ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN. CIENCIAS

GRADO EN INGENIERÍA EN TECNOLOGÍAS Y SERVICIOS DE TELECOMUNICACIÓN

Área de Ingeniería Telemática

Análisis del estado del arte, diseño e integración de plataformas de IA generativa para enriquecer la información del software inventariado en herramientas de Gestión de Activos

González González del Rey, Pablo

Tutor: García Pañeda, Xicu Xabiel

Tutor de empresa: Castro Valdés, Alejandro

19 de julio de 2024

Resumen

La IA Generativa es una herramienta que en la actualidad ha supuesto una revolución tecnológica a todos los niveles. En el presente proyecto se plantea el uso de la IA Generativa como posible solución a alguno de los problemas relacionados con la gestión de activos software. En este trabajo se ha observado que el uso de una de estas IAs Generativas, ChatGPT, ha sido realmente útil para mejorar la información disponible en una herramienta de gestión de activos como es Proactivanet. Se ha conseguido automatizar este proceso de enriquecimiento de la información mediante un MVP basado en un diseño sin servidor sobre AWS (*Amazon Web Services*), un proveedor de servicios en la nube. Aunque los resultados han sido positivos, dado el avance imparable de la IA Generativa, parece lógico y fundamental plantear nuevas y más actuales soluciones, apoyadas en la IA Generativa, que mejoren la gestión de activos software.

Índice

Índice de figuras

Índice de tablas

Capítulo 1
Introducción

A pesar de que la Inteligencia Artificial (IA) Generativa parece una invención reciente, con la aparición de ChatGPT o Gemini, entre otros muchos ejemplos como Llama3, BERT o Claude, sus bases teóricas fueron sentadas por Alan Turing y John Von Neumann a mediados del siglo veinte. A medida que esta tecnología fue avanzando, se comenzaron a utilizar modelos probabilísticos y redes neuronales como principales métodos para el modelado de datos y patrones. Posteriormente, emergieron las máquinas de vectores de soporte como valiosas herramientas de reconocimiento de patrones. Sin embargo, el año 2014 ha sido el punto de inflexión gracias a la llegada de las Redes Generativas Antagónicas (GAN). [1]

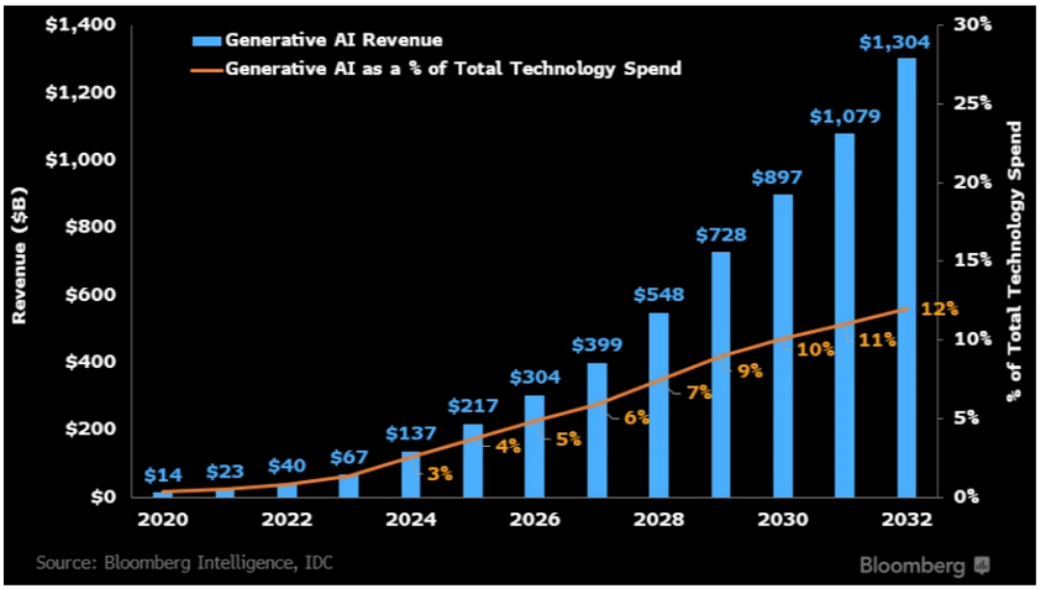

Figura 1.1: Ingresos IA Generativa y estimación para los próximos años. [2]

La IA Generativa representa lo que puede ser la siguiente revolución tecnológica, consiguiendo grandes avances en los últimos años (ver figura 1.1). Durante esta última década, la IA Generativa se ha expandido hacia diferentes sectores, con aplicaciones casi infinitas, teniendo un impacto significativo en múltiples industrias (ver figura 1.2). [3]

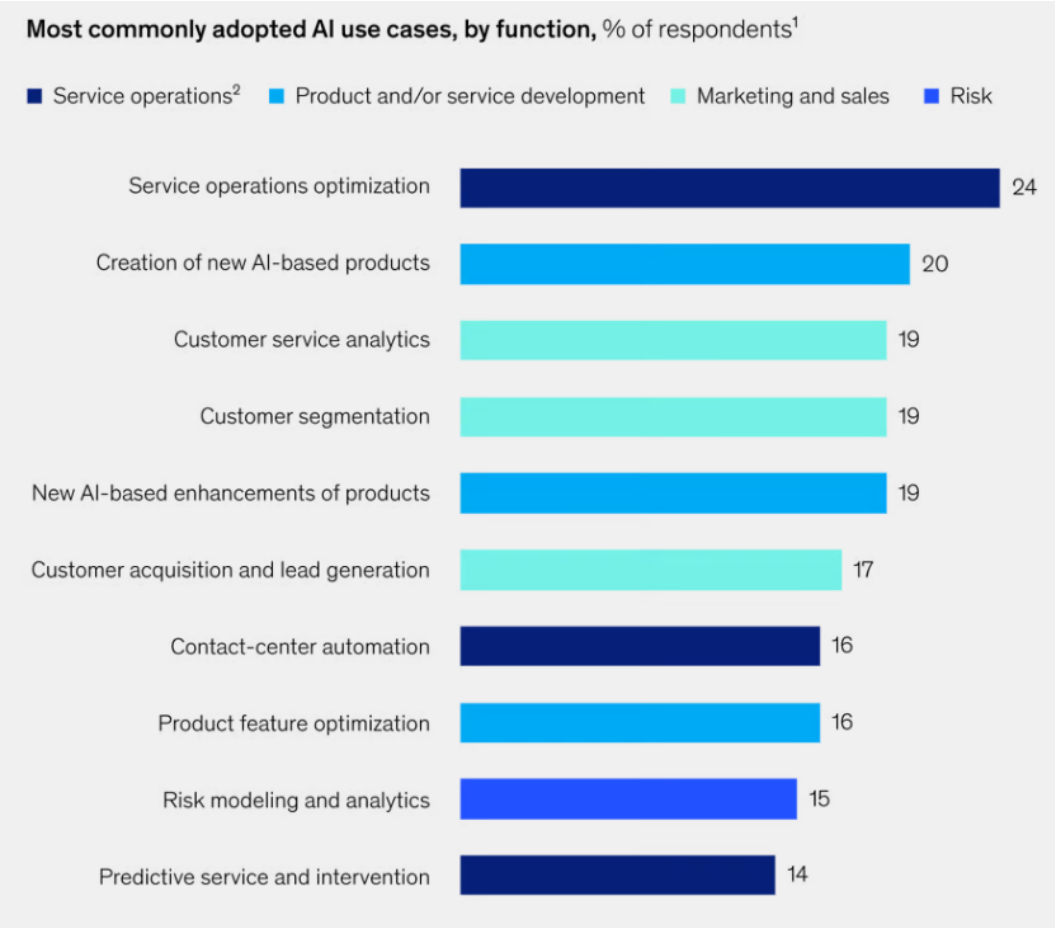

Figura 1.2: Usos más comunes para la IA Generativa. [4]

La IA Generativa es un tipo de IA capaz de crear nuevas muestras de datos similares a los datos ya existentes utilizando redes neuronales generativas, con el objetivo de encontrar patrones que, de cualquier otra forma, serían muy complicados de encontrar. Esto se completa, en gran medida, con el uso de las GAN, que permiten el desarrollo de un aprendizaje sin supervisión, donde la parte generadora es capaz de crear contenido nuevo y la discriminadora analiza su veracidad forzando a la parte generadora a realizar intentos hasta que la generadora acepte el resultado (ver figura 1.3), y los autocodificadores variacionales (VAEs), modelos generativos que aprenden representaciones comprimidas de sus datos de entrenamiento (espacios latentes), que utilizan para crear nuevos datos haciendo variaciones de las representaciones aprendidas (ver figura 1.4). [3, 5]

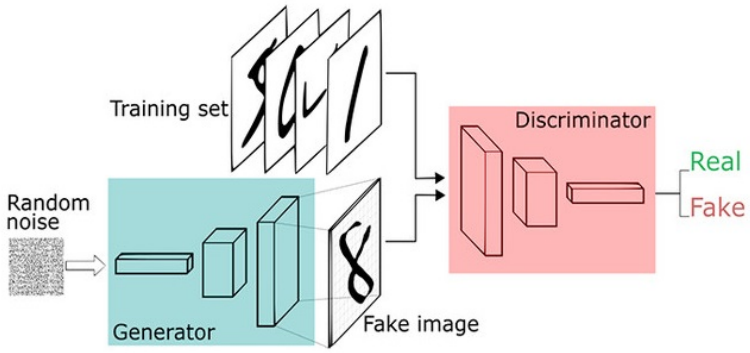

Figura 1.3: Esquema de funcionamiento de una GAN. [6]

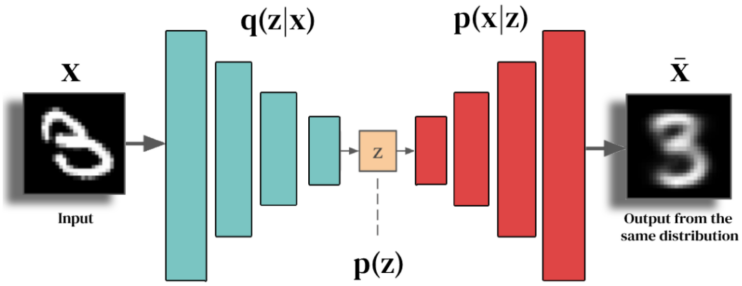

Figura 1.4: Esquema de funcionamiento de un VAE. [7]

El *Natural Language Processing* (NLP) es un campo de conocimiento encargado de investigar la forma de comunicar a las máquinas con las personas mediante el uso de lenguajes naturales. En sus inicios se utilizaban sistemas basados en reglas, donde lingüistas y científicos crearon a mano tanto reglas gramaticales como diccionarios para procesar el lenguaje natural. NLP utiliza algoritmos informáticos para procesar la forma de comunicación hablada o escrita, pudiendo realizar diversas tareas como la clasificación de temas, la detección de intenciones y la traducción de idiomas, a partir de la identificación de las raíces de las palabras. Los componentes básicos de un sistema de NLP eficaz incluyen [8]:

- Modelos de aprendizaje automático capaces de extraer el significado de los datos textuales.
- Modelos de aprendizaje profundo que procesan incrustaciones de palabras.
- Modelos lingüísticos que determinan las propiedades sintácticas de las lenguas.

Durante la década de los 80s, se avanzó hacia enfoques basados en datos, utilizando modelos de Markov ocultos, modelos estadísticos de traducción automática y las gramáticas probabilísticas libres de contexto. El siglo XXI marcó una nueva era caracterizada por el auge de las representaciones distribuidas, como Word2Vec o GloVe, y las redes neuronales, con arquitecturas como las Redes Neuronales Recurrentes (RNN) y las Redes de Memoria a Corto y Largo Plazo (LSTM). Sin embargo, fueron los modelos preentrenados como BERT, GPT o T5 los que revolucionaron la comprensión del lenguaje, basados en la idea de transferencia de aprendizaje, es decir, tomar un modelo entrenado para una tarea y aplicarlo a otra tarea que esté relacionada, mediante la extracción de características de los datos utilizando capas de redes neuronales ya entrenadas. [1]

Hoy en día se utilizan, mayoritariamente, lo que se conocen como *Large Language Models* (LLM), un tipo de modelos de aprendizaje profundo entrenados sobre inmensas cantidades de datos, que han representado un avance significativo tanto en el NLP como en la IA Generativa. Además, se construyen sobre un tipo específico de redes neuronales, denominadas modelos transformadores, que son capaces de aprender el contexto gracias a una técnica matemática denominada autoatención, la cual permite detectar formas sutiles en las que los elementos de una secuencia tienen relación. [9]

A lo largo de los años, los objetivos y finalidades para los que se han desarrollado modelos de inteligencia artificial han ido cambiando con la evolución de las tecnologías y las necesidades de las personas y organizaciones y, con el paso del tiempo, evolucionado hacia problemas más amplios y complejos (ver figura 1.5).

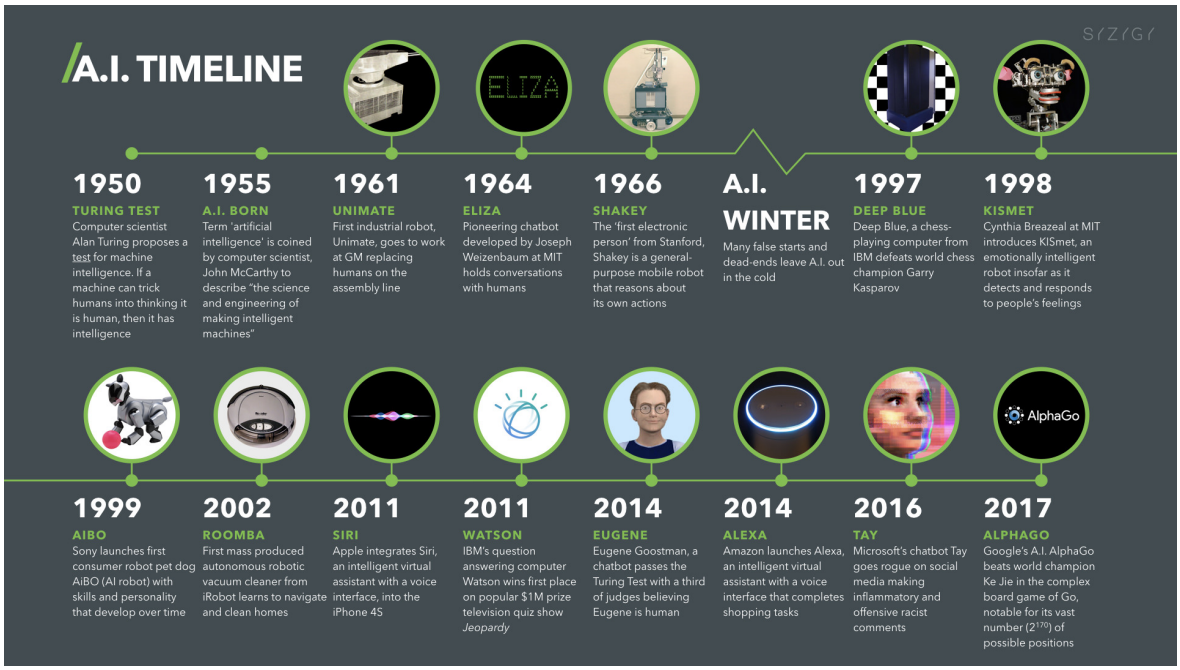

Figura 1.5: Evolución de los usos de la IA Generativa a lo largo de los últimos años. [10]

El *IT Asset Management* (ITAM) ha pasado a ser, en los últimos años, un aspecto de vital importancia dentro de las organizaciones. Se puede definir como el seguimiento y gestión integral de los activos informáticos de la organización, garantizando su correcta utilización, actualización, mantenimiento y eliminación durante su ciclo de vida, es decir, que todos los activos de IT se utilicen de forma eficiente y eficaz para permitir la correcta gestión de los costes, el riesgo y las oportunidades que pueden conllevar. [11]

Al tratarse de una tarea compleja de realizar, dada su amplitud y profundidad, han surgido diversos subconjuntos dentro de ITAM, incluyendo el *Hardware Asset Management* (HAM) y el *Software Asset Management* (SAM), siendo este último en el que se enfocará este trabajo (ver figura 1.6) [11].

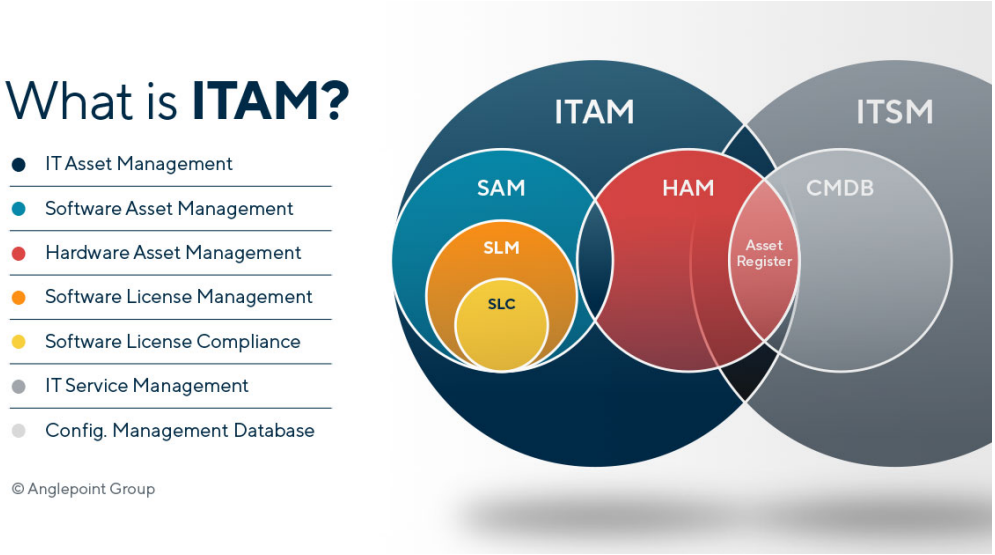

Figura 1.6: Relación entre ITAM, SAM y HAM. [12]

SAM nace para tratar específicamente con activos software, siendo una práctica empresarial que permite obtener a las organizaciones una visión general acerca de todo el software que se ha instalado, utilizado o licenciado. Esto implica la compra, implementación, mantenimiento, utilización y eliminación de software dentro de la organización. Su implementación proporciona una serie de beneficios de gran importancia como pueden ser [13]:

- **Ahorro económico:** ayuda a conocer todos los activos software de la organización permitiendo una mejor toma de decisiones en el momento de realizar adquisiciones, También facilita la identificación de licencias infrautilizadas o no utilizadas para su posterior reasignación o eliminación, ayudando a evitar sanciones por incumplimiento de contratos de nivel de servicio (SLA).
- **Mejor control de los riesgos:** permite identificar software desactualizado u obsoleto, así como incumplimientos y/o políticas inadecuadas de uso. Proporciona un mayor nivel de control al limitar la descarga de software no autorizado, utilizando políticas de creación y uso de credenciales de acceso.
- **Mejora de los procesos:** permite mejorar todos aquellos procesos que tengan relación con el software que haya sido instalado, garantizando el cumplimiento de las condiciones del contrato de licencia de software y proporcionando información detallada a los usuarios sobre cada programa utilizado permitiendo, por tanto, una labor más eficiente.

Para implementar SAM en una organización es necesario seguir un proceso que incluya los siguientes pasos [13]:

1. **Inventariar el software:** se realiza una auditoría de software para poder inventariar todos aquellos productos software que la organización tiene en posesión, tanto los que son utilizados como los que no (ver figura 1.7). Es más que recomendable utilizar plataformas que permitan centralizar esta información proveniente de distintas fuentes de inventariado.

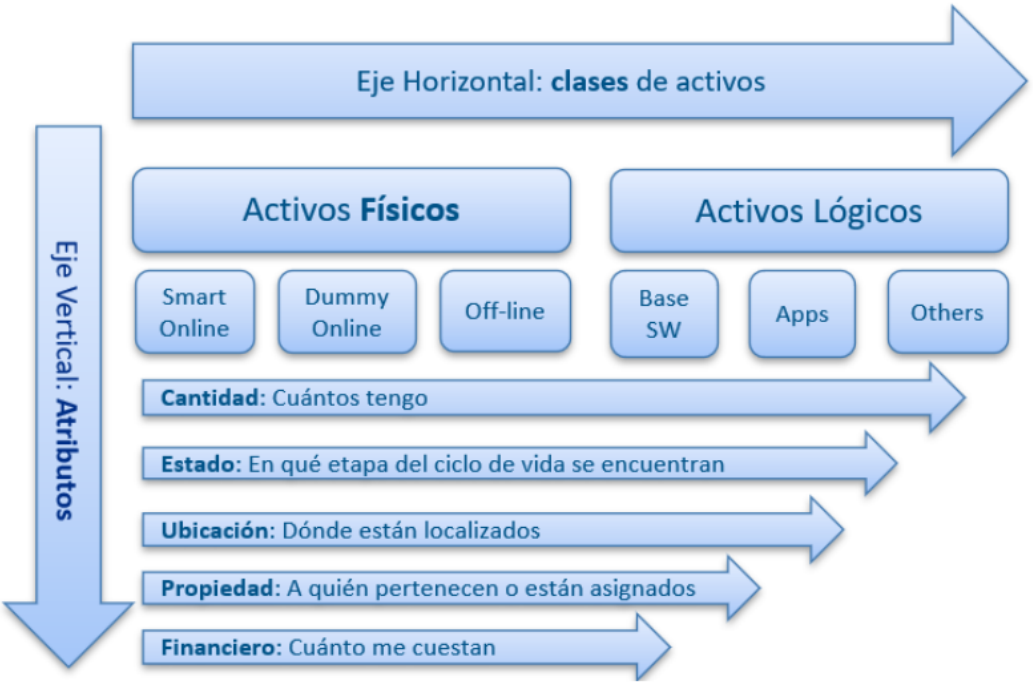

Figura 1.7: Forma de estudio de los activos software en dos ejes. [13]

2. **Recopilar las licencias y derechos de uso:** se recogen todos los contratos asociados a las licencias con derecho de uso de software que se hayan llevado a cabo para tener conocimiento de los permisos de la organización.
3. **Medir el uso real del software:** también es importante identificar y evaluar el uso real del software, de forma que se facilite la detección de las necesidades relacionadas y poder aplicar las acciones necesarias para satisfacerlas (ver figura 1.8).

Figura 1.8: Relación y problemas entre los softwares de una organización. [13]

4. **Regular el ciclo de vida del software:** el ciclo de vida del software comienza en el momento que aparece la necesidad de uso, pasando por la evaluación y decisión de compra, el despliegue, su uso y mantenimiento y, por último, su renovación o retirada. Para optimizar cada etapa es necesario que existan políticas que indiquen y regulen los pasos que se han de seguir en cada una de ellas.

Las diferentes herramientas que existen en el mercado para implementar ITAM y SAM, como pueden ser Proactivanet, ServiceNow o SolarWinds, facilitan enormemente seguir el proceso mencionado. Sin embargo, a la hora de realizar el inventariado, la información obtenida para todo el software que posee la organización, y que en ciertos casos puede ser de vital importancia, como por ejemplo las vulnerabilidades asociadas al software inventariado, puede quedarse corta o ser muy tediosa de obtener, más aún en la actualidad teniendo en cuenta la magnitud de los inventarios de activos de las organizaciones.

Proactivanet, mencionada en el párrafo anterior, es una herramienta innovadora creada para la gestión de activos y servicios TI (ITAM e ITSM), para la cual se destina la solución propuesta en este proyecto. Cuenta con diversos módulos, entre ellos Discovery & Gestión de Activos, Service Desk y CMDB, que permiten a la gerencia de TI obtener una notable mejoría en los procesos de gestión de la organización. Además, establece nuevos procesos que permiten aumentar la productividad, proporcionar automatización, mejorar el control, reducir riesgos y disminuir costes aportando valor. A nivel mundial, es la única herramienta de su clase que cuenta con trece procesos certificados por *ITIL (Information Technology Infrastructure Library) Software Scheme* en el máximo nivel Oro. [14]

Esta herramienta permite ahorrar costes evitando riesgos innecesarios, sin que la organización se vea afectada, y de forma permanente, a través de la detección de licencias infrautilizadas, con el control de la infraestructura *IaaS* (infraestructura como servicio), reduciendo los costes de soporte o evitando costes que deriven de fallos de seguridad y ciberataques. También garantiza el conocimiento del inventario en su totalidad detectando infraestructura TI obsoleta, equipos desactualizados o configuraciones potencialmente peligrosas. Cabe destacar también que con Proactivanet existe la posibilidad de ofrecer soporte remoto, habilitar nuevos métodos de contacto y agilizar los que están ya implementados, administrar equipos de forma remota o reducir las peticiones estándar con la ayuda del self-support. [14]

Para el ámbito concreto de este trabajo, Proactivanet permitirá automatizar en gran medida cada uno de los pasos necesarios a seguir para implementar SAM en una organización, desde el inventariado hasta la regulación del ciclo de vida de los activos software de la organización, gracias a los diversos módulos mencionados con los que cuenta actualmente.

Teniendo en cuenta todo lo anterior, podemos afirmar que las cualidades de la IA Generativa, tal y como la conocemos actualmente, podrían beneficiar y facilitar enormemente el proceso de inventariado el cual, en muchos casos, puede verse limitado por el tiempo que supondría conseguir la información con suficiente nivel de detalle para cada activo software. A través de preguntas concretas, podemos conseguir que una IA Generativa, véase ChatGPT, Llama o Gemini entre otras, sea capaz de responder con todos los datos necesarios para incrementar el nivel de detalle de la información almacenada de cada activo software.

La solución propuesta consiste en la realización de un diseño y arquitectura de un sistema de integración genérico entre las plataformas de IA Generativa y Proactivanet, poniendo especial atención en los mecanismos de comunicación basados en API Rest y la estructuración de la información intercambiada. Además, sería interesante concretar dicho diseño en un caso concreto de una de las plataformas analizadas, desarrollando un Mínimo Producto Viable (MVP) de un dispositivo software que añada información sobre los activos de tipo software inventariados en Proactivanet.

Esta integración final entre las plataformas mencionadas busca el enriquecimiento de la información disponible para cada uno de los activos software inventariados en Proactivanet, considerando, según su viabilidad, la información siguiente:

- Descripción.
- Clasificación del software.
- Tipo y métrica de licenciamiento.
- Fecha de lanzamiento.
- Fecha de fin de vida útil (EOL).
- Coste de licencia.
- Alternativas para dicho activo (con foco en alternativas gratuitas).
- Listado de los identificadores CPE relacionados.

1.1.- Objetivos y motivación

Actualmente las plataformas de gestión de activos software se ven limitadas, en tiempo y capacidad, a la hora de obtener suficiente información acerca de los activos software gestionados, más allá del nombre y versión del activo en cuestión, siendo necesaria, en muchos casos, la búsqueda manual de los datos más relevantes que el software no ha podido obtener de forma automática.

La falta de información relacionada con los activos obtenida durante la etapa de inventariado del software, puede derivar en numerosos problemas en las siguientes etapas de

la gestión de activos, desde usos fraudulentos de las licencias hasta la aparición de riesgos de seguridad debidos a activos software con vulnerabilidades relacionadas, pasando por costes innecesarios o ineficiencias debidas a un uso poco efectivo. Datos como son la clasificación, la descripción, o las alternativas facilitan la propia gestión de los activos, permitiendo comprender qué son, su útilidad y funcionamiento, y sus alternativas. Otros datos, como el tipo de licencia o las vulnerabilidades relacionadas con el activo pueden mejorar la seguridad de la organización y evitar posibles problemas derivados de malos usos de las licencias.

Por lo mencionado anteriormente, el objetivo principal del presente proyecto es demostrar la utilidad y beneficios que puede ofrecer la IA Generativa dentro de la gestión de activos, además de demostrar su posibilidad de implementación con las herrramientas de gestión de activos existentes actualmente en el mercado.

Para ello se han establecido los siguientes objetivos parciales:

- Realizar un estudio y análisis del estado actual en el que se encuentran las diferentes IAs Generativas que existen actualmente, poniendo el foco en su capacidad de integración con softwares/plataformas de gestión de activos, en este proyecto Proactivanet.
- Elaborar un posible diseño y arquitectura de un sistema genérico para posibilitar la integración entre Proactivanet y las plataformas de IA Generativa, poniendo especial atención en mecanismos que estén basados en API REST, además de la estructuración de la información intercambiada en el proceso en formatos como pueden ser JSON (*JavaScript Object Notation*) o XML (*Extensible Markup Language*).
- Desarrollar un Mínimo Producto Viable (MVP) de un artefacto software que consiga añadir más información sobre los activos de tipo software inventariados en Proactivanet.

Capítulo 2
Planificación

En este capítulo se detallará la estructura y fases del proyecto realizado, además de los costes de la realización del mismo. En la figura 2.1 se muestra la estructura general del proyecto y el orden seguido al lo largo de su desarrollo.

Fase del proyecto	**Descripción**
Enfoque inicial	Realización del análisis general de los objetivos del proyecto y los problemas que podrían surgir a lo largo de este para establecer el camino a seguir a lo largo del proyecto.
Evaluación de las alternativas	Evaluación y selección de la IA Generativa a utilizar para la obtención de la información enriquecida, en base a su disponibilidad, accesibilidad, coste y aspectos más relevantes en relación con el objetivo del proyecto.
Entrenamiento del modelo	Generación de los datos de entrenamiento y validación a partir de las categorías de activos software obtenidas directamente de Proactivanet y la creación de un modelo mejorado con *fine-tuning* para realizar la clasificación de software en base a dichas categorías como única tarea.
Desarrollo del MVP	El desarrollo involucra el diseño de la arquitectura, analizando qué elementos son necesarios para el desarrollo de la prueba de concepto y la implementación de un MVP basado en la arquitectura mencionada para conseguir la funcionalidad mínima que permita obtener resultados válidos.
Evaluación de los resultados	En esta fase se verificarán los resultados obtenidos en diferentes casos, tanto en términos de calidad como de tiempo de respuesta, y se realizará un análisis sobre el código desarrollado y posibles mejoras que puedan introducirse para mejorar los resultados.

Tabla 2.1: Descripción de las fases del proyecto.

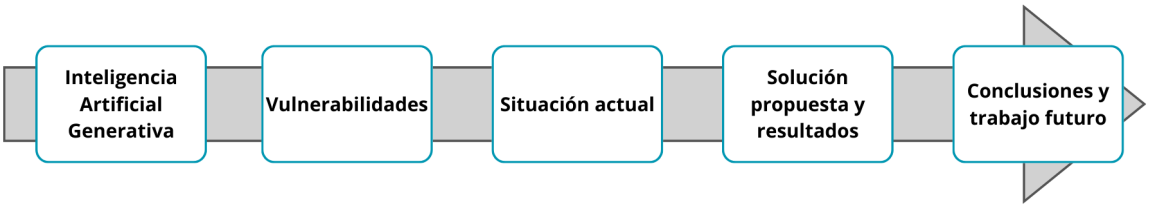

Figura 2.1: Estructura del proyecto.

2.1.- Presupuesto

Se ha realizado una estimación del presupuesto completo para este proyecto. Para el desarrollo del mismo se requiere de personal y equipamento software. No se ha requerido de equipamiento hardware adicional de ningún tipo. La duración del proyecto abarca desde el 1 de febrero hasta el 31 de junio de 2024.

Para realizar la estimación del presupuesto respectivo al personal se ha tenido en cuenta que el precio/mes se ha obtenido a partir de la ayuda percibida por el estudiante durante sus prácticas. A partir de los presupuestos totales expuestos en las tablas 2.2 y 2.3, se muestra el presupuesto total, que incluirá los gastos indirectos sobre el presupuesto total (15 %), el beneficio industrial y el Impuesto sobre el Valor Añadido (IVA del 21 %).

Concepto	**Coste Total**
Total personal	500,00 €
Total software	83,66 €
Total hardware	0,00 €
Total	**583,66 €**
Costes indirectos (15 %)	87,55 €
Beneficio industrial (6 %)	35,02 €
Total sin IVA	**706,23 €**
IVA (21 %)	148,31 €
Total con IVA	**854,54 €**

Categoría	**Coste**
Becario de investigación	100 € / mes
Total	**500 €**

Tabla 2.2: Coste estimado del personal.

Categoría	**Coste Total**
Coste uso modelos premium	100 $
Crédito adicional	25 $
Coste peticiones GPT 4 Turbo	11,26 $
Coste entrenamientos GPT 3.5 Turbo	5,77 $
Tasa de cambio	1 € = 1,10 $
Total	**83,66 €**

Tabla 2.3: Coste estimado software.

Capítulo 3
IA Generativa

En este capítulo, y antes de comenzar el análisis de las diferentes IAs Generativas, se explicará el concepto de IA Generativa en sí, además de describir brevemente su funcionamiento, para posteriormente analizar algunas las plataformas de IA Generativa escogidas, y finalmente explicar el proceso de selección.

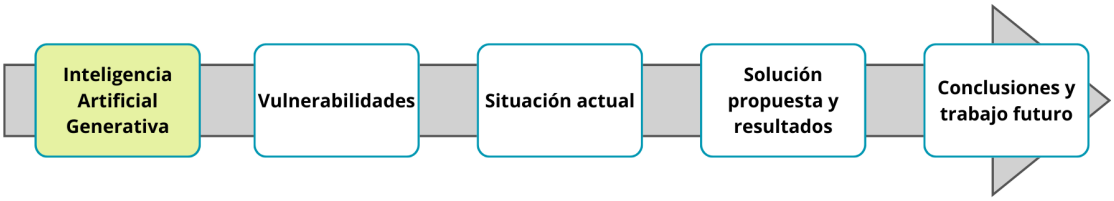

Figura 3.1: Estructura del proyecto.

El concepto de IA Generativa hace referencia a modelos de aprendizaje profundo que son capaces de generar texto de alta calidad, imágenes y otros tipos de contenido basándose en el conjunto de datos que se ha utilizado para su entrenamiento. Estos modelos pueden, a partir de datos en bruto, "aprender" a generar salidas estadísticamente probables cuando se les solicita. [15]

Antigüamente, los modelos de IA Generativa se ha utilizado durante años en estadística para el análisis de datos numéricos [15]. Sin embargo, lo que hoy se conoce como IA Generativa, ya sea ChatGPT, Gemini o similares, no es más que uno de los usos más comunes de los LLM, a los que se les proporciona una indicación o pregunta y son capaces de devolver una respuesta en forma de texto.

Los LLM son programas capaces de reconocer y generar texto, entrenados con enormes conjuntos de datos. Se trata de programas informáticos a los que se les han proporcionado suficientes ejemplos como para sean capaces de reconocer e interpretar el lenguaje

humano y otros tipos de datos. A través de este aprendizaje, son capaces de comprender el funcionamiento de los caracteres, palabras y frases en conjunto, e implica, además, el análisis probabilístico de datos no estructurados, lo que les permite reconocer distinciones entre los bloques de contenido sin necesidad de intervención humana. [16]

El aprendizaje profundo es un tipo de aprendizaje automático (práctica donde se proporciona a un programa gran cantidad de datos para su entrenamiento con el objetivo de que identifique patrones en los datos mencionados sin intervención humana) que utiliza la probabilidad para "aprender", haciendo capaces a estos modelos de reconocer patrones complejos, con el fin de generar información y predicciones precisas. Se contruyen sobre redes neuronales, modeladas a partir del cerebro humano, y están formadas por un gran número de capas de neuronas artificiales que trabajan en conjunto. Cada neurona artificial es un módulo software denominado nodo que utiliza diversos cálculos matemáticos para procesar los datos, y las redes neuronales artificiales son algoritmos de aprendizaje profundo que usan dichos nodos para la resolución de problemas. [17]

En el caso de los LLM, se utiliza un tipo específico de redes neuronales denominadas modelos de transformadores, introducidos por Google en 2017, que son capaces de aprender el contexto, algo especialmente importante para el lenguaje humano. Aplican un conjunto de técnicas matemáticas denominadas atención o atención propia para detectar la forma en la que los diferentes elementos de un conjunto de datos dependen entre ellos y como se influyen unos a otros. [18]

La mayoría de las redes neuronales se componen de grandes bloques de codificación y decodificación que procesan la información, donde el codificador procesa toda la secuencia datos de entrada y los transforma en una representación matemática compacta (extrae la idea principal de la entrada). Posteriormente, el decodificador generará paso a paso la secuencia de salida a partir de dicha representación. [18, 19]

Este proceso ocurre secuencialmente, procesando cada parte de los datos una tras otra. En cambio, los modelos de transformadores incluyen además el mecanismo de atención, que permite al modelo observar distintas partes de la secuencia al mismo tiempo y determinar cuales son las de mayor importancia. [19]

La arquitectura de los transformadores se compone de varias capas o etapas software que trabajan al unísono para generar el resultado final.

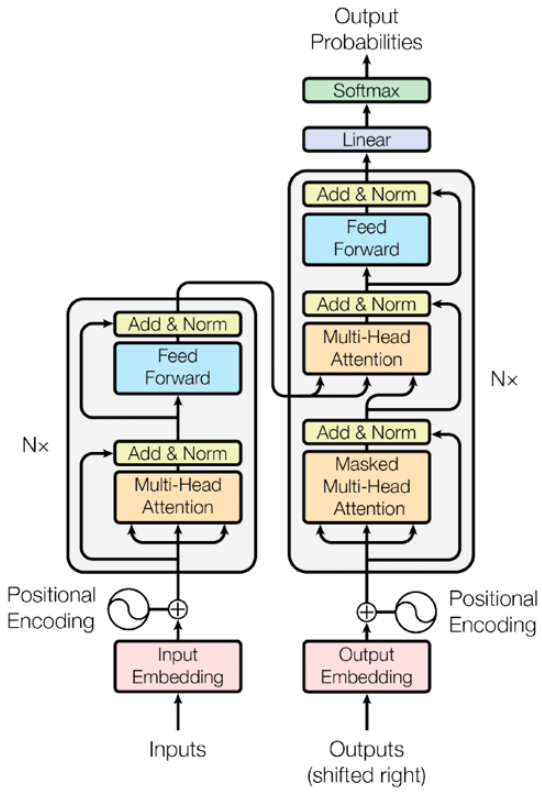

Figura 3.2: Esquema de los componentes de la arquitectura de los transformadores. [19]

En la etapa de entrada se divide la secuencia de datos en una seria de tokens (componentes de secuencia individuales) y posteriormente se tranforma cada token en una secuencia vectorial matemática, vectores que contienen la información semántica y sintáctica, en forma numérica, y cuyos atributos se aprenden durante el entrenamiento. [19]

La codificación posicional es un paso esencial, ya que permite al transformador conocer el orden de los tokens en la secuencia de entrada añadiendo información que indique su posición en la secuencia al final de cada uno. Esto permite al modelo preservar el orden de los símbolos y comprender el contexto. [19]

Un modelo de transformador se compone habitualmente de varios bloques de trans-

formadores apilados. Cada uno se compone a su vez de un mecanismo de atención con múltiples cabezales y una red neuronal de retroalimentación por posición. El mecanismo de atención es el que permitirá al modelo valorar la importancia de los tokens dentro de la secuencia, de forma que al hacer predicciones, se centre en las partes más importantes de los datos de entrada. [19]

Por último, el modelos necesita predecir de forma concreta, para lo cual se utiliza el bloque lineal, también conocido como capa densa. Es la parte que, a partir de las representaciones internas, las convierte en predicciones específicas que pueden interpretar y usar, y ofrece como resultado un conjunto de putuaciones para cada token posible. La función *softmax* toma las puntuaciones y las normaliza en una distribución de probabilidad. Cada elemento a su salida representa la confianza del modelo en un token concreto. [19]

Existen numerosos tipos de modelos de transformadores, sin embargo, los que se utilizan actualmente en los *chatbots* más conocidos, y que se valorarán para su posible utilización en este proyecto, son lo que se denominan modelos de transformadores generativos preentrenados (GPT). También existen otros tipos como son los transformadores bidireccionales, los transformadores bidireccionales y autorregresivos, los transformadores para tareas multimodales y los transformadores de visión. [19]

3.1.- Plataformas y modelos analizados

Para este proyecto se han analizado varias de las plataformas más conocidas y de mayor expansión actualmente, con especial enfoque en su rendimiento, calidad de las respuestas obtenidas y facilidad para la implementación, y que proporcionan acceso a diferentes LLMs generativas preentrenadas, en forma de *chatbots* accesibles tanto a través de Internet como haciendo uso de su API o APIs de terceros.

3.1.1.- OpenAI

Es una plataforma dedicada principalmente a la investigación y despliegue de inteligencia artificial. Fue fundada en 2015 con la misión de asegurar que la inteligencia artificial general beneficiase a todo el mundo. Su investigación abarca una amplia gama de temas, co-

mo pueden ser el procesamiento del lenguaje natural o la visión por ordenador y el desarrollo de modelos de inteligencia artificial como la serie GPT. [20]

Dispone de varios modelos con diferentes usos y entrenamientos. En este caso se valorarán los modelos de generación de texto, dados los objetivos de este proyecto. Sus modelos más recientes y actualizados cuentan con datos de entrenamiento muy recientes, hasta diciembre de 2023 en el caso de ChatGPT-4-Turbo. Es importante también tener en cuenta que ofrece la posibilidad de utilizar modelos con *fine-tuning*, y el procesode entrenamiento es relativamente sencillo que además está explicado en su documentación con ejemplos prácticos. [21]

Cuenta con API propia con gran diversidad de funcionalidades que da acceso a sus modelos desde otras aplicaciones. Dispone tanto de versión gratuita así como de varios *tiers* de pago, que permiten el uso de los mejores modelos y la utilización de la API (con ciertos límites dependiedo del *tier*) y permite el uso de sus modelos para fines comerciales (siguiendo siempre sus políticas de uso). [21]

También cabe destacar su documentación y referencias para el uso de la API con gran nivel de detalle y ejemplos prácticos para facilitar su entendimiento. Cabe mencionar también la posibilidad de obtener respuestas de la API tanto en lenguaje humano como en formato JSON, lo que puede facilitar enormemente su implementación en otras aplicaciones. [21]

3.1.2.- Meta

Es una plataforma dedicada a multitud de temas, comenzó su inversión en inteligencia artificial en 2022, en un principio para mejorar el rendimiento de sus anuncios, pero que evolucionó posteriormente hacia la inteligencia de código abierto, con el desarrollo de la serie de modelos Llama (ahora Llama2 y Llama3) y el objetivo de mantenerse competitiva frente a Google o Microsoft. [22]

Ofrece diversos modelos actualmente, aunque al momento de realizar el análisis comparativo entre las diferentes IAs Generativas no estaba disponible el modelo Llama3,

por lo que se analizó la familia de modelos Llama2 como posibilidad para utilizar. Los datos de entrenamiento utilizados son bastante recientes, entre enero y julio de 2023 para Llama2. También ofrece la posibilidad de utilizar modelos con *fine-tuning*, sin embargo, el proceso de entrenamiento es complejo. [23, 24]

No cuenta con API propia, sin embargo existen numerosas alternativas creadas por usuarios. El uso de sus modelos es gratuito y, en algunos casos, están disponibles para utilizar en Amazon Web Services. Su documentación no es demasiado amplia y no cuenta con muchos ejemplos prácticos.

3.1.3.- Google Research

Esta plataforma es una rama de investigación y desarrollo de inteligencia artificial de Google, enfocada hacia proyectos de investigación que promuevan el desarrollo y avance de la inteligencia artificial y sus funciones, de forma que sean aprovechables por los productos de Google. Un ejemplo de sus avances es la serie de modelos Gemini. [25]

Al igual que las plataformas anteriores, ofrece múltiples modelos. Sus modelos más recientes y actualizados cuentan con información muy reciente, hasta noviembre de 2023, e incluso ofrece la posibilidad de realizar búsquedas a través de Internet, aunque esto pueda suponer un riesgo para la calidad de la información de la respuesta. [26]

Cuenta tanto con versión gratuita como con versión de pago (por uso), diferenciados principalmente por el límite de uso de ambas opciones. Ofrece además la posibilidad de utilizar modelos con *fine-tuning*, sin embargo, el proceso de entrenamiento es medianamente sencillo pero no se dispone de ejemplos prácticos en la documentación. [27]

Cuenta tanto con API propia como con API de terceros, aunque en ambos casos son bastante recientes. Además, algunos modelos permiten pasar en la petición el esquema JSON específico requerido en la respuesta. La documentación es breve y cuenta con pocos ejemplos prácticos de uso. [28, 29]

3.1.4.- IBM

Es una plataforma dedicada, al igual que Meta, a múltiples temas, entre ellos al desarrollo de soluciones de inteligencia artificial que mejoren y aporten al crecimiento de las organizaciones. Entre ellas se incluye la serie de modelos Watson. [30]

Los modelos Watson ofrecen diversas opciones para la implementación e integración de la IA dentro de la organización. Está diseñado para ayudar en la toma de decisiones mediante el análisis de grandes cantidades de datos. Además, ha sido entrenado con gran cantidad de datos incluyendo registros médicos, informes financieros e investigaciones académicas que lo hacen adecuado para aplicaciones en campos relacionados con dichos datos. [31]

Cuenta con API propia, además de tener tanto versión gratuita como versión de pago. Se enfoca principalmente hacia el desarrollo de *chatbots* personalizados para otras aplicaciones. Cuenta también con la posibilidad de utilizar modelos con *fine-tuning*, pero el entrenamiento es algo complejo.Además, la documentación y referencias de la API son extensas y diferenciadas para cada uno de los modelos (*Assistant, Speech To Text, Text to Speech*, etc.). [32]

3.1.5.- Anthropic

Es una plataforma creada para el desarrollo de sistemas de inteligencia artificial a gran escala para la creación de modelos más seguros y fiables. Sus equipos de investigación tienen la misión de investigar acerca de la seguridad, funcionamiento interno e impacto de los modelos de inteligencia artificial, de forma que su impacto sea positivo a medida que estos avancen. Entre ellas se incluye la serie de modelos Claude. [33]

Los modelos Claude son fiable y de alto rendimiento, destacando en diversas tareas, como son la generación y mejora de código, la categorización y resumen, o la clasificación de incidencias. Tiene la posibilidad de uso gratuito y de pago en dos *tiers* diferenciados. [33,34]

Cuenta también con API propia y varios de sus modelos son accesibles a través de

Amazon Bedrock. Su documentación es sencilla de utilizar y amplia, además de que cuenta también con referencias de la API. El *fine-tuning* está disponible en los modelos de AWS, y el proceso de entrenamiento es sencillo y la documentación cuenta con ejemplos prácticos. [34]

3.2.- Criterios de selección

Para seleccionar adecuadamente la plataforma de IA Generativa a utilizar, es necesario tener en cuenta varios aspectos que pueden llegar a ser determinantes para conseguir un correcto desempeño a la hora de obtener la información sobre cada activo software.

En primer lugar, el desempeño de los diferentes modelos de cada plataforma, es decir, que la información procedente del modelo sea lo más precisa y reciente posible. La información desactualizada o imprecisa puede dar lugar a confusiones o problemas más graves en la organización, por lo que este es uno de los criterios clave para decidir que plataforma utilizar.

En segundo lugar, la facilidad de integración. En este caso concreto, es necesario obtener la información enriquecida del software a través de la API respectiva de cada modelo. Para poder llevar a cabo esta tarea, evitando cualquier inconveniente en la medida de lo posible, es imprescindible la alta disponibilidad y calidad de las APIs, además de la posibilidad de obtener la respuesta en formato JSON (*JavaScript Object Notation*) consiguiendo un programa más sencillo de nuestro lado.

En tercer lugar, el costo y licenciamiento. Obtener la información para cada uno de los activos software puede llegar a ser muy costoso, más aún en una organización donde su número puede ser muy elevado. Por ello, evaluar y comparar los costes de las diferentes opciones disponibles en el mercado es otro punto que se ha de tener muy en cuenta en el momento de escoger el modelo. También es importante tener en cuenta el tipo de licenciamiento del modelo, dado que el posible producto final que puede derivar de este MVP se va a utilizar con fines lucrativos, es posible que cambie el costo por la utilización del modelo, o que incluso no se permita su uso con este tipo de fin.

En cuarto lugar, quizás no imprescindible, pero no por ello menos importante, la documentación y soportes disponibles. Para realizar la implementación y explotar de manera eficaz el modelo elegido, es beneficioso poder contar con una documentación amplia y sencilla, además de un equipo de soporte que pueda resolver problemas que puedan surgir en los modelos que se utilizarán y que escapan a nuestro alcance.

Por último, la flexibilidad y personalización de los modelos. En el caso concreto de la categorización del activo software en base a las categorías ya existentes en Proactivanet, es necesario disponer de un modelo capaz de, como mínimo, realizar dicha tarea de manera consistente. El *fine-tuning* permite entrenar un modelo para luego responder de la forma más parecida posible si recibe preguntas similares a las de los ejemplos.

3.3.- Análisis comparativo de las plataformas

Para este proyecto se han analizado diversas plataformas de IA Generativa en base a los criterios mencionados en el punto anterior, entre ellas ChatGPT (OpenAI), Llama2 (Meta), Gemini (Google), Watson (IBM) o Claude (Anthropic).

Actualmente para evaluar de forma precisa la IA Generativa existen numerosos *benchmarks*, herramientas y métodos que se emplean para medir el redimiento de los modelos (entre ellos los modelos de generación de texto). Son una parte esencial para comprender la eficacia, precisión y creatividad de estos sistemas. [35] Para el caso concreto de los LLM, existen múltiples *benchmarks*, algunos de ellos son:

- MMLU: es una prueba que permite medir la precisión multitarea de un modelo. Abarca 57 tareas, desde matematicas elementales hasta historia o informática. Una alta puntuación en esta prueba requiere que los modelos posean amplios conocimientos y capacidad para resolver problemas adecuadamente. [35]
- Human Evaluation: es una evaluación subjetiva realizada por personas en términos de coherencia, precisión y creatividad en las respuestas de los modelos. [35]
- MT-Bench: es un conjunto de pruebas para LLMs para comprobar su capacidad de

mantener una conversación natural y significativa. Además, estas pruebas están diseñadas para suponer un reto al modelo, por lo que una puntuación alta significa que un LLM es realmente bueno entendiendo las preguntas y respondiendo adecuadamente. [36]

Todos los modelos que se analizan en este proyecto obtienen puntuaciones altas en estos y otros *benchmarks*, además de tener de las mejores puntuaciones en diferentes comparativas entre los modelos actuales. Las figuras 3.3 y 3.4 muestran los resultados que obtienen diferentes LLMs obtenidos de *Chatbot Arena* [37], una plataforma abierta para la evaluación de LLMs por preferencia humana. Cuanto mayor sea el resultado, mejor será el modelo.

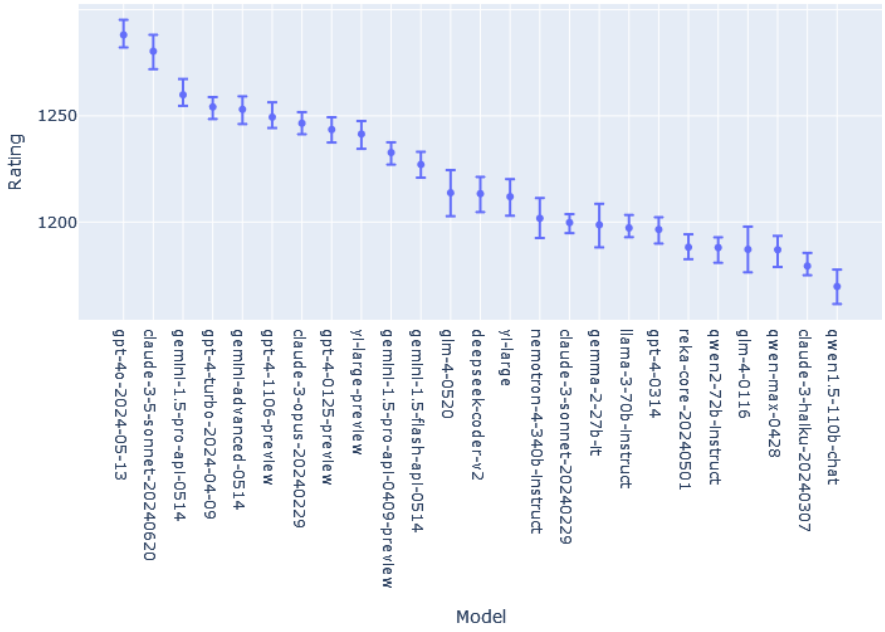

Figura 3.3: Intervalos de confianza basados en la fuerza del modelo. [37]

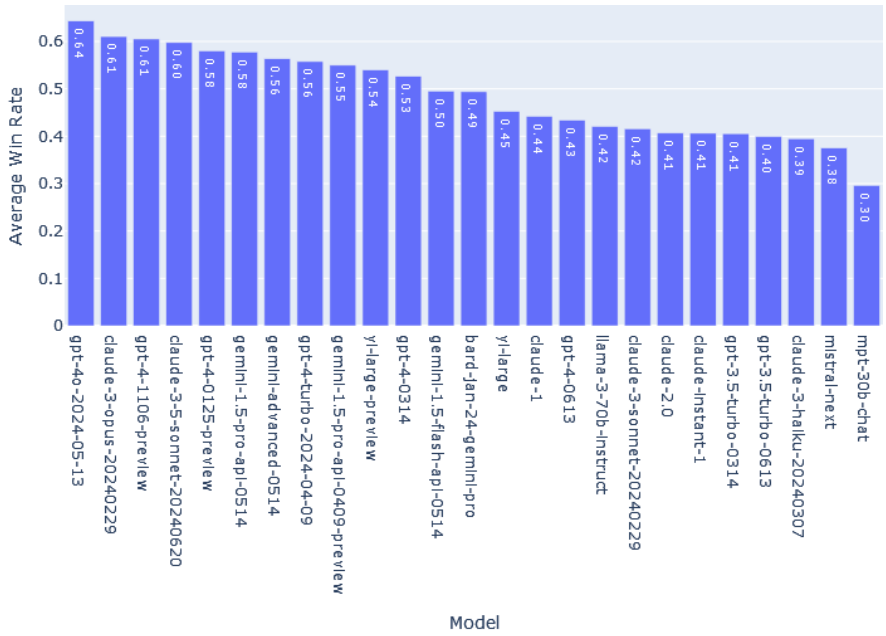

Figura 3.4: Media de victorias ante el resto de modelos. [37]

A pesar de que Llama2 es bastante completo, si es cierto se ve superada con creces en términos de número de parámetros utilizados para su entrenamiento, los modelos más grandes de Llama2 tienen en torno a 70 mil millones de parámetros, mientras que modelos como ChatGPT-4 se estima que tienen en torno al billón. Cabe mencionar también que Llama2 no cuenta con su propia API dedicada, lo que puede dificultar su utilización dentro del ámbito de este proyecto.

Gemini, aún siendo un modelo muy completo, está más enfocado en la publicidad digital, la documentación disponible es más limitada que en otros casos, y la versión estable de la API ofrecida por Google actualmente es muy reciente. Existen también otras plataformas como VertexAI que ofrecen APIs para trabajar con Gemini. Cabe mencionar también que las tareas de finne-tuning pueden ser más complejas para Gemini en comparación con otros modelos.

A pesar de que Watson es muy preciso en el análisis de datos estructurados, puede tener dificultades con datos no estructurados, como puede ser preguntas más generales acerca de una activos software en el caso de este proyecto.

Aunque Claude es de los modelos más completos y potentes actualmente, está diseñado para sobresalir en tareas concretas, como son la moderación de contenido, tareas clasificación o el resumen de textos. Teniendo esto en cuenta, además de no tener disponible la opción de fine-tuning, la implementación de Claude puede ser más compleja que en otros casos.

A partir de este análisis, y teniendo en cuenta las diferentes ventajas e inconvenientes planteados para cada modelo analizado, se ha llegado a la conclusión de que, en conjunto, la plataforma más completa de las analizadas es ChatGPT ofrecida por OpenAI. La elección se basa en las siguientes razones, acordes con los criterios de selección previamente mencionados:

- **Desempeño:** los modelos de ChatGPT que se utilizarán para la posterior elaboración del MVP (GPT 4 Turbo) cuentan con datos de entrenamiento muy actualizados, hasta diciembre de 2023 (de los más recientes de los modelos de las diferentes plataformas del mercado). No se tienen en cuenta los modelos con capacidad de realizar búsquedas por Internet para completar las respuestas, dado que la información puede llegar a ser muy imprecisa o incluso falsa. Además, en muchos casos, esta opción todavía no está disponible para las peticiones realizadas a través de la API de la plataforma.
- **Integración:** ChatGPT ofrece diversos *tiers* de uso que le permiten ajustarse a las necesidades de la organización. También se ha de tener en cuenta la rapidez de las respuestas de los modelos, incluso en casos de alta complejidad, y la posibilidad de obtener las respuestas en formato JSON, facilitando la interpretación de los datos recibidos de cada activo software.
- **Costo y licenciamiento:** nuevamente, al igual que para la integración, los *tiers* de uso ofrecen gran flexibilidad en el costo, pudiendo adaptarse a las necesidades de la organización, incluso pudiendo limitar el gasto de la propia API. De igual forma, se permite utilizar los modelos y la API para fines lucrativos gracias a los diferentes planes ofrecidos por la plataforma.
- **Documentación y soporte:** la documentación ofrecida por la plataforma de ChatGPT es amplia, concisa y sencilla, facilitando enormemente la comprensión y utilización de las diferentes funcionalidades de la API, además de contar con un servicio de soporte con gran diversidad de artículos que permiten resolver las dudas que puedan surgir

durante la implementación o el uso de la plataforma.

- **Flexibilidad y personalización:** ChatGPT tiene disponible una funcionalidad de *fine-tuning*, que permite entrenar uno de sus modelos con datos, tanto de entrenamiento como de validación, en forma de pares pregunta-respuesta. Esto permitirá a dicho modelo entrenado clasificar un activo software en base a las categorías que existen ya en Proactivanet.

3.4.- Metodología para la formulación de las preguntas al modelo

Los modelos de generación de texto que han sido entrenados para comprender el lenguaje natural son capaces de ofrecer salidas de texto en respuesta a las entradas que reciben denominadas *prompts*. El diseño de cada *prompt* es, en esencia, la forma de "programar" estos modelos, normalmente utilizando instrucciones o ejemplos para completar la tarea. [21]

Los modelos de chat, que se utilizarán en este proyecto, reciben una lista de mensajes como entrada y devuelven la respuesta generada como salida, además del modelo concreto al que se le enviarán dicha lista de mensajes. La lista mencionada será el parámetro principal enviado y contendrá diferentes mensajes, cada uno con un rol determinado (sistema, usuario o asistente) y un contenido. Se suele comenzar con un mensaje de sistema, que ayudará a establecer el comportamiento que tendrá el modelo a la hora de procesar el mensaje, seguido de mensajes de usuario, que proporcionan las peticiones que el asistente deberá responder, y asistente, que guardarán las respuestas.

Para obtener resultados fácilmente interpretables, la API cuenta con la posibilidad de indicar a los modelos, a través de un parámetro, que sus respuestas deberán redactarse en formato JSON. En el caso de utilizar esta funcionalidad, será necesario indicar en el mensaje de usuario, y de forma explícita, que se requiere una respuesta en dicho formato.

Por defecto, las respuestas ofrecidas por los diferentes modelos son, hasta cierto punto, aleatorias. En el caso de este proyecto, es necesario poder obtener respuestas deterministas, es decir, que a pesar de realizar la misma petición varias veces en distintitos momentos, la respuesta no varíe, en la medida de lo posible, como puede ser el caso al des-

cribir un activo software. Para ello, se ofrece cierto control sobre las salidas deterministas, pudiendo elegir la semilla utilizada para generar la respuesta, además de poder variar el valor de la temperatura (valor entre 0 y 2 que determina el grado de aleatoriedad de la respuesta).

```
{
    "model": "gpt-3.5-turbo",
    "messages": [
        {"role": "system", "content": "You are a helpful assistant."},
        {"role": "user", "content": "Who won the world series in 2020?"}
    ],
    "temperature": 0.7
}
```

Código 1: Cuerpo de la petición realizada a la API de ChatGPT (modelo 3.5 Turbo). [21]

3.5.- *Fine-tuning* de un modelo de ChatGPT

El *fine-tuning* es una funcionalidad que permite sacar más partido a los modelos disponibles y mejorar los modelos en tareas específicas proporcionando resultados de mejor calidad, la posibilidad de entrenar con más ejemplos de los que podrían incluirse en un solo mensaje a un modelo a través de la API, el ahorro de *tokens* y menor latencia en las peticiones.

Los datos de entrenamiento consisten en una lista de ejemplos (2) muy similares a las conversaciones que se enviarán al modelo en cuestión. Cada uno de los ejemplos tendrá el mismo formato que el mencionado en el punto anterior, especificando una lista de mensajes. Cada ejemplo tendrá la forma de una conversación con el modelo. Los mensajes con rol de usuario seguirán siendo las peticiones realizadas, sin embargo, los mensajes con rol de asistente contendrán la respuesta que debería ofrecer el modelo al recibir la petición que contiene el mensaje previo de usuario.

```
{"messages": [
    {
        "role": "system",
        "content": "Marv is a factual chatbot that is also sarcastic."
    },
    {
        "role": "user",
        "content": "What's the capital of France?"
    },
    {
        "role": "assistant",
        "content": "Paris, as if everyone doesn't know that already."
    }
]}
```

Código 2: Ejemplo de conjunto de mensajes para *fine-tuning* de un modelo de ChatGPT. [21]

Es necesario que se aporten como mínimo 10 ejemplos, aunque se recomienda aportar en torno a 50 para que el modelo muestre signos de mejora. También se aconseja que parte de estos datos se separe y utilice como datos de validación.

Capítulo 4
Vulnerabilidades

Este capítulo se centrará en la definición y análisis de las vulnerabilidades y su obtención para cada activo software. Esta información es de vital importancia, más aún en el ámbito empresarial, donde la seguridad es un aspecto extremadamente importante a tener en cuenta.

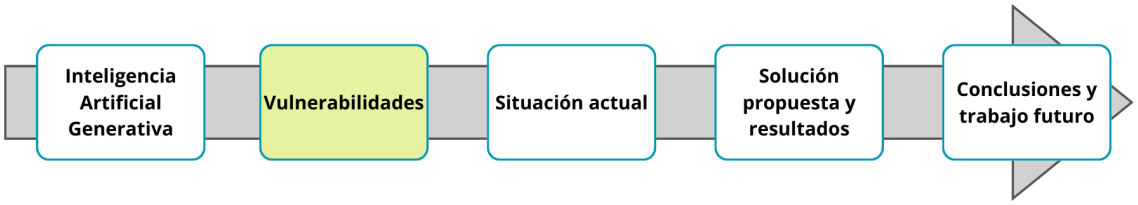

Figura 4.1: Estructura del proyecto.

El sentido más amplio de la palabra, vulnerabilidad se asocia con la violación de una política de seguridad. Esto puede ser debido a la debilidad de las reglas de seguridad o directamente a problemas con el propio software. En general, todo sistema informático tiene vulnerabilidad, aunque su gravedad depende directamente de su posible uso para causar daños en el sistema. [38]

La gestión de vulnerabilidades es un proceso que permite identificar y gestionar las vulnerabilidades de una red o sistema informático con el objetivo de reducir al mínimo el riesgo de las amenazas de seguridad identificando, analizando, evaluando y respondiendo adecuadamente a cada vulnerabilidad.

En las organizaciones, el descubrimiento de las vulnerabilidades es un proceso centrado en la comprobación de todos los activos de TI conocidos con el objetivo de detectar vulnerabilidades conocidas y potenciales. Teniendo esto en cuenta, obtener información sobre las diferentes vulnerabilidades conocidas de forma automatizada puede facilitar enormemente este proceso.

4.1.- *Common Platform Enumeration*

El *Common Platform Enumeration* (CPE) es un esquema de nombramiento estructurado dirigido a sistemas de TI, software y paquetes. Esta basado en la sintaxis genérica de los Identificadores de Recursos Uniformes (URIs), aunque incluye adicionalmente un formato formal de nombre, un método para comprobar los nombres en un sistema y un formato de descripción que permita vincular texto y pruebas a un nombre. La versión actual del CPE es la 2.3, y se define mediante un conjunto de especificaciones en un modelo basado en pila [39]. Los CPE pueden encontrarse en formato URI o en *Well-Formed CPE Name* (WFN). Cada nombre CPE contendrá:

- ***CPE version***: indica la versión de CPE actual.
- ***Part***: indica el tipo de entidad con la que se relaciona, y tendrá únicamente los siguientes valores: .ª"para aplicaciones, .º"para sistemas operativos y "h"para dispositivos hardware.
- ***Vendor***: describe o identifica a la persona u organización que ha manufacturado o creado el producto.
- ***Product***: describe o identifica el título o nombre más común o reconocible del producto.
- ***Version***: cadena alfanumérica que indique la versión de lanzamiento del producto.
- ***Update***: cadena alfanumérica que identifique la actualización particular, paquete de servicio o versión menor del producto.
- ***Edition***: indica la edición específica del producto. En la versión 2.3 está deprecado.
- ***Language***: indica, si es aplicable, el idioma del producto.
- ***Software edition***: edición específica del software.
- ***Target software***: software objetivo para el que está diseñado el producto.
- ***Target hardware***: hardware objetivo para el que está diseñado el producto.
- ***Other***: información adicional que no está incluida en los atributos ya mencionados.

Además, se utilizan ciertos caracteres, como por ejemplo “*” como caracter co-

modín, es decir, que pueden sustituir a cualquiera de un subconjunto definido de todos los caracteres posibles, o "-" para definir un atributo como no aplicable (NA). En los siguientes ejemplos se representa primeramente a Windows 7 Service pack 2 y a continuación la representación de Internet Explorer 8.0.6001 Beta.

```
cpe:2.3:o:microsoft:windows_7:-:sp2:*:*:*:*:*:*:*
cpe:2.3:a:microsoft:internet_explorer:8.0.6001:beta:*:*:*:*:*:*:*
```

Código 3: Ejemplos de nombres CPE. [39]

4.2.- *National Vulnerability Database*

El *National Vulnerability Database* (NVD) es un recurso fundamental de ciberseguridad que proporciona información detallada acerca de las vulnerabilidades en una amplia gama de software y hardware. Está mantenida por el Instituto Nacional de Estándares y Tecnología (NIST) y sirve como repositorio de datos de gestión de vulnerabilidades basados en estándares. Cataloga las vulnerabilidades basándose en el estándar de nomenclatura de *Common Vulnerabilities and Exposures* (CVE). [40]

Cada CVE contiene información importante como descripciones, puntuaciones de severidad y referencias a avisos y soluciones relacionados. Cada uno incluye además una lista con todos los softwares que se ven afectados por la vulnerabilidad en cuestión. Dicha lista viene en forma de URIs de CPE, lo que facilita la búsqueda de los CVEs relacionados con cada producto software utilizando su CPE asociado.

Gracias a la API que el NIST ofrece para realizar búsquedas en el NVD, es posible obtener de manera sencilla los CPE de cada activo software.

Capítulo 5
Situación actual

En este capítulo se describirá el software Proactivanet y su relación con SAM (concretamente el módulo Discovery & Gestión de Activos y Proveedores), además de la relevancia de este proyecto para la mejora en el proceso de descubrimiento y gestión de activos software.

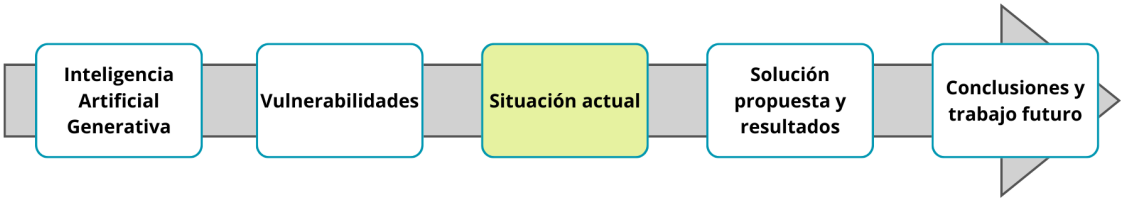

Figura 5.1: Estructura del proyecto.

Proactivanet Discovery & Gestión de Activos y Proveedores [41] permite conocer de forma inmediata y de manera exhaustiva el inventario del parque informático al completo, incluyendo sus licencias de software y configuración de forma totalmente automática y desatendida. También permite el seguimiento y el control de los proveedores y contratos, tanto para la compra de software y hardware, como para la contratación de servicio.

Actualmente, Proactivanet cuenta con gestión de activos software (SAM) y hardware (HAM) separados y, en este caso, el proyecto se centrará en el primero. Dentro del apartado de software, contenido en el panel de activos, se distinguen varias categorías de software aunque, dada la finalidad de este proyecto, se prestará atención únicamente a la categoría de aplicaciones. Cada entrada correspondiente a una aplicación dentro de Proactivanet (ver figura 5.2) contiene, además del nombre, la versión, el distribuidor/creador y la plataforma en la que se utiliza dicha aplicación (ver figura 5.3). Sin embargo, esta información disponible para los usuarios puede ser limitada, sobre todo si se carece de conocimiento acerca de la aplicación en cuestión.

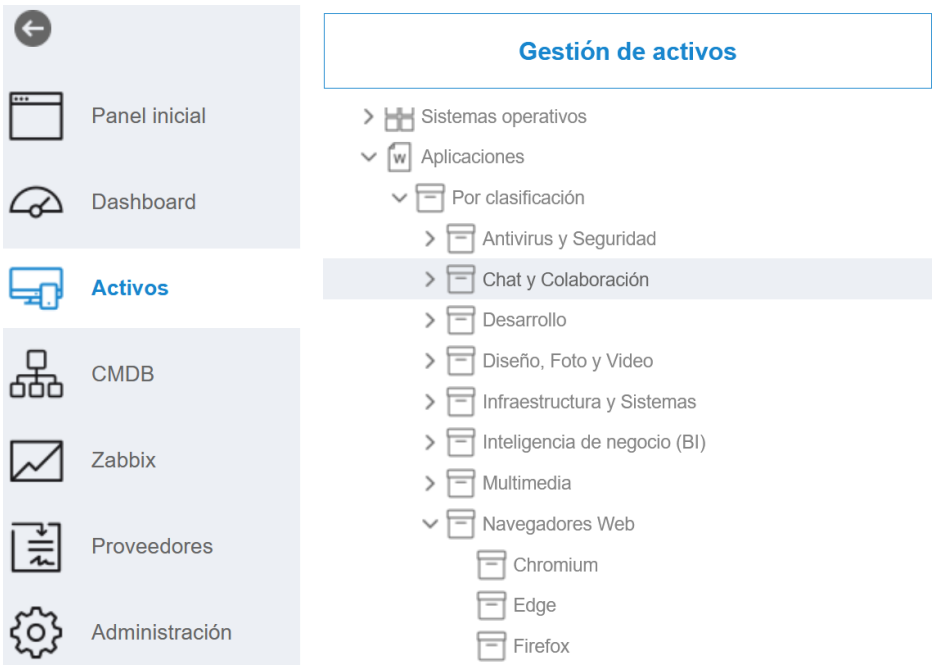

Figura 5.2: Clasificación de activos actual encontrada en Proactivanet. [41]

Software	Versión	Publisher
Skype versión 8.89	8.89	Skype Technologies S.A.

Figura 5.3: Entrada correspondiente a un activo software en Proactivanet. [41]

Proactivanet cuenta también con un sistema básico de clasificación de aplicaciones que, a través de reglas sencillas (generalmente expresiones regulares) aplicadas al nombre de la aplicación en cuestión, es capaz de clasificar adecuadamente cada una de ellas. Sin embargo, aquellos casos en los que el nombre no se corresponde adecuadamente con las reglas de clasificación pueden dar lugar a fallos en este sistema de clasificación.

Tanto la obtención de la información como la clasificación de los activos software puede llegar a ser una tarea tediosa y costosa, más aún en organizaciones de gran tamaño donde la cantidad de información a obtener es extremadamente extensa. En este caso, se puede aprovechar la capacidad de Proactivanet para el descubrimiento de activos con el objetivo de que la mínima información obtenida sirva como guía para que posteriormente una IA Generativa proporcione la información restante.

Capítulo 6
Solución propuesta

Partiendo de la situación actual, comentada en el apartado anterior, en este capítulo se describirá la solución propuesta. Además, se comentará la prueba de concepto realizada, en forma de Mínimo Producto Viable (MVP), y los resultados conseguidos.

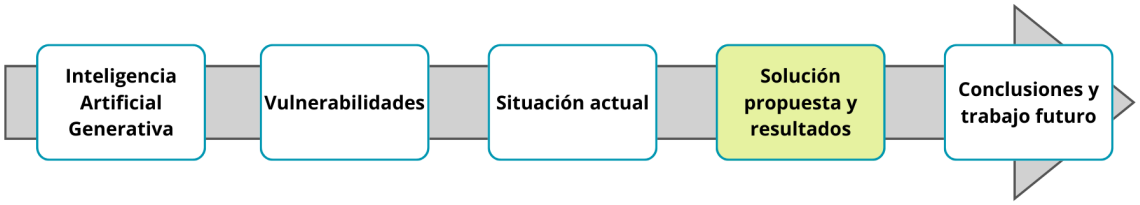

Figura 6.1: Estructura del proyecto.

6.1.- Diseño y arquitectura

En base a los objetivos establecidos, la arquitectura diseñada busca posibilitar la integración entre Proactivanet, el NVD y, en este caso, la plataforma OpenAI, y que tendrá especial enfoque en los mecanismos de comunicación basados en API REST.

Con la arquitectura propuesta se pretende ofrecer una forma de enriquecer la información de cualquier activo software, además de un mejor sistema de clasificación de software y una detección rápida de las vulnerabilidades existentes entre los activos software de la organización.

Se ha utilizado un patrón sin servidor (*serverless*) que permite la ejecución de código sin necesitad de aprovisionamiento ni de administrar servidores. A pesar de utilizar servidores, estos se encuentran aislados del desarrollo. Será el proveedor quién se encargue de las tareas de preparación, mantenimiento y adaptación de la infraestructura. Para la implementación solo será necesario el empaquetado del código. Las aplicaciones de este tipo

responden a la demanda, amplíandose o reduciéndose en función de las necesidades de cada momento. Para este caso concreto, este tipo de patrón es adecuado dadas las condiciones de uso que podría tener, ya que las peticiones no se recibirán siempre en la misma cantidad y, por ejemplo, en casos donde se quiera realizar el enriquecimiento por primera vez de los activos software en una organización, el uso será mucho mayor en comparación con cualquier otro momento [42]. Además, la informática sin servidor ofrece varias ventajas:

- Reduce el coste de operación y aumenta la productividad gracias a la delegación de tareas de implementación y gestión. También se reducen los costos de operación gracias al pago por procesamiento según la necesidad.
- Facilita un enfoque de DevOps, eliminando la necesidad de describir de forma explícita la infraestructura necesaria.
- Permite optimizar el desarrollo incorporando elementos de otros proveedores.

A cambio de obtener dichos beneficios se pierde capacidad de gestión y flexibilidad del sistema y se generá un alto grado de dependecia del proveedor que gestione los diferentes servicios utilizados.

En base a todo lo anterior, se ha propuesto, como se puede observar en la figura 6.2, la siguiente arquitectura sobre *Amazon Web Services* (AWS).

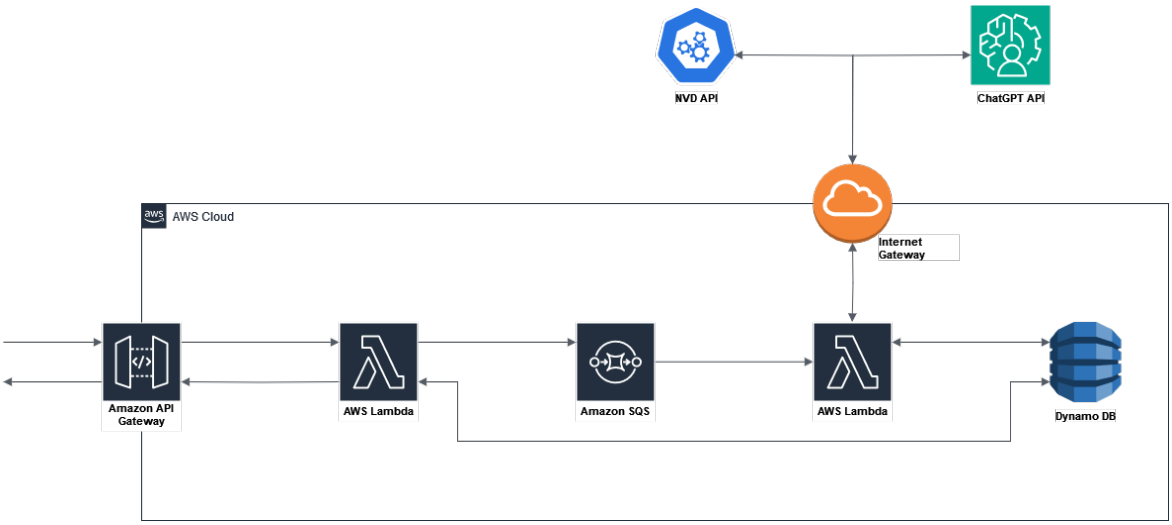

Figura 6.2: Diagrama de la arquitectura diseñada.

Para la solución descrita se ha descartado el uso del almacenamiento de datos seguro en la nube (S3) de AWS, utilizando a cambio una DynamoDB. Dado que el tipo de información que se va a almacenar estará en formato JSON, DynamoDB es una opción mucho más eficiente y eficaz.

También se ha descartado el uso de una única función Lambda para usar a cambio dos funciones Lambda separadas por una cola SQS, una para procesar la petición y enviar la información enriquecida como respuesta y otra para hacer la búsqueda de la información haciendo uso de las APIs de OpenAI (haciendo preguntas a los modelos) y del NVD (buscándo CPEs relacionados). El objetivo de utilizar la cola y dos funciones Lambda, en vez de una sola función Lambda, es poder regular el número de accesos a las APIs externas para evitar un crecimiento desmesurado de los costes y evitar llegar a los límites impuestos por el *tier* elegido para utilizar la API de ChatGPT.

6.1.1.- Proactivanet

La aplicación de Proactivanet servirá como fuente de información esencial para la búsqueda de nueva información de cada activo software, así como de *display* para la nueva información obtenida.

6.1.2.- Amazon API Gateway

Es un servicio completamente administrado que facilitará la creación y uso de la API y que, además, funcionará como puerta de entrada para que las aplicaciones, en este caso Proactivanet, puedan acceder a esta nueva funcionalidad de enriquecimiento de la información de los activos software. El Amazon API Gateway recibe las peticiones con la información mínima del software para el cual se solicita el enriquecimiento. Cada petición generará un evento con la información recibida y, a su vez, este creará una nueva instancia de la Lambda contigua. [43]

6.1.3.- AWS Lambda

Es un servicio sin servidor que permite ejecutar código en respuesta a los eventos, en este caso, aquellos procedentes de la API Gateway. Proporciona escalamiento automático, optimización del rendimiento y pago por uso, siendo muy adecuado en este contexto tanto

para organizaciones con grandes cantidades de activos software como para organizaciones más pequeñas con un número de activos más limitado [44]. Para la solución propuesta se requiere de dos Lambdas distintas, cada una con una función y objetivo diferente:

- Lambda de respuesta: primera Lambda (aquella conectada directamente con la API Gateway). Se encargará de procesar cada petición. Para cada activo software buscará la información en la DynamoDB, que será descrita más adelante. En caso de hallar información válida para dicho activo software, devolverá a cliente la información mencionada como respuesta. En cambio, si la información no es considerada como válida o no existe ninguna entrada para el activo software en la DynamoDB, se buscará la información con ayuda de ChatGPT y el NVD.
- Lambda de búsqueda: segunda Lambda (aquella conectada a la SQS y al Internet Gateway). Será la encargada de obtener la información para el activo software, valiéndose de las APIs del NVD y ChatGPT. Una vez ha obtenida la información para el activo software, esta se almacenará en una nueva entrada en la DynamoDB o se actualizará la existente para el activo mencionado.

6.1.4.- Amazon SQS

Es una cola de mensajes completamente administrada destinada, entre otras, a aplicaciones sin servidor. Permite el envío, almacenamiento y recepción de mensajes entre componentes. Se encargará de comunicar las Lambdas mencionadas anteriormente, en aquellos casos en los que sea necesario realizar una nueva búsqueda de información. Tendrá la función de regulador, es decir, limitará la cantidad de Lambdas de búsqueda generadas. De esta forma, se limita el número de accesos a las APIs externas. [45]

6.1.5.- DynamoDB

Es un servicio de base de datos NoSQL completamente administrado sin servidor. Se encargará de almacenar la información de cada activo software para el que se solicite el enriquecimiento, lo que puede incluir el nombre y versión del software, información enriquecida que se ha obtenido de las APIs de OpenAI y el NVD, además de la fecha de obtención de dicha información (utilizada para evitar respuestas con información desactualizada). [46]

6.1.6.- Internet Gateway

Permite a los productos de AWS, en este caso concreto a la Lambda de búsqueda, acceder a servicios externos, en este caso a las APIs del NVD y ChatGPT. [47]

6.1.7.- APIs externas

A través de la API de ChatGPT se realizarán las preguntas pertinentes para obtener la información "enriquecida" de cada activo software para el que se solicite. A través de la API del NVD se obtendrán los CPE relacionados con el activo software para el que se solicita la información.

6.2.- Mínimo producto viable

Para la prueba de concepto se ha desarrollado un MVP con el se pretende demostrar la utilidad de la solución descrita anteriormente, además de los posibles beneficios de la utilización de las IAs Generativas en SAM. Sin embargo, no se pretende buscar una optimización en tiempo de respuesta o coste, si no la claridad y calidad de la información obtenida.

Incluirá como única funcionalidad la búsqueda de información para cada activo software, tanto de forma "síncrona" como "asíncrona", a partir del nombre y versión del activo software para el que se realiza. Tanto la propia obtención del nombre y la versión como la representación de la información obtenida no se incluyen en el desarrollo, únicamente se incluye el procesado de petición (que tendrá que incluir nombre, versión y tipo de respuesta, síncrona o asíncrona), la obtención de la información enriquecida, su almacenamiento y su posterior envío en forma de respuesta.

Para cada activo software se han decidido, como campos relevantes para el enriquecimiento de la información, los mencionados a continuación:

- Última versión: última versión conocida del software.
- Fecha de salida de la última versión.
- Fecha de fin de vida útil (EOL) de la última versión.

- Tipo de licencia: licencia de software propietario, licencia de software libre (GNU o GPL), licencia de software de código abierto, licencia de software de dominio público o licencia de software como servicio (SaaS).
- Categoría: categorización realizada por la IA Generativa sin tener en cuenta las utilizadas en Proactivanet.
- Página web.
- URL de descarga del software.
- Descripción corta y larga del software.
- Alternativas gratuitas y comerciales.
- Categoría específica de Proactivanet: categoría asignada por la IA Generativa basándose en las categorías existentes en Proactivanet.
- CPEs relacionados con la versión del software para el que se realiza la búsqueda.

El funcionamiento del MVP para obtener la información enriquecida de un activo software se puede describir en los siguientes pasos:

1. Llega la petición con el nombre y la versión del activo software y el tipo de respuesta requerido, generando un evento que invoca a la Lambda de respuesta.
2. La Lambda de respuesta busca el activo software, según su nombre y versión, en la DynamoDB. En función de si la búsqueda es exitosa o no procederá de una de las siguientes formas:
    - Si encuentra la información sobre el activo software en la DynamoDB y además es válida (suficientemente reciente para no cosiderarla desactualizada), la devuleve como respuesta a la petición realizada.
    - Si no encuentra la información sobre el activo software en la DynamoDB o la información que encuentra no es válida, envía la información del activo software a la SQS para que una Lambda de búsqueda obtenga la información enriquecida y la almacene en la DynamoDB.
3. En caso de tener que obtener la información, y tras haber enviado la información del

activo a la SQS, puede proceder de dos formas en función del tipo de respuesta requerida:

- Síncrona: comienza de nuevo el proceso anterior, el número de veces que se haya determinado, hasta que pueda obtener información válida de la DynamoDB. Si no es capaz de obtener la información tras el número máximo de intentos permitidos, se avisa al cliente con un error HTTP 404.
- Asíncrona: al no ser capaz de obtener la información, se avisa al cliente con un error HTTP 404.

6.2.1.- Peticiones

Las peticiones realizadas a la API desplegada en AWS incluirán, en el cuerpo de la petición únicamente, el nombre y la versión del activo software, además del tipo de respuesta solicitado.

Las peticiones realizadas a la API de ChatGPT incluirán, en el cuerpo de la petición, el modelo que se debe utilizar para generar la respuesta, los mensajes enviados al modelo y el tipo de respuesta requerido, en este caso, formato JSON.

Las peticiones realizadas a la API del NVD incluirán como parámetro únicamente el nombre del activo software para el que se realiza la búsqueda de CPEs relacionados para, posteriormente, filtrar en la propia Lambda de búsqueda, mencionada anteriormente, en función de la versión.

6.2.2.- Modelos de ChatGPT utilizados

En este caso concreto se utilizarán dos modelos distintos y con diferentes objetivos para obtener todos los campos relevantes mencionados con anterioridad:

- ChatGPT-4-Turbo: este modelo ha sido entrenado con la información más reciente (diciembre de 2023) en comparación con el resto de los modelos existentes actualmente. Se utilizará para la búsqueda de todos los campos mencionados anteriormente, exceptuando a la categoría específica de Proactivanet y los CPEs relacionados.

- ChatGPT-3.5-Turbo (con *fine-tuning*): este modelo se ha utilizado para realizar *fine-tuning* con el único objetivo de categorizar el activo software que recibe como pregunta en base a las categorías existentes en Proactivanet. Para su entrenamiento se han utilizado pares pregunta-respuesta que contenían el nombre y versión del software en la pregunta y la categorización completa como respuesta. Estos ejemplos han sido obtenidos tanto de Proactivanet (utilizados como datos de validación para el entrenamiento), como de preguntas realizadas al propio ChatGPT (utilizados como datos de entrenamiento).

6.2.3.- *Fine-tuning* de ChatGPT 3.5 Turbo

La finalidad del entrenamiento de ChatGPT-3.5-Turbo (modelo disponible para *fine-tuning* al momento de realizar el entrenamiento) es obtener un modelo capaz de clasificar de forma eficaz cada activo software en base a las categorías que exiten dentro de Proactivanet.

Partiendo de las categorías mencionadas, en formato XML (véase 4), se ha generado un ejemplo de activos software para cada una de ellas con ayuda de otro modelo de IA Generativa, en este caso ChatGPT-4-Turbo, partiendo únicamente de la categoría completa (desde la categoría principal hasta la subcategoría en concreto). Para cada categoría se ha indicado al modelo que devuelva un ejemplo que pueda clasificarse según esa categoría, y se han guardado cada uno de los pares categoría-ejemplo.

```
<Category id="identificador categoría" creationDateTime="2010/10/10 00:00:00"
↪  lastUpdateDateTime="2010/10/10 00:00:00" name="nombre"
↪  panSoftCategories_id="identificador categoría padre">
    <CategoryRules>
        <CategoryRule id="identificador regla" creationDateTime="2010/10/10 00:00:00 AM"
        ↪  lastUpdateDateTime="2010/10/10 00:00:00 AM" pattern="patrón de la regla"
        ↪  patternMode="tipo de patrón" active="true" vendorPattern=""
        ↪  vendorPatternMode="" panAgentTypes="0;0;0000;000;000;000;000"/>
    </CategoryRules>
    <ProcessRules/>
</Category>
```

Código 4: Ejemplo de categoría en formato XML de Proactivanet. [41]

Cada categoría cuenta con un identificador, las fechas de creación y última actua-

lización, el nombre y el identificador de la categoría padre que la engloba (*panSoftCategories_id*). También tendrá asociadas una o varias reglas de categorización utilizadas para categorizar un activo software cualquiera con esa categoría si cumple una de las reglas.

Posteriormente, se han generado las preguntas de entrenamiento partiendo de los pares categoría-ejemplo mencionados antes en formato JSON Lines (véase 5). Para el entrenamiento no será necesario que la respuesta esté en formato JSON o que la pregunta lo exija para evitar errores a la hora de utilizar el modelo entrenado. Cuando se utilice el modelo con *fine-tuning*, se indicará explícitamente que la respuesta esté en formato JSON.

```
{"messages": [
    {
        "role": "system",
        "content": "El sistema ofrece orientación para la identificación y clasificación
        ↪  precisa del software según las categorías definidas por Proactivanet."
    },
    {
        "role": "user",
        "content": "¿Como clasificarías el software \"Ad-Aware 12.10\" según las
        ↪  categorías (y/o subcategorías) de Proactivanet?"
    },
    {
        "role": "assistant",
        "content": "Clasificaría el software \"Ad-Aware 12.10\" según las categorías (y/o
        ↪  subcategorías) de Proactivanet así: \"/Antivirus y Seguridad/Ad-Aware\""
    }
]}
```

Código 5: Ejemplo de conjunto de mensajes de categorización para *fine-tuning* del modelo.

Se ha creado un archivo de entrenamiento y otro de validación (ejemplos y categorización obtenidos directamente de Proactivanet) utilizando ejemplos como el anterior. Con ambos archivos se ha entrenado a ChatGPT-3.5-Turbo para que realice la tarea de categorización de activos software. La respuesta del modelo para, por ejemplo, *7-Zip*, sería la que puede observarse en la figura 6.

```
{
    "category": "/Utilidades/Compresores/7-zip"
}
```

Código 6: Ejemplo de respuesta par *7-Zip* del modelo con *fine-tuning*.

6.2.4.- Almacenamiento de la información en la DynamoDB

La DynamoDB utilizada contendrá los siguientes elementos para almacenar, además de la información enriquecida, aquellos datos necesarios para su búsqueda y validación:

- Nombre del activo software.
- Versión del activo software.
- Fecha de la última actualización de la información.
- Información enriquecida para el activo software.

Los campos de nombre y versión se utilizan como clave principal y clave secundaria respectivamente y serán utilizados para la búsqueda de información en la base de datos. El campo de fecha se utilizará como validador de la información, es decir, si la información es suficientemente reciente o no. Por último, el campo de información enriquecida incluirá toda la información que se obtuvo de la búsqueda realizada.

6.3.- Resultados

Para visualizar adecuadamente los resultados obtenidos (véase 8) partiremos de una petición de ejemplo (véase 7) con los siguientes campos para un activo software de Proactivanet:

```
{
    "name": "7-zip",
    "version": "22.01",
    "responseType": "sync"
}
```

Código 7: Cuerpo de la petición en formato JSON.

```
{
    "SoftwareName": "7-zip",
    "SoftwareVersion": "22.01",
    "SoftwareInformationDate": "7/6/2024 4:48:29 AM",
    "SoftwareData": {
        "latestVersion": "23.01",
        "latestVersionLaunchDate": "2023-07-15",
        "latestVersionEOLDate": null,
        "licenseType": "GNU LGPL",
        "softwareCategory": "Compresión de archivos",
        "website": "https://www.7-zip.org/",
        "downloadUrl": "https://www.7-zip.org/download.html",
        "shortDescription": "7-Zip es un software de compresión de archivos gratuito y de
        ↪  código abierto.",
        "longDescription": "7-Zip es una herramienta de compresión de archivos gratuita y
        ↪  de código abierto que admite una amplia variedad de formatos de archivo,
        ↪  incluidos 7z, ZIP, RAR, CAB, GZIP, BZIP2, TAR y muchos más. Es conocido por
        ↪  su alta relación de compresión y su capacidad para manejar archivos grandes
        ↪  de manera eficiente. Además, 7-Zip ofrece características avanzadas como la
        ↪  creación de archivos autoextraíbles, la encriptación AES-256 y la integración
        ↪  con el menú contextual de Windows.",
        "topFreeAlternatives": [
            "PeaZip",
            "WinRAR (versión de prueba)",
            "Bandizip",
            "Zipware"
        ],
        "topCommercialAlternatives": [
            "WinRAR (versión completa)",
            "WinZip",
            "PowerArchiver"
        ],
        "proactivanetCategory": "Utilidades/Compresores/7-zip",
        "relatedCpe": [
            {
                "deprecated": false,
                "cpeName": "cpe:2.3:a:7-zip:7-zip:-:*:*:*:*:*:*:*",
                "cpeNameId": "A0A96E53-21E1-41C7-8DA4-43EB68A3D917",
                "namelastModified": null,
                "created": "2022-04-25T16:29:41.797"
            },
            {
                "deprecated": false,
                "cpeName": "cpe:2.3:a:7-zip:7-zip:22.01:*:*:*:*:*:*:*",
                "cpeNameId": "97E2C2FC-FE05-4257-9406-C60EC8D93056",
                "namelastModified": null,
                "created": "2023-11-09T14:10:59.897"
            }
        ]
    }
}
```


Código 8: Cuerpo de la respuesta en formato JSON.

De igual forma, en cuanto a tiempos de respuesta, no se observa una variación notable en función de si la información existe en la DynamoDB y es válida, o si, por el contrario, es necesaria su búsqueda. También puede observarse muy poca variación en el tiempo de respuesta debido a la cantidad de CPEs relacionados con el activo software.

Debido al arranque en frío, fuera del periodo de retención del entorno de ejecución, sí se denota cierto aumento en el tiempo de respuesta si las Lambda llevan cierto tiempo sin haberse utilizado. Esto se debe a que las funciones Lambda, una vez termina su invocación, entorno de ejecución se mantiene durante un periodo de tiempo, pudiendo ser reutilizado si llega otra petición.

Para poder verlo gráficamente, se ha probado con 10 activos software diferentes para ver el tiempo de respuesta en todos los casos posibles, ya sea obtención directa de la DynamoDB o búsqueda de la información a través de las APIs de OpenAI y el NVD.

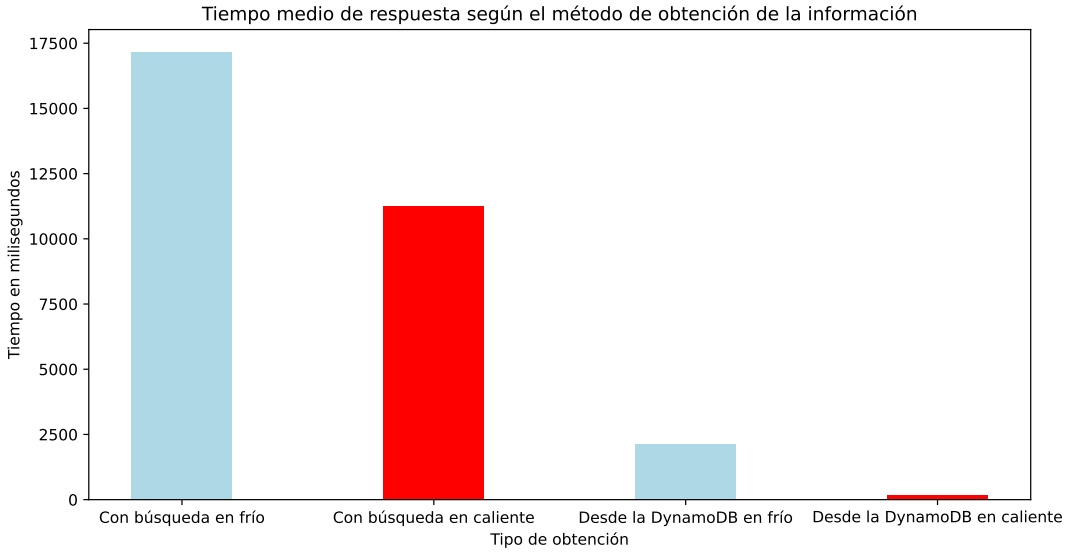

Figura 6.3: Tiempo medio de respuesta según el tipo de obtención.

Capítulo 7
Conclusiones y trabajo futuro

En este proyecto se han abordado diferentes formas de utilizar la IA Generativa y todas aquellas ventajas que su uso puede aportar para el enriquecimiento de la información relacionada con los activos software de las organizaciones a través de un software de gestión de activos como es Proactivanet.

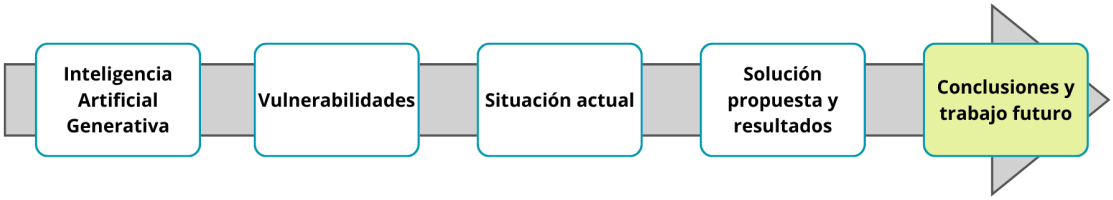

Figura 7.1: Estructura del proyecto.

Basando el trabajo realizado a partir de la situación actual existente en Proactivanet, se ha llevado ha cabo el desarrollo e implementación de un MVP que permite la obtención y almacenamiento de la información enriquecida de forma eficiente y efectiva, cumpliendo con los objetivos de calidad y relevancia de la información. El uso de ChatGPT como IA generativa ha sido altamente efectivo para la obtención de dicha información, gracias a los datos tan recientes con los que han sido entrenados sus últimos modelos, además de su sencillez en cuestiones de implementación y administración.

Sin embargo, a lo largo de el proyecto han salido a relucir las principales limitaciones y desafíos a los que se enfrenta. Algunos de los retos principales han sido la obtención de información suficientemente reciente en las respuestas de las IAs Generativas o la generación de datos de entrenamiento para su posterior uso en el *fine-tuning* con resultados admisibles. Además, dadas las necesidades expuestas en apartados anteriores, ha sido necesario el uso de múltiples modelos de ChatGPT y la realización de varias peticiones para obtener la información enriquecida, cuando la solución óptima, en términos de tiempo de respuesta, coste y

rendimiento, hubiera sido utilizar un único modelo, aunque en ese caso se perdería calidad en la información.

En cuanto a futuras mejoras, se pueden distinguir entre aquellas dependientes del avance de las aplicaciones externas utilizadas y todas las que dependen únicamente de mejoras y cambios realizados en la arquitectura y el MVP desarrollado. En casos como ChatGPT, donde se realizan mejoras y avances continuamente, cabe esperar la llegada de nuevos y mejores modelos con mejor entrenamiento y con datos más recientes, además de nuevas funcionalidades que faciliten las tareas realizadas por la arquitectura expuesta como solución frente a la situación actual. También se ha de tener en cuenta el uso de IAs Generativas diferentes a ChatGPT, existentes o futuras, que puedan aportar soluciones diferentes y de mayor calidad a las utilizadas en la solución propuesta.

Por otra parte, mejoras en los datos de entrenamiento, el uso de un único modelo para la búsqueda de información en lugar de dos como es este caso, o incluso la utilización de RAG (generación mejorada por recuperación) para la optimización de la salida de un LLM haciendo referencia a una base de conocimientos externa a los datos de entrenamiento del modelo para generar la respuesta, pueden suponer mejoras notables en el rendimiento y calidad de la generación de información enriquecida sin necesidad de esperar por los avances externos. [48]

Como conclusión y a modo de resumen, la solución propuesta ha permitido entrever los posibles beneficios que puede aportar las IAs Generativas a la gestión de activos software y las posibilidades que puede ofrecer para facilitar y mejorar las tareas que se llevan a cabo.

Bibliografía

[1] S. Rodríguez, "La Revolución Creativa: La Historia Completa de la Inteligencia Artificial Generativa," https://bigdatamagazine.es/la-revolucion-creativa-la-historia-completa-de-la-inteligencia-artificial-generativa, Nov. 2023, accedido por última vez: julio 2024.

[2] "Generative AI to Become a $1.3 Trillion Market by 2032, Research Finds," https://www.bloomberg.com/company/press/generative-ai-to-become-a-1-3-trillion-market-by-2032-research-finds/, 2024, accedido por última vez: julio 2024.

[3] A. Figueiredo, "The Generative AI Revolution - A Primer," https://afigueiredo.medium.com/the-generative-ai-revolution-a-primer-19382961bba9, Jun. 2023, accedido por última vez: julio 2024.

[4] "+150 Estadísticas de Inteligencia Artificial en 2024," https://www.techopedia.com/es/estadisticas-inteligencia-artificial, 2024, accedido por última vez: julio 2024.

[5] "What is Generative AI?" https://aws.amazon.com/what-is/generative-ai/, 2024, accedido por última vez: julio 2024.

[6] M. Merino, "Conceptos de inteligencia artificial: qué son las GANs o redes generativas antagónicas," https://www.xataka.com/inteligencia-artificial/conceptos-inteligencia-artificial-que-gans-redes-generativas-antagonicas, Mar. 2019, accedido por última vez: julio 2024.

[7] H. Ali, "Variational autoencoders: A vanilla implementation," https://mlarchive.com/deep-learning/variational-autoencoders-a-vanilla-implementation/, Oct. 2022, accedido por última vez: julio 2024.

[8] A. Moreno, "Procesamiento del lenguaje natural ¿Qué es?" https://www.iic.uam.es/inteligencia/que-es-procesamiento-del-lenguaje-natural/, Oct. 2017, accedido por última vez: julio 2024.

[9] "¿Qué son los grandes modelos de lenguaje (LLM)?" https://www.ibm.com/es-es/topics/large-language-models, Apr. 2024, accedido por última vez: julio 2024.

[10] P. Marsden, "Artificial Intelligence timeline infographic – from Eliza to Tay and beyond," https://digitalwellbeing.org/artificial-intelligence-timeline-infographic-from-eliza-to-tay-and-beyond/, Aug. 2017, accedido por última vez: julio 2024.

[11] M. Raza, "ITAM vs SAM: What's the difference?" https://www.bmc.com/blogs/itam-vs-sam/, 2024, accedido por última vez: julio 2024.

[12] A. G. Inc, "What exactly does ITAM consist of? There are a lot of IT industry acronyms that can be a little confusing [Tweet]," X.com https://x.com/AnglepointGroup/status/1458556328696328200, 2024, accedido por última vez: julio 2024.

[13] "¿Qué es SAM y cómo empezar?" https://www.ambit-bst.com/blog/qu%C3%A9-es-sam-y-como-empezar, 2024, accedido por última vez: julio 2024.

[14] B. Vásquez, "Proactivanet - Gestión de Activos TI," https://www.conytec.com/proactivanet/, Mar. 2021, accedido por última vez: julio 2024.

[15] K. Martineau, "What is generative AI?" https://research.ibm.com/blog/what-is-generative-AI, Apr. 2023, accedido por última vez: julio 2024.

[16] "¿Qué es un modelo lingüístico (LLM) grande?" https://www.cloudflare.com/es-es/learning/ai/what-is-large-language-model, 2024, accedido por última vez: julio 2024.

[17] "¿Qué es el aprendizaje profundo?" https://aws.amazon.com/what-is/deep-learning/, 2024, accedido por última vez: julio 2024.

[18] R. Merritt, "¿Qué Es un Modelo Transformer?" https://la.blogs.nvidia.com/blog/que-es-un-modelo-transformer/, Apr. 2022, accedido por última vez: julio 2024.

[19] "¿Qué son los transformadores en la inteligencia artificial?" https://aws.amazon.com/what-is/transformers-in-artificial-intelligence/, 2024, accedido por última vez: julio 2024.

[20] "Chat GPT: What is it?" https://uca.edu/cetal/chat-gpt/, 2024, accedido por última vez: julio 2024.

[21] "OpenAI developer platform Docs," https://platform.openai.com/docs/overview, 2024, accedido por última vez: julio 2024.

[22] "Everything You Need to Know About Meta AI," https://www.cukeragency.com/everything-you-need-know-about-meta-ai/, 2024, accedido por última vez: julio 2024.

[23] "Llama 2: open source, free for research and commercial use," https://llama.meta.com/llama2/, 2024, accedido por última vez: julio 2024.

[24] "Google Cloud console - Llama 2," https://console.cloud.google.com/vertex-ai/publishers/meta/model-garden/llama2, 2024, accedido por última vez: julio 2024.

[25] N. Barney, "What is Google AI?" https://www.techtarget.com/whatis/definition/Google-AI, 2024, accedido por última vez: julio 2024.

[26] "Google Gemini vs Azure OpenAI GPT: Pricing considerations," https://www.vantage.sh/blog/gcp-google-gemini-vs-azure-openai-gpt-ai-cost, 2024, accedido por última vez: julio 2024.

[27] T. Dang, "Fine-tuning Gemini AI model: A step-by-step guide," https://thinhdanggroup.github.io/gemini/, Feb. 2024, accedido por última vez: julio 2024.

[28] "Documentación y referencia de la API para desarrolladores de la API de Gemini," https://ai.google.dev/gemini-api/docs?hl=es-419, 2024, accedido por última vez: julio

2024.

[29] "Envía solicitudes a la API de Vertex AI para Gemini," https://cloud.google.com/vertex-ai/generative-ai/docs/start/quickstarts/quickstart-multimodal?hl=es-419, 2024, accedido por última vez: julio 2024.

[30] "Artificial Intelligence (AI) Solutions," https://www.ibm.com/artificial-intelligence, 2024, accedido por última vez: julio 2024.

[31] "IBM Watson vs ChatGPT: Diferencias en IA," https://www.toolify.ai/es/ai-news-es/ibm-watson-vs-chatgpt-diferencias-en-ia-2627827, 2024, accedido por última vez: julio 2024.

[32] "Watsonx.AI," https://cloud.ibm.com/apidocs/watsonx-ai, 2024, accedido por última vez: julio 2024.

[33] "Research \anthropic," https://www.anthropic.com/research, 2024, accedido por última vez: julio 2024.

[34] "Welcome to Claude," https://docs.anthropic.com/en/docs/welcome, 2024, accedido por última vez: julio 2024.

[35] W. b. Katerinaptrv, "Benchmarks: How do we evaluate and compare LLMs and Multimodal Models?" https://medium.com/@daniellefranca96/benchmarks-how-do-we-evaluate-and-compare-llms-and-multimodals-models-105cec4f2ad4, Dec. 2023, accedido por última vez: julio 2024.

[36] A. Joshi, "Introducing AICon LLM models: MT-bench scores," https://blog.worqhat.com/a-benchmark-excellence-of-aicon-ee6337d1e7eb, Dec. 2023, accedido por última vez: julio 2024.

[37] W.-L. Chiang, L. Zheng, Y. Sheng, A. N. Angelopoulos, T. Li, D. Li, H. Zhang, B. Zhu, M. Jordan, J. E. Gonzalez, and I. Stoica, "Chatbot arena: An open platform for evaluating llms by human preference," 2024, accedido por última vez: julio 2024. [Online]. Available: https://arxiv.org/abs/2403.04132

[38] GReAT, "Las vulnerabilidades de software," https://encyclopedia.kaspersky.es/knowledge/software-vulnerabilities/, Nov. 2015, accedido por última vez: julio 2024.

[39] B. A. Cheikes, D. Waltermire, and K. Scarfone, "Common Platform Enumeration: Naming Specification Version 2.3," Gaithersburg, MD, Tech. Rep., 2011, accedido por última vez: julio 2024.

[40] "NVD: What is the National Vulnerability Database?" https://www.lacework.com/cloud-security-fundamentals/nvd-what-is-the-national-vulnerability-database, accedido por última vez: julio 2024.

[41] "Proactivanet," https://www.proactivanet.com/, accedido por última vez: julio 2024.

[42] "¿Qué es la informática sin servidor?" https://www.redhat.com/es/topics/cloud-native-apps/what-is-serverless, accedido por última vez: julio 2024.

[43] "AWS API Gateway," https://aws.amazon.com/api-gateway/, 2024, accedido por última vez: julio 2024.

[44] "AWS Lambda," https://aws.amazon.com/lambda/, 2024, accedido por última vez: julio 2024.

[45] "AWS SQS," https://aws.amazon.com/sqs/, 2024, accedido por última vez: julio 2024.

[46] "AWS DynamoDB," https://aws.amazon.com/dynamodb/, 2024, accedido por última vez: julio 2024.

[47] "VPC Internet Gateway," https://docs.aws.amazon.com/es_es/vpc/latest/userguide/VPC_Internet_Gateway.html, 2024, accedido por última vez: julio 2024.

[48] "¿Qué es la RAG (generación aumentada por recuperación)?" https://aws.amazon.com/what-is/retrieval-augmented-generation/, accedido por última vez: julio 2024.

Apéndice A
Descripción del código desarrollado

En este anexo se explicará detalladamente el código principal desarrollado para la prueba de concepto, y las clases y métodos que lo conforman.

A.1.- Estructura

La solución creada para este trabajo consta de dos proyectos separados, destinados a cada una de las Lambdas descritas en la solución propuesta, siendo *TFG_SwInformation_Api* el destinado a la Lambda de respuesta, y *TFG_SwInformation_Updater* el destinado a la Lambda de búsqueda.

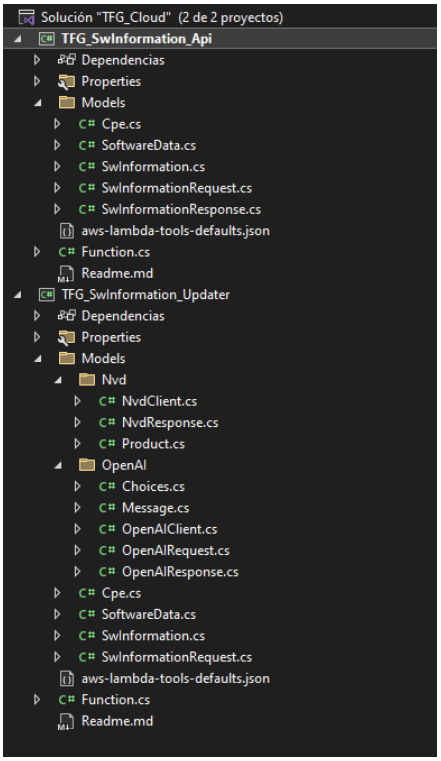

Figura A.1: Estructura de archivos de la solución.

A.2.- Código principal

Cada Lambda ejecutará el código contenido en el archivo *Function.cs* del proyecto que se le proporciona. Además, cada una hará uso de las diferentes clases mostradas más adelante para serializar, deserializar, procesar y almacenar la información con la que trabaje en cada momento. En este caso, aquellos archivos nombrados de la misma forma en ambos proyectos tienen el mismo código, por lo que cada clase se describe una única vez en este anexo.

A.2.1.- Código de la Lambda de respuesta

Esta parte del código es la encargada de procesar las peticiones recibidas por la API Gateway.

```
public async Task<APIGatewayProxyResponse> FunctionHandler(APIGatewayProxyRequest request,
↪  ILambdaContext context)
{
    var swInformationRequest =
    ↪  JsonSerializer.Deserialize<SwInformationRequest>(request.Body);
    if (swInformationRequest == null)
    {
        return new APIGatewayProxyResponse
        {
            StatusCode = (int)HttpStatusCode.BadRequest
        };
    }

    if (swInformationRequest.ResponseType.ToLower().Equals("sync"))
    {
        var swInformationResponse = await ProccessRequestSync(swInformationRequest);
        return swInformationResponse;
    }
    else if (swInformationRequest.ResponseType.ToLower().Equals("async"))
    {
        var swInformationResponse = await ProccessRequestAsync(swInformationRequest);
        return swInformationResponse;
    }
    else
    {
        return new APIGatewayProxyResponse
        {
            StatusCode = (int)HttpStatusCode.BadRequest
        };
    }
}
```

Código 9: Código *Function.cs* Lambda de respuesta parte 1.

Para su funcionamiento se proporcionan como variables de entorno el nombre de la cola SQS a la que deberá enviar las peticiones cuando no sea capaz de obtener la información de la DynamoDB, el número máximo de intentos de obteneción de la información de la DynamoDB, el intervalo de espera entre intentos, y el tiempo de vida de la información. También serán necesarias las variables correspondientes para la DynamoDB y el cliente de la SQS.

```
private readonly IDynamoDBContext _dbContext;
private readonly IAmazonSQS _sqsClient;
private readonly string? _sqsQueueName =
↪  Environment.GetEnvironmentVariable("sqsQueueName");
private readonly int _maxRetries =
↪  Convert.ToInt16(Environment.GetEnvironmentVariable("maxRetries"));
private readonly int _delay =
↪  Convert.ToInt16(Environment.GetEnvironmentVariable("retryDelay"));
private readonly int _lifeTime =
↪  Convert.ToInt16(Environment.GetEnvironmentVariable("lifeTime"));
```

Código 10: Código *Function.cs* Lambda de respuesta parte 2.

Procesará cada petición en función del tipo de respuesta requerido, *ProcessRequestSync* para el caso síncrono o *ProcessRequestAsync* para el caso asíncrono. En cualquiera de los casos, devolverá como respuesta la información enriquecida del activo software para el que se solicita en caso de poder obtenerla o un error 404 en caso de no ser posible.

```
public async Task<APIGatewayProxyResponse> ProccessRequestSync(SwInformationRequest
↪  swInformationRequest)
{
    try
    {
        for (int i = 0; i < _maxRetries; i++)
        {
            var swInformation = await
            ↪  _dbContext.LoadAsync<SwInformation>(swInformationRequest.Name.ToLower(),
            ↪  swInformationRequest.Version.ToLower());
            if (swInformation != null && ValidSoftwareInformation(swInformation))
            {
                var swInformationResponse = new SwInformationResponse(swInformation);
                Console.WriteLine(swInformationResponse);
                return new APIGatewayProxyResponse
                {
                    StatusCode = (int)HttpStatusCode.OK,
                    Body = JsonSerializer.Serialize(swInformationResponse),
                    Headers = new Dictionary<string, string> { { "Content-Type",
                    ↪  "application/json" } }
                };
            }

            var sqsUrl = await _sqsClient.GetQueueUrlAsync(_sqsQueueName);
            await _sqsClient.SendMessageAsync(new SendMessageRequest
            {
                QueueUrl = sqsUrl.QueueUrl,
                MessageBody = JsonSerializer.Serialize(swInformationRequest),
            });
            await Task.Delay(_delay);
        }

        return new APIGatewayProxyResponse
        {
            StatusCode = (int)HttpStatusCode.NotFound,
            Body = "Information not found. Request enqueued. Try again later!",
            Headers = new Dictionary<string, string> { { "Content-Type", "text/plain" } }
        };
    }
    catch (Exception ex)
    {
        return new APIGatewayProxyResponse
        {
            StatusCode = (int)HttpStatusCode.InternalServerError
        };
    }
}
```

Código 11: Código *Function.cs* Lambda de respuesta parte 3.

```
public async Task<APIGatewayProxyResponse> ProccessRequestAsync(SwInformationRequest
↪  swInformationRequest)
{
    var swInformation = await
    ↪  _dbContext.LoadAsync<SwInformation>(swInformationRequest.Name.ToLower(),
    ↪  swInformationRequest.Version.ToLower());
    if (swInformation != null && ValidSoftwareInformation(swInformation))
    {
        return new APIGatewayProxyResponse
        {
            StatusCode = (int)HttpStatusCode.OK,
            Body = JsonSerializer.Serialize(new SwInformationResponse(swInformation)),
            Headers = new Dictionary<string, string> { { "Content-Type",
            ↪  "application/json" } }
        };
    }

    var sqsUrl = await _sqsClient.GetQueueUrlAsync(_sqsQueueName);
    await _sqsClient.SendMessageAsync(new SendMessageRequest
    {
        QueueUrl = sqsUrl.QueueUrl,
        MessageBody = JsonSerializer.Serialize(swInformationRequest),
    });
    return new APIGatewayProxyResponse
    {
        StatusCode = (int)HttpStatusCode.NotFound,
        Body = "Information not found. Request enqueued. Try again later!",
        Headers = new Dictionary<string, string> { { "Content-Type", "text/plain" } }
    };
}
```

Código 12: Código *Function.cs* Lambda de respuesta parte 4.

Para la comprobación de la validez de la información se utiliza la función *ValidSoftwareInformation*.

```
public bool ValidSoftwareInformation(SwInformation swInformation)
{
    if (swInformation.SoftwareInformationDate != null)
    {
        var timeDiff = (DateTime.Now -
        ↪  DateTime.Parse(swInformation.SoftwareInformationDate)).Days;
        var maxTimeDiff = _lifeTime;
        if (timeDiff <= maxTimeDiff) { return true; }
    }
    return false;
}
```

Código 13: Código *Function.cs* Lambda de respuesta parte 5.

A.2.2.- Código de la Lambda de búsqueda

Esta parte del código es la encargada de procesar las peticiones para las que sea necesaria realizar la búsqueda (nuevo activo software o información desactualizada).

```
public async Task FunctionHandler(SQSEvent sqsEvent, ILambdaContext context)
{
    await ProcessMessage(sqsEvent.Records[0], context);
}

public async Task ProcessMessage(SQSEvent.SQSMessage message, ILambdaContext context)
{
    var swInformationRequest =
    ↪  JsonSerializer.Deserialize<SwInformationRequest>(message.Body);
    var swInformation = GetSoftwareInformation(swInformationRequest.Name,
    ↪  swInformationRequest.Version);
    await _dbContext.SaveAsync(swInformation);
}
```

Código 14: Código *Function.cs* Lambda de búsqueda parte 1.

Para su funcionamiento se proporcionan como variables de entorno las ApiKeys y URIs correspondientes a las APIs de ChatGPT y el NVD, los modelos que se van a utilizar para obtener tanto la información general como la categoría específica de Proactivanet, el valor de temperatura proporcionado a ChatGPT para generar las respuestas y un valor booleano que indique si la búsqueda de CPEs relacionados ha de ser exacta o no. También será necesaria la variable correspondiente para la DynamoDB.

```
private readonly IDynamoDBContext _dbContext;
private readonly string? _openaiApiKey =
↪  Environment.GetEnvironmentVariable("openaiApiKey");
private readonly string? _openaiUri = Environment.GetEnvironmentVariable("openaiUri");
private readonly string? _openaiInfoModel =
↪  Environment.GetEnvironmentVariable("openaiInfoModel");
private readonly string? _openaiCategoryModel =
↪  Environment.GetEnvironmentVariable("openaiCategoryModel");
private readonly string? _openaiTemperature =
↪  Environment.GetEnvironmentVariable("openaiTemperature");
private readonly string? _nvdApiKey = Environment.GetEnvironmentVariable("nvdApiKey");
private readonly string? _nvdUri = Environment.GetEnvironmentVariable("nvdUri");
private readonly bool _nvdExactMatchSearch =
↪  Convert.ToBoolean(Environment.GetEnvironmentVariable("nvdExactMatchSearch"));
```

Código 15: Código *Function.cs* Lambda de búsqueda parte 2.

La función *GetSoftwareInformation* se utilizará para obtener la información enriquecida para el activo software solicitado.

```
public SoftwareInformation GetSoftwareInformation(string swName, string swVersion)
{
    var swData = GetSoftwareData(swName);
    swData.proactivanetCategory = GetSoftwareProactivanetCategory(swName);
    swData.relatedCpe = GetSoftwareRelatedCpe(swName, swVersion);
    var softwareInformation = new SoftwareInformation()
    {
        SoftwareName = swName.ToLower(),
        SoftwareVersion = swVersion.ToLower(),
        SoftwareData = JsonSerializer.Serialize(swData),
        SoftwareInformationDate = DateTime.Now.ToString()
    };

    return softwareInformation;
}
```

Código 16: Código *Function.cs* Lambda de búsqueda parte 3.

La función *GetSoftwareData* se utilizará para enviar una petición con los mensajes necesarios para obtener como respuesta la información enriquecida sobre el activo software a la API de ChatGPT , sin contar la categoría específica de Proactivanet ni el listado de CPEs relacionados.

```
public string GetSoftwareProactivanetCategory(string swName)
{
    var openaiClient = new OpenAIClient(_openaiApiKey, _openaiUri);
    var openaiMessages = new string[]
    {
        "{\"role\": \"system\", \"content\": \"El sistema ofrece orientación para la
        ↪  identificación y clasificación precisa del software según las categorías
        ↪  definidas por Proactivanet.\"}",
        "{\"role\": \"user\", \"content\": \"¿Como clasificarías el software " + swName +
        ↪  " según las categorías (y/o subcategorías de Proactivanet)? Responde en
        ↪  formato JSON rellenando las xxx: 'category':'xxx/xxx'.\"}"
    };
    var response = openaiClient.Create(_openaiCategoryModel, openaiMessages,
    ↪  _openaiTemperature);
    var swProactivanetCategory = response.Split(':')[1].Split('\"')[1];

    return swProactivanetCategory;
}
```

Código 17: Código *Function.cs* Lambda de búsqueda parte 4.

La función *GetSoftwareProactivanetCategory* se utilizará para enviar una petición a la API de ChatGPT con los mensajes necesarios para obtener como respuesta la categoría específica de Proactivanet.

```
public string GetSoftwareProactivanetCategory(string swName)
{
    var openaiClient = new OpenAIClient(_openaiApiKey, _openaiUri);
    var openaiMessages = new string[]
    {
        "{\"role\": \"system\", \"content\": \"El sistema ofrece orientación para la
        ↪  identificación y clasificación precisa del software según las categorías
        ↪  definidas por Proactivanet.\"}",
        "{\"role\": \"user\", \"content\": \"¿Como clasificarías el software " + swName +
        ↪  " según las categorías (y/o subcategorías de Proactivanet)? Responde en
        ↪  formato JSON rellenando las xxx: 'category':'xxx/xxx'.\"}"
    };
    var response = openaiClient.Create(_openaiCategoryModel, openaiMessages,
    ↪  _openaiTemperature);
    var swProactivanetCategory = response.Split(':')[1].Split('\"')[1];

    return swProactivanetCategory;
}
```

Código 18: Código *Function.cs* Lambda de búsqueda parte 5.

La función *GetSoftwareRelatedCpe* se utilizará para enviar una petición con el nombre del activo software a la API del NVD para obtener como respuesta el listado de CPEs relacionados, que posteriormente se filtrarán según la versión del activo software.

```
public Cpe[] GetSoftwareRelatedCpe(string SoftwareName, string SoftwareVersion)
        {
            var nvdClient = new NvdClient(_nvdApiKey);
            var nvdApiUri = _nvdUri + SoftwareName;
            if (_nvdExactMatchSearch)
            {
                nvdApiUri += "&keywordExactMatch";
            }

            return nvdClient.GetRelatedCpe(SoftwareName, SoftwareVersion, nvdApiUri);
        }
```

Código 19: Código *Function.cs* Lambda de búsqueda parte 6.

A.3.- Clases utilizadas

Para el serializado, deserializado, procesado y almacenamiento en la DynamoDB de la información enriquecida se definen varias clases.

A.3.1.- Cpe.cs

```
public class Cpe
{
    [JsonPropertyName("deprecated")]
    public bool? deprecated { get; set; }

    [JsonPropertyName("cpeName")]
    public string? cpeName { get; set; }

    [JsonPropertyName("cpeNameId")]
    public string? cpeNameId { get; set; }

    [JsonPropertyName("namelastModified")]
    public string? lastModified { get; set; }

    [JsonPropertyName("created")]
    public string? created { get; set; }
}
```

Código 20: Código *Cpe.cs*.

A.3.2.- SoftwareData.cs

```
public class SoftwareData
{
    [JsonPropertyName("latestVersion")]
    public string? latestVersion { get; set; }

    [JsonPropertyName("latestVersionLaunchDate")]
    public string? latestVersionLaunchDate { get; set; }

    [JsonPropertyName("latestVersionEOLDate")]
    public string? latestVersionEOLDate { get; set; }

    [JsonPropertyName("licenseType")]
    public string? licenseType { get; set; }

    [JsonPropertyName("softwareCategory")]
    public string? softwareCategory { get; set; }

    [JsonPropertyName("website")]
    public string? website { get; set; }

    [JsonPropertyName("downloadUrl")]
    public string? downloadUrl { get; set; }

    [JsonPropertyName("shortDescription")]
    public string? shortDescription { get; set; }

    [JsonPropertyName("longDescription")]
    public string? longDescription { get; set; }

    [JsonPropertyName("topFreeAlternatives")]
    public string[]? topFreeAlternatives { get; set; }

    [JsonPropertyName("topCommercialAlternatives")]
    public string[]? topCommercialAlternatives { get; set; }

    [JsonPropertyName("proactivanetCategory")]
    public string? proactivanetCategory { get; set; }

    [JsonPropertyName("relatedCpe")]
    public Cpe[]? relatedCpe { get; set; }
}
```

Código 21: Código *SoftwareData.cs*.

A.3.3.- SwInformation.cs

```
[DynamoDBTable("TFG_SwInformation_DB")]
public class SwInformation
{
    [DynamoDBHashKey("softwareName")]
    public string? SoftwareName { get; set; }

    [DynamoDBRangeKey("softwareVersion")]
    public string? SoftwareVersion { get; set; }

    [DynamoDBProperty("softwareInformationDate")]
    public string? SoftwareInformationDate { get; set; }

    [DynamoDBProperty("softwareData")]
    public string? SoftwareData { get; set; }
}
```

Código 22: Código *SwInformation.cs*.

A.3.4.- SwInformationRequest.cs

```
public class SwInformationRequest
{
    [JsonPropertyName("name")]
    public string Name { get; set; } = default!;

    [JsonPropertyName("version")]
    public string Version { get; set; } = default!;

    [JsonPropertyName("responseType")]
    public string ResponseType { get; set; } = default!;
}
```

Código 23: Código *SwInformationRequest.cs*.

A.3.5.- SwInformationResponse.cs

```
public class SwInformationResponse
{
    public string? SoftwareName { get; set; } = default!;
    public string? SoftwareVersion { get; set; } = default!;
    public string? SoftwareInformationDate { get; set; } = default!;
    public SoftwareData? SoftwareData { get; set; } = default!;

    public SwInformationResponse(SwInformation swInformation)
    {
        SoftwareName = swInformation.SoftwareName;
        SoftwareVersion = swInformation.SoftwareVersion;
        SoftwareInformationDate = swInformation.SoftwareInformationDate;
        SoftwareData =
        ↪  JsonSerializer.Deserialize<SoftwareData>(swInformation.SoftwareData);
    }
}
```

Código 24: Código *SwInformationResponse.cs*.

A.3.6.- NvdClient.cs

```
public class NvdClient
{
    private readonly string _apiKey;

    public NvdClient(string apiKey)
    {
        _apiKey = apiKey;
    }

    public Cpe[] GetRelatedCpe(string name, string version, string apiUri)
    {
        using var client = new HttpClient();
        var httpRequest = new HttpRequestMessage(HttpMethod.Get, apiUri);
        client.DefaultRequestHeaders.Authorization = new
        ↪  AuthenticationHeaderValue("Bearer", _apiKey);

        var httpResponse = client.Send(httpRequest);
        var responseStream = httpResponse.Content.ReadAsStream();
        using var reader = new StreamReader(responseStream);
        var httpResponseContent = reader.ReadToEnd();

        NvdResponse nvdResponse;
        try
        {
            nvdResponse = JsonSerializer.Deserialize<NvdResponse>(httpResponseContent);
        }
        catch (JsonException jsonEx)
        {
            Console.Error.WriteLine(jsonEx.Message);
            return Array.Empty<Cpe>();
        }

        var cpeList = new List<Cpe>();
        foreach (Product product in nvdResponse.products)
        {
            string[] cpeNameSplit = product.cpe.cpeName.Split(':');
            var nameCondition = cpeNameSplit[4] == name.Replace(' ', '_');
            var versionCondition = cpeNameSplit[5].Equals(version) ||
            ↪  cpeNameSplit[5].Contains('*') || cpeNameSplit[5].Contains('-');

            if (nameCondition && versionCondition)
            {
                cpeList.Add(product.cpe);
            }
        }
        var cpeArray = cpeList.ToArray();
        return cpeArray;
    }
}
```

Código 25: Código *NvdClient.cs*.

A.3.7.- NvdResponse.cs

```
public class NvdResponse
{
    [JsonPropertyName("products")]
    public List<Product>? products { get; set; }
}
```

Código 26: Código *NvdResponse.cs*.

A.3.8.- Product.cs

```
public class Product
{
    [JsonPropertyName("cpe")]
    public Cpe? cpe { get; set; }
}
```

Código 27: Código *Product.cs*.

A.3.9.- Choices.cs

```
public class Choices
{
    [JsonPropertyName("message")]
    public Message? message { get; set; }
}
```

Código 28: Código *Choices.cs*.

A.3.10.- Message.cs

```
public class Message
{
    [JsonPropertyName("role")]
    public string? role { get; set; }

    [JsonPropertyName("content")]
    public string? content { get; set; }
}
```

Código 29: Código *Message.cs*.

A.3.11.- OpenAIClient.cs

```
public class OpenAIClient
{
    private readonly string _apiKey;
    private readonly string _apiUri;

    public OpenAIClient(string apiKey, string apiUri)
    {
        _apiKey = apiKey;
        _apiUri = apiUri;
    }

    public string Create(string model, string[] messages, string temperature)
    {
        using var client = new HttpClient();
        var openaiRequest = new OpenAIRequest(model, messages, temperature);
        var httpRequest = new HttpRequestMessage(HttpMethod.Post, _apiUri);
        client.DefaultRequestHeaders.Authorization = new
        ↪  AuthenticationHeaderValue("Bearer", _apiKey);
        httpRequest.Content = new StringContent(openaiRequest.content, Encoding.UTF8,
        ↪  "application/json");

        var httpResponse = client.Send(httpRequest);

        var responseStream = httpResponse.Content.ReadAsStream();
        using var reader = new StreamReader(responseStream);
        var httpResponseContent = reader.ReadToEnd();

        var openaiResponse =
        ↪  JsonSerializer.Deserialize<OpenAIResponse>(httpResponseContent);

        return openaiResponse.choices[0].message.content;
    }
}
```

Código 30: Código *OpenAIClient.cs*.

A.3.12.- OpenAIRequest.cs

```
using System.Text.Json.Serialization;

namespace TFG_SwInformation_Updater.Models.OpenAI
{
    public class OpenAIRequest
    {
        public string? model { get; set; }
        public string[]? messages { get; set; }
        public string? temperature { get; set; }
        public string content { get; set; }

        public OpenAIRequest(string model, string[] messages, string
        ↪ temperature)
        {
            this.model = model;
            this.messages = messages;
            this.temperature = temperature;
            this.content = GenerateRequestContent();
        }

        private string GenerateRequestContent()
        {
            string content = $$"""
                {
                    "model": "{{model}}",
                    "response_format": { "type": "json_object" },
                    "messages": [
                        {{messages?[0]}},
                        {{messages?[1]}}
                        ],
                    "temperature": {{temperature}}
                }
                """;
            return content;
        }
    }
}
```

Código 31: Código *OpenAIRequest.cs*.

A.3.13.- OpenAIResponse.cs

```
public class OpenAIResponse
{
    [JsonPropertyName("choices")]
    public Choices[]? choices { get; set; }
}
```

Código 32: Código *OpenAIResponse.cs*.