados para la compilación y despliegue en los diferentes entornos?"
    }
  },
  "gestion_del_proyecto": {
    "equipo_comunicacion_coordinacion": {
      "comunicacion": {
        "16": "¿Cuáles son los canales oficiales de comunicación del equipo (chat, email, videollamadas)?",
        "17": "¿Cómo puedo contactar a cada miembro del equipo y cuál es su rol o área de responsabilidad?"
      },
      "roles": {
        "18": "¿Quién es el líder del proyecto o responsable final de las decisiones?",
        "19": "¿Quiénes son los responsables de cada subsistema o área del proyecto y cuáles son sus responsabilidades específicas?"
      },
      "reuniones_ceremonias": {
        "21": "¿Qué reuniones periódicas o ceremonias están establecidas en el proyecto y cuál es su propósito?",
        "88a": "¿Con qué frecuencia se realizan reuniones de equipo o seguimiento del proyecto?",
        "89a": "¿Cuál es la periodicidad de las reuniones (diarias, semanales, etc.) y su duración estimada?",
        "90a": "¿Cuáles son los objetivos y entregables esperados para cada tipo de reunión?"
```

```
62        }
63      },
64      "metodologia_contribucion": {
65        "23": "¿Dónde puedo encontrar las guías oficiales de
             contribución al proyecto?",
66        "24": "¿Cuál es el proceso completo para contribuir código
             al proyecto, desde la asignación hasta la integración?",
67        "25": "¿Existen tareas marcadas como 'good first issues'
             para nuevos contribuyentes? ¿Dónde puedo encontrarlas?",
68        "91a": "¿Cuál es el procedimiento detallado para entregar
             una tarea completada (revisión, validación, merge)?"
69      },
70      "gestion_tareas_requisitos": {
71        "sistema_gestion_tareas": {
72          "26": "¿Qué herramienta específica se utiliza para
               gestionar las tareas del proyecto (Jira, Trello,
               GitHub Projects, etc.)?",
73          "27": "¿En qué ubicación exacta dentro del sistema de
               gestión están descritas las tareas pendientes?",
74          "28": "¿Cómo se identifican y clasifican las tareas por
               prioridad o urgencia?",
75          "92a": "¿Qué herramienta de gestión de tareas se utiliza
               en el proyecto y cómo accedo a ella?",
76          "93a": "¿Dónde puedo encontrar la descripción detallada
               de las tareas asignadas o disponibles?",
77          "94ae": "¿Qué tareas específicas tengo asignadas
               actualmente o debo realizar?",
78          "95a": "¿Cómo puedo identificar cuáles son las tareas más
                urgentes o prioritarias en este momento?"
79        },
80        "gestion_requisitos": {
81          "29": "¿En qué sistema o plataforma se documentan y
               gestionan los requisitos del proyecto?",
82          "30": "¿Cuál es el proceso establecido para analizar,
               validar y aprobar nuevos requisitos?",
83          "96a": "¿Dónde están registrados formalmente los
               requisitos del proyecto (Jira, Confluence, documentos,
                hojas de cálculo)?"
84        }
85      },
86      "informacion_cliente": {
```

```
      "97a": "¿Quién es el cliente final o usuario principal de esta aplicación y cuál es su contexto de uso?",
      "98a": "¿Qué nivel de participación tiene el cliente en el proceso de desarrollo y toma de decisiones?",
      "99ae": "¿Es necesario consultar directamente al cliente para resolver dudas sobre determinadas funcionalidades o requisitos?"
    }
  },
  "estandares_y_practicas": {
    "estandares_nomenclatura_organizacion": {
      "33": "¿Cuáles son los estándares definidos para la nomenclatura y gestión de branches, commits y pull requests?",
      "34": "¿Cuál es el proceso completo para entregar una tarea, desde su finalización hasta la aprobación?",
      "100a": "¿Existe un estándar documentado para la nomenclatura de branches, commits y otros elementos del repositorio?"
    },
    "estandares_codigo": {
      "35": "¿Para qué lenguajes de programación existen estándares de codificación definidos en el proyecto?",
      "101a": "¿Se utiliza alguna guía de estilo o formato de código específico para cada lenguaje del proyecto?",
      "102a": "¿Existe una estructura o arquitectura de código predefinida que deba seguirse?"
    },
    "estandares_diseno": {
      "36": "¿Existe un sistema de diseño o guía de estilos para la interfaz de usuario?",
      "103a": "¿Hay diseños en Figma u otra herramienta para las nuevas pantallas o componentes a desarrollar?",
      "104a": "¿Dónde puedo encontrar la documentación sobre el diseño visual y la experiencia de usuario a implementar?",
      "106a": "¿Cuál es el procedimiento a seguir si no existen diseños definidos para un componente o pantalla?"
    },
    "practicas": {
      "ci_cd": {
```

```
111         "37": "¿Qué procesos de Integración Continua y Despliegue
                Continuo (CI/CD) están implementados?",
112         "107a": "¿Cuál es el flujo completo de integración
                continua, desde el commit hasta la validación?",
113         "108a": "¿Qué herramientas específicas se utilizan para
                los procesos de integración y despliegue continuo?"
114       },
115       "ci": {
116         "38": "¿Cómo funciona detalladamente el proceso de
                integración continua y qué validaciones incluye?"
117       },
118       "cd": {
119         "39": "¿Cómo se ejecuta el proceso de despliegue continuo
                y qué entornos abarca?"
120       }
121     },
122     "aspectos_legales": {
123       "40": "¿Qué licencias de software se utilizan en el
              proyecto y sus dependencias?",
124       "41": "¿Cuáles son las consideraciones legales específicas
              que deben tenerse en cuenta durante el desarrollo?",
125       "110a": "¿Cuál es el protocolo para gestionar adecuadamente
                las licencias de componentes externos?"
126     },
127     "estandares_seguridad": {
128       "42": "¿Qué herramientas o procesos se utilizan para
              identificar vulnerabilidades de seguridad en el código?"
              ,
129       "44": "¿Cómo se gestionan y auditan las dependencias desde
              la perspectiva de seguridad?",
130       "112a": "¿Cómo verifico que las dependencias externas que
              utilizo son seguras y se mantienen actualizadas?",
131       "113a": "¿Cuál es el procedimiento a seguir si descubro una
                vulnerabilidad crítica en una dependencia?",
132       "114a": "¿Cuáles son las mejores prácticas de seguridad
              establecidas que debo aplicar en mi código para este
              proyecto?"
133     },
134     "pruebas_calidad": {
135       "45": "¿Qué tipos de pruebas se realizan en el proyecto (
              unitarias, funcionales, integración, rendimiento)?",
```

```
      "46": "¿Cuál es la política establecida para la ejecución
         de pruebas (por commit, por merge request, al final del
         sprint)?",
      "47a": "¿Cómo se automatizan las pruebas y qué herramientas
          y tecnologías se utilizan para ello?",
      "48": "¿Cuál es el porcentaje actual de cobertura de
         pruebas y cuál es el objetivo establecido?",
      "115a": "¿Existen pruebas unitarias implementadas para el c
         ódigo actual? ¿Dónde se encuentran?",
      "116a": "¿Qué clases o métodos tienen pruebas unitarias
         documentadas y cuáles necesitan implementación?"
    }
  },
  "documentacion": {
    "49a": "¿Qué fuentes de documentación existen para el
       proyecto y dónde puedo encontrarlas (API, guías, licencias
       , estándares)?",
    "50": "¿Cuál es la estructura organizativa de la documentació
       n del proyecto?",
    "51e": "¿Qué documentación específica debo consultar para
       realizar esta tarea concreta?",
    "52": "¿Cuál es el proceso para modificar o actualizar la
       documentación del proyecto?",
    "53": "¿Cuál es el procedimiento establecido para documentar
       cambios en el código?",
    "118a": "¿Existe un espacio de documentación centralizado
       como Confluence para el proyecto?",
    "120a": "¿Es obligatorio documentar los cambios realizados en
        el código? ¿Qué nivel de detalle se requiere?"
  },
  "recursos_adicionales": {
    "54e": "¿Existen preguntas frecuentes o discusiones en
       StackOverflow relacionadas con este tema o tecnología?",
    "124ae": "¿Dónde puedo encontrar la documentación técnica
       actualizada para las tecnologías o herramientas especí
       ficas que necesito utilizar?",
    "123a": "¿Qué herramientas o utilidades podrían ayudarme a
       ejecutar mis tareas de manera más eficiente?",
    "formaciones": {
      "55e": "¿Qué recursos formativos están disponibles sobre
         estas tecnologías y cuáles son más relevantes para mi
```

```
        tarea actual?",
      "122ae": "¿Dónde puedo encontrar material de formación
        específico para las tecnologías utilizadas en este
        proyecto?"
    }
  },
  "arquitectura_del_sistema": {
    "tecnologias_y_plataformas": {
      "stack_tecnologico": {
        "56a": "¿Cuáles son todas las tecnologías, frameworks y
          lenguajes utilizados en el proyecto?",
        "57": "¿Para qué componente o funcionalidad específica se
           utiliza cada tecnología del proyecto?",
        "129a": "¿Qué herramientas específicas se utilizan para
          gestionar las migraciones de esquemas de base de datos
          ?"
      },
      "plataformas_disponibles": {
        "58": "¿Qué plataformas o herramientas de soporte están
          disponibles para el proyecto (diseño, colaboración,
          monitoreo)?"
      }
    },
    "dependencias": {
      "126a": "¿Qué herramientas o procesos se utilizan para
        gestionar las dependencias en este proyecto?",
      "127a": "¿Cuál es el procedimiento para revisar, actualizar
         o reemplazar dependencias vulnerables o desactualizadas
        ?"
    },
    "modelo_c4": {
      "nivel_1_contexto_sistema": {
        "actores": {
          "63": "¿Quiénes son los actores o usuarios que interact
            úan con el sistema?",
          "64": "¿De qué manera específica interactúa cada tipo
            de actor con el sistema?",
          "65": "¿Cuáles son los niveles de permiso o roles
            definidos para cada tipo de actor en el sistema?"
        },
        "sistemas_externos": {
```

```
184           "66": "¿Qué sistemas externos se integran o comunican
                  con este sistema?",
185           "67": "¿Mediante qué protocolos, estándares o
                  interfaces se conectan los sistemas externos?"
186         }
187       },
188       "nivel_2_contenedores": {
189         "68": "¿Qué aplicaciones, servicios o componentes
                  principales conforman el sistema y cuál es la función
                  de cada uno?",
190         "105a": "¿La aplicación está diseñada para funcionar en m
                  últiples plataformas o dispositivos? ¿Cuáles?",
191         "130a": "¿Qué estrategias o patrones se aplican para
                  optimizar el rendimiento de las consultas a bases de
                  datos?",
192
193         "comunicacion_entre_contenedores": {
194           "69": "¿Qué protocolos, patrones o estándares se
                  utilizan para la comunicación entre los diferentes
                  contenedores?"
195         },
196         "dependencias_entre_contenedores": {
197           "70": "¿Cómo se gestionan las dependencias y el orden
                  de inicio entre los diferentes contenedores?"
198         },
199         "tecnologias": {
200           "71": "¿Qué tecnologías, frameworks o bibliotecas espec
                  íficas utiliza cada contenedor o servicio?"
201         },
202         "evaluacion_y_mejora": {
203           "138a": "¿El enfoque de desarrollo actual es óptimo o
                  existen oportunidades de mejora identificadas?",
204           "139a": "¿Cuál sería el costo en tiempo y recursos para
                   optimizar el proceso de desarrollo actual?"
205         }
206       },
207       "nivel_3_diagrama_componentes": {
208           "72": "¿Cuáles son los componentes internos de cada
                  contenedor y cuál es la función específica de cada
                  uno?",
209           "131a": "¿Cuál es la arquitectura de integración entre
```

```
            los diferentes servicios o componentes del sistema?"
            ,
          "comunicacion_entre_componentes": {
            "73e": "¿Qué patrones o protocolos de comunicación se
                utilizan entre los componentes dentro de un mismo
                contenedor?"
          },
          "dependencias_entre_componentes": {
            "74": "¿Cómo se gestionan las dependencias y el ciclo
                de vida entre los componentes de diferentes
              contenedores?"
          },
          "tecnologias": {
            "71": "¿Qué tecnologías, frameworks o bibliotecas
              específicas utiliza cada componente?"
          }
        },
      "nivel_4_diagrama_codigo": {
        "estructura_diagrama_clases": {
          "75e": "¿Cuál es la estructura detallada de clases,
              interfaces y objetos dentro de un componente especí
              fico?",
          "76e": "¿Cuál es la responsabilidad y función principal
               de cada clase o interfaz dentro del componente?",
          "132ae": "¿Puedes proporcionar un diagrama de paquetes
              y clases para entender la estructura del código?",
          "133ae": "¿Puedes mostrarme la jerarquía completa de
              llamadas para este método específico?",
          "134ae": "¿Cuáles son los métodos más complejos o difí
              ciles de entender en el código y por qué?"
        },
        "patrones_diseno": {
          "77": "¿Qué patrones de diseño, estructuras de herencia
               o composición se implementan en el código?",
          "135a": "¿Qué patrones arquitectónicos o de diseño se
              utilizan en el proyecto (MVC, MVVM, etc.)?",
          "136ae": "¿Por qué se eligieron estos patrones
              arquitectónicos o de diseño específicos?",
          "137ae": "¿Qué arquitectura o patrones debo implementar
               para mi desarrollo actual?"
        },
```

```
        "principios_diseno": {
          "78": "¿Qué principios de diseño (SOLID, DRY) o buenas
              prácticas de código se aplican en el proyecto?"
        }
      }
    }
  }
}
```

Bibliografía

[1] S.E. Sim and R.C. Holt. The ramp-up problem in software projects: a case study of how software immigrants naturalize. In *Proceedings of the 20th International Conference on Software Engineering*, pages 361–370. ISSN: 0270-5257. URL: https://ieeexplore.ieee.org/abstract/document/671389, doi:10.1109/ICSE.1998.671389. Ver página 3.

[2] Igor Steinmacher, Marco Aurelio Graciotto Silva, Marco Aurelio Gerosa, and David F. Redmiles. A systematic literature review on the barriers faced by newcomers to open source software projects. 59:67–85. URL: https://www.sciencedirect.com/science/article/pii/S0950584914002390, doi:10.1016/j.infsof.2014.11.001. Ver página 3.

[3] Eva Ritz, Fabio Donisi, Edona Elshan, and Roman Rietsche. Artificial socialization? how artificial intelligence applications can shape a new era of employee onboarding practices. URL: http://hdl.handle.net/10125/102648, doi:10.24251/HICSS.2023.020. Ver página 4.

[4] Maider Azanza, Juanan Pereira, Arantza Irastorza, and Aritz Galdos. Can LLMs facilitate onboarding software developers? an ongoing industrial case study. In *2024 36th International Conference on Software Engineering Education and Training (CSEE&T)*, pages 1–6. ISSN: 2377-570X. URL: https://ieeexplore.ieee.org/document/10662989, doi:10.1109/CSEET62301.2024.10662989. Ver página 4.

[5] Andrei Cristian Ionescu, Sergey Titov, and Maliheh Izadi. A multi-agent onboarding assistant based on large language models, retrieval augmented generation, and chain-of-thought. URL: http://arxiv.org/abs/2503.23421, arXiv:2503.23421[cs], doi:10.48550/arXiv.2503.23421. Ver página 4.

[6] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. URL: http://arxiv.org/abs/1706.03762, arXiv:1706.03762[cs], doi:10.48550/arXiv.1706.03762. Ver página 4.

[7] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. URL: http://arxiv.org/abs/2210.03629, arXiv:2210.03629[cs], doi:10.48550/arXiv.2210.03629. Ver página 6.

[8] Model context protocol (MCP). URL: https://docs.anthropic.com/en/docs/agents-and-tools/mcp. Ver página 7.

[9] Fengbin Zhu, Wenqiang Lei, Chao Wang, Jianming Zheng, Soujanya Poria, and Tat-Seng Chua. Retrieving and reading: A comprehensive survey on open-domain question answering. URL: http://arxiv.org/abs/2101.00774, arXiv:2101.00774[cs], doi:10.48550/arXiv.2101.00774. Ver página 10.

[10] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. URL: http://arxiv.org/abs/2312.10997, arXiv:2312.10997[cs], doi:10.48550/arXiv.2312.10997. Ver página 10.

[11] Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. Query rewriting for retrieval-augmented large language models. Ver página 10.

[12] Yoav Levine, Itay Dalmedigos, Ori Ram, Yoel Zeldes, Daniel Jannai, Dor Muhlgay, Yoni Osin, Opher Lieber, Barak Lenz, Shai Shalev-Shwartz, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. Standing on the shoulders of giant frozen language models. URL: http://arxiv.org/abs/2204.10019, arXiv:2204.10019[cs], doi:10.48550/arXiv.2204.10019. Ver página 10.

[13] Omar Khattab, Christopher Potts, and Matei Zaharia. Relevance-guided supervision for OpenQA with ColBERT. URL: http://arxiv.org/abs/2007.00814, arXiv:2007.00814[cs], doi:10.48550/arXiv.2007.00814. Ver página 10.

[14] Omar Khattab, Keshav Santhanam, Xiang Lisa Li, David Hall, Percy Liang, Christopher Potts, and Matei Zaharia. Demonstrate-search-predict: Composing retrieval and language models for knowledge-intensive NLP. URL: http://arxiv.org/abs/2212.14024, arXiv:2212.14024[cs], doi:10.48550/arXiv.2212.14024. Ver página 10.

[15] Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9248–9274. Association for Computational Linguistics. URL: https://aclanthology.org/2023.findings-emnlp.620/, doi:10.18653/v1/2023.findings-emnlp.620. Ver página 10.

[16] Peng Qi, Haejun Lee, Oghenetegiri "TG"Sido, and Christopher D. Manning. Answering open-domain questions of varying reasoning steps from text. URL: http://arxiv.org/abs/2010.12527, arXiv:2010.12527[cs], doi:10.48550/arXiv.2010.12527. Ver página 10.

[17] Huaixiu Steven Zheng, Swaroop Mishra, Xinyun Chen, Heng-Tze Cheng, Ed H. Chi, Quoc V. Le, and Denny Zhou. Take a step back: Evoking reasoning via abstraction in large language models. URL: http://arxiv.org/abs/2310.06117, arXiv:2310.06117[cs], doi:10.48550/arXiv.2310.06117. Ver página 10.

[18] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. URL: http://arxiv.org/abs/2212.10509, arXiv:2212.10509[cs], doi:10.48550/arXiv.2212.10509. Ver página 10.

[19] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Jirong Wen. A survey on large language model based autonomous agents. 18(6):186345. doi:10.1007/s11704-024-40231-1. Ver páginas 10, 11.

[20] Hongxin Zhang, Weihua Du, Jiaming Shan, Qinhong Zhou, Yilun Du, Joshua B. Tenenbaum, Tianmin Shu, and Chuang Gan. Building cooperative embodied agents modularly with large language models. URL: http://arxiv.org/abs/2307.02485, arXiv:2307.02485[cs], doi:10.48550/arXiv.2307.02485. Ver página 10.

[21] Kevin A. Fischer. Reflective linguistic programming (RLP): A stepping stone in socially-aware AGI (SocialAGI). URL: http://arxiv.org/abs/2305.12647, arXiv:2305.12647[cs], doi:10.48550/arXiv.2305.12647. Ver página 10.

[22] Xinnian Liang, Bing Wang, Hui Huang, Shuangzhi Wu, Peihao Wu, Lu Lu, Zejun Ma, and Zhoujun Li. *Unleashing Infinite-Length Input Capacity for Large-scale Language Models with Self-Controlled Memory System*. doi:10.48550/arXiv.2304.13343. Ver páginas 10, 11.

[23] Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM agents are experiential learners. 38(17):19632–19642. Number: 17. URL: https://ojs.aaai.org/index.php/AAAI/article/view/29936, doi:10.1609/aaai.v38i17.29936. Ver página 10.

[24] Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. MemoryBank: Enhancing large language models with long-term memory. 38(17):19724–19731. Number: 17. URL: https://ojs.aaai.org/index.php/AAAI/article/view/29946, doi:10.1609/aaai.v38i17.29946. Ver página 11.

[25] Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, pages 1–22. ACM. URL: https://dl.acm.org/doi/10.1145/3586183.3606763, doi:10.1145/3586183.3606763. Ver página 11.

[26] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. URL: http://arxiv.org/abs/2201.11903, arXiv:2201.11903[cs], doi:10.48550/arXiv.2201.11903. Ver página 11.

[27] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. Ver página 11.

[28] Yancheng Wang, Ziyan Jiang, Zheng Chen, Fan Yang, Yingxue Zhou, Eunah Cho, Xing Fan, Xiaojiang Huang, Yanbin Lu, and Yingzhen Yang. RecMind: Large language model powered agent for recommendation. URL: http://arxiv.org/abs/2308.14296, arXiv:2308.14296[cs], doi:10.48550/arXiv.2308.14296. Ver página 11.

[29] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. Ver página 11.

[30] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. SELF-REFINE: Iterative refinement with self-feedback. Ver página 11.

[31] Ning Miao, Yee Whye Teh, and Tom Rainforth. SelfCheck: Using LLMs to zero-shot check their own step-by-step reasoning. URL: http://arxiv.org/abs/2308.00436, arXiv:2308.00436[cs], doi:10.48550/arXiv.2308.00436. Ver página 11.

[32] Bill Yuchen Lin, Yicheng Fu, Karina Yang, Faeze Brahman, Shiyu Huang, Chandra Bhagavatula, Prithviraj Ammanabrolu, Yejin Choi, and Xiang Ren. SWIFTSAGE: A generative agent with fast and slow thinking for complex interactive tasks. Ver página 11.

[33] Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. Ver página 11.

[34] Zihao Wang, Shaofei Cai, Guanzhou Chen, Anji Liu, Xiaojian Ma, and Yitao Liang. Describe, explain, plan and select: Interactive planning with large language models enables open-world multi-task agents. URL: http://arxiv.org/abs/2302.01560, arXiv:2302.01560[cs], doi:10.48550/arXiv.2302.01560. Ver página 11.

[35] Xizhou Zhu, Yuntao Chen, Hao Tian, Chenxin Tao, Weijie Su, Chenyu Yang, Gao Huang, Bin Li, Lewei Lu, Xiaogang Wang, Yu Qiao, Zhaoxiang Zhang, and Jifeng Dai. Ghost in the minecraft: Generally capable agents for open-world environments via large language models with text-based knowledge and memory. URL: http://arxiv.org/abs/2305.17144, arXiv:2305.17144[cs], doi:10.48550/arXiv.2305.17144. Ver página 11.

[36] Chan Hee Song, Brian M. Sadler, Jiaman Wu, Wei-Lun Chao, Clayton Washington, and Yu Su. LLM-planner: Few-shot grounded planning for embodied agents with large language models. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2986–2997. IEEE. URL: https://ieeexplore.ieee.org/document/10378628/, doi:10.1109/ICCV51070.2023.00280. Ver página 11.

[37] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. URL: http://arxiv.org/abs/2305.16291, arXiv:2305.16291[cs], doi:10.48550/arXiv.2305.16291. Ver página 11.

[38] Shunyu Liu, Yaoru Li, Kongcheng Zhang, Zhenyu Cui, Wenkai Fang, Yuxuan Zheng, Tongya Zheng, and Mingli Song. Odyssey: Empowering minecraft agents with open-world skills. URL: http://arxiv.org/abs/2407.15325, arXiv:2407.15325[cs], doi:10.48550/arXiv.2407.15325. Ver página 11.

[39] Shreyas Sundara Raman, Vanya Cohen, Eric Rosen, Ifrah Idrees, David Paulius, and Stefanie Tellex. Planning with large language models via corrective re-prompting. Ver página 11.

[40] LLM+p: Empowering large language models with optimal planning proficiency. URL: http://arxiv.org/abs/2304.11477, doi:10.48550/arXiv.2304.11477. Ver página 11.

[41] Gautier Dagan, Frank Keller, and Alex Lascarides. Dynamic planning with a LLM. URL: http://arxiv.org/abs/2308.06391, arXiv:2308.06391[cs], doi:10.48550/arXiv.2308.06391. Ver página 11.

[42] DeepSeek-r1/DeepSeek_r1.pdf at main · deepseek-ai/DeepSeek-r1. URL: https://github.com/deepseek-ai/DeepSeek-R1/blob/main/DeepSeek_R1.pdf. Ver página 11.

[43] Ehud Karpas, Omri Abend, Yonatan Belinkov, Barak Lenz, Opher Lieber, Nir Ratner, Yoav Shoham, Hofit Bata, Yoav Levine, Kevin Leyton-Brown, Dor Muhlgay, Noam Rozen, Erez Schwartz, Gal Shachaf, Shai Shalev-Shwartz, Amnon Shashua, and Moshe Tenenholtz. MRKL systems: A modular, neuro-symbolic architecture that combines large language models, external knowledge sources and discrete reasoning. URL: http://arxiv.org/abs/2205.00445, arXiv:2205.00445[cs], doi:10.48550/arXiv.2205.00445. Ver página 11.

[44] Yingqiang Ge, Wenyue Hua, Kai Mei, Jianchao Ji, Juntao Tan, Shuyuan Xu, Zelong Li, and Yongfeng Zhang. OpenAGI: When LLM meets domain experts. Ver página 11.

[45] Mingchen Zhuge, Haozhe Liu, Francesco Faccio, Dylan R. Ashley, Róbert Csordás, Anand Gopalakrishnan, Abdullah Hamdi, Hasan Abed Al Kader Hammoud, Vincent Herrmann, Kazuki Irie, Louis Kirsch, Bing Li, Guohao Li, Shuming Liu, Jinjie Mai, Piotr Piękos, Aditya Ramesh, Imanol Schlag, Weimin

Shi, Aleksandar Stanić, Wenyi Wang, Yuhui Wang, Mengmeng Xu, Deng-Ping Fan, Bernard Ghanem, and Jürgen Schmidhuber. Mindstorms in natural language-based societies of mind. URL: http://arxiv.org/abs/2305.17066, arXiv:2305.17066[cs], doi:10.48550/arXiv.2305.17066. Ver página 11.

[46] Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. Ver página 11.

[47] Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. ChatDev: Communicative agents for software development. URL: http://arxiv.org/abs/2307.07924, arXiv:2307.07924[cs], doi:10.48550/arXiv.2307.07924. Ver página 11.

[48] Sirui Hong, Mingchen Zhuge, Jiaqi Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. MetaGPT: Meta programming for a multi-agent collaborative framework. URL: http://arxiv.org/abs/2308.00352, arXiv:2308.00352[cs], doi:10.48550/arXiv.2308.00352. Ver página 11.

[49] Eirini Kalliamvakou. Research: quantifying GitHub copilot's impact on developer productivity and happiness. URL: https://github.blog/news-insights/research/research-quantifying-github-copilots-impact-on-developer-productivity-and-happiness/. Ver página 12.

[50] GitHub copilot features. URL: https://docs-internal.github.com/en/copilot/about-github-copilot/github-copilot-features. Ver página 12.

[51] sourcegraph/scip. original-date: 2022-05-10T13:18:47Z. URL: https://github.com/sourcegraph/scip. Ver página 12.

[52] sourcegraph/sourcegraph-public-snapshot: Code AI platform with code search & cody. URL: https://github.com/sourcegraph/sourcegraph-public-snapshot. Ver página 12.

[53] Kamal Acharya. Devin: A cautionary tale of the autonomous AI engineer. URL: https://medium.com/@lotussavy/devin-a-cautionary-tale-of-the-autonomous-ai-engineer-e1339ede8f8a. Ver página 12.

[54] Building a better repository map with tree sitter. URL: https://aider.chat/2023/10/22/repomap.html. Ver página 12.

[55] Number of parameters in GPT-4 (latest data). URL: https://explodingtopics.com/blog/gpt-parameters. Ver página 13.

[56] Baian Chen, Chang Shu, Ehsan Shareghi, Nigel Collier, Karthik Narasimhan, and Shunyu Yao. FireAct: Toward language agent fine-tuning. URL: http://arxiv.org/abs/2310.05915, arXiv:2310.05915[cs], doi:10.48550/arXiv.2310.05915. Ver página 13.

[57] Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. AgentTuning: Enabling generalized agent abilities for LLMs. URL: http://arxiv.org/abs/2310.12823, arXiv:2310.12823[cs], doi:10.48550/arXiv.2310.12823. Ver página 13.

[58] Zehui Chen, Kuikun Liu, Qiuchen Wang, Wenwei Zhang, Jiangning Liu, Dahua Lin, Kai Chen, and Feng Zhao. Agent-FLAN: Designing data and methods of effective agent tuning for large language models. URL: http://arxiv.org/abs/2403.12881, arXiv:2403.12881[cs], doi:10.48550/arXiv.2403.12881. Ver página 13.

[59] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. URL: http://arxiv.org/abs/2106.09685, arXiv:2106.09685[cs], doi:10.48550/arXiv.2106.09685. Ver página 14.

[60] C4 - abstractions. URL: https://c4model.com/abstractions. Ver página 31.

[61] Qinyu Luo, Yining Ye, Shihao Liang, Zhong Zhang, Yujia Qin, Yaxi Lu, Yesai Wu, Xin Cong, Yankai Lin, Yingli Zhang, Xiaoyin Che, Zhiyuan Liu, and Maosong Sun. RepoAgent: An LLM-powered open-source framework for repository-level code documentation generation. URL: http://arxiv.org/abs/2402.16667, arXiv:2402.16667[cs], doi:10.48550/arXiv.2402.16667. Ver página 33.