Facultad de Ingeniería
Universidad de Concepción

UNIVERSIDAD DE CONCEPCIÓN
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA INDUSTRIAL

Optimización en proceso de creación de productos usando Grandes Modelos de Lenguaje (LLM) y agentes IA con aplicación a productos de una empresa de telecomunicaciones.

POR

Rodrigo Santiago Espinoza Bustamante

Memoria de Título presentada a la Facultad de Ingeniería de la Universidad de Concepción
para optar al título profesional de Ingeniero Civil Industrial

Profesor Guía
Juan Carlos Caro Seguel

Profesional Supervisor
César Andrés Zamudio Salazar

Agosto 2025
Concepción (Chile)

A mi familia, novia, amigos y compañeros, que fueron parte de todo este complejo camino que conlleva la universidad.

Agradecimientos

A mi familia por motivarme y promover mi curiosidad en las diversas ramas de la vida, apoyarme en todas las decisiones que me llevaron hasta este momento, además de entregarme todas las herramientas que tenían a su alcance para lograr este objetivo.

A mi madre, que desde pequeños su sueño fue ver a sus hijos profesionales, nos incentivó, motivó y apoyo todo lo que fue posible hasta que consiguió su sueño y sé que será la más feliz al termino de esta etapa.

A mis hermanos, por apoyarme e incentivarme a realizar desafíos difíciles incentivándome a conseguirlos con su apoyo.

A mi novia, que ha sido parte de mi formación como persona y profesional desde que somos pequeños, siempre apoyando, motivándome e incentivando a cumplir cada una de mis metas y sueños.

A mis amigos, que fueron fundamentales a la hora de cortar la rutina, pasar buenos momentos y despejarse de las cargas que entrega día a día la vida.

Al profesor guía Juan Carlos Caro, que supo apoyarme con correcciones en momentos claves, con motivación y consejos para encaminar este proyecto

Finalmente, quisiera agradecer a cada una de las personas que aportaron con un grano de arena para que este proyecto fuera exitoso, los product managers y supervisores del trabajo Luciana, Karina, César y Marcelo.

Gracias por acompañarme en este camino.

Sumario

La generación automática de documentos mediante inteligencia artificial generativa se presenta como una herramienta atractiva para minimizar u optimizar tiempos y recursos en organizaciones. En esta Memoria de Título se desarrolló y validó una solución basada en modelos grandes de lenguaje, que permite automatizar la redacción de documentos técnicos a través de plantillas personalizadas para la organización que contienen "placeholders", es decir, marcadores de posición en la plantilla que posteriormente será rellenado con información obtenida. La solución fue implementada en Python con uso de una arquitectura que integra el framework "Langchain", llamadas a API de Google Gemini y Groq. El proyecto fue probado bajo diferentes configuraciones, utilizando los modelos de lenguaje "Gemini-2.0-flash", "Llama3-70b", "Llama3-8b" y un agente inteligente que utiliza el mismo modelo de lenguaje "Llama3-8b". Cada iteración fue evaluada con métricas de tiempo de ejecución, ahorro estimado en costos de generación, una evaluación humana realizada por expertos en documentación técnica (Product Managers) y relevancia semántica evaluada por el modelo BETO. Los resultados muestran una relevancia semántica similar y aceptable para todos los modelos probados (Promedio BETO > 0,92), sin embargo, se concluye en base al resto de evaluaciones que el agente impulsado con Llama3-8b alcanzó el mayor rendimiento en general con los siguientes resultados, una puntuación de 4,2 sobre 5 por parte de los product managers, una completitud global de la tarea del 88,5% del documento, una disminución de 18 días de trabajo para realizar el documento y generando un ahorro estimado de $131.948 CLP. No obstante, el enfoque del agente presentó el tiempo de ejecución más alto siendo cercano a los 30 minutos (1833 s), en comparación al resto de modelos que requieren menos 10 minutos aproximadamente para ejecutarse. Como recomendaciones, se sugiere agregar otro tipo de mejoras al agente a través de herramientas que permitan realizar personalizaciones con gráficos, tablas o cambios dinámicos de prompt.

Abstract

The automatic generation of documents using generative artificial intelligence is presented as an attractive tool to minimize or optimize time and resources in organizations. In this Thesis Project, a solution based on large language models was developed and validated, allowing the automation of the drafting of technical documents through customized templates for the organization that contain “placeholders,” meaning designated positions in the template that are later filled with retrieved information. The solution was implemented in Python using an architecture that integrates the "Langchain" framework, and calls to the Google Gemini and Groq APIs. The project was tested under different configurations, using the language models "Gemini-2.0-flash," "Llama3-70b," "Llama3-8b," and an intelligent agent that uses the same language model "Llama3-8b." Each iteration was evaluated with metrics of execution time, estimated cost savings in document generation, a human evaluation carried out by experts in technical documentation (Product Managers), and semantic relevance evaluated by the BETO model. The results show similar and acceptable semantic relevance for all the models tested (BETO > 0.92), however, it is concluded based on the other evaluations that the agent powered by Llama3-8b achieved the highest overall performance with the following results: a score of 4.2 out of 5 from the Product Managers, 88.5% document completeness, a reduction of 18 working days to complete the document, and an estimated savings of $131,948 CLP. Nevertheless, the agent approach presented the highest execution time, being close to 30 minutes (1833 s), compared to the rest of the models that require approximately less than 10 minutes to execute. As recommendations, it is suggested to add other types of improvements to the agent through tools that allow customizations with charts, tables, or dynamic prompt changes.

Tabla de Contenido

Lista de Tablas

Lista de figuras

Introducción

La generación de documentación técnica es un proceso primordial, pero a menudo laborioso dentro de las organizaciones. En este contexto, se presenta una empresa de telecomunicaciones chilena con más de 40 años de trayectoria, la necesidad real de producir documentación completa para la operación de los productos o servicios, auditorías externas y posibles cambios en la estructura organizacional vuelve esta tarea indispensable y de alta relevancia para la organización.

Tradicionalmente, esta tarea requiere una inversión significativa de recursos humanos por parte de los encargados de crear y gestionar productos, desde ahora "product managers" y equipos técnicos a cargo de realizar entregas con información detallada de cada producto que la organización planea introducir al mercado. Con las tecnologías que actualmente están en constante evolución, este proceso puede disminuir en el uso del tiempo necesario para generar los documentos con información necesaria para la operación.

En la actualidad, el procesamiento de lenguaje natural (PLN) está avanzando para lograr entendimientos buenos en diferentes contextos, incluyendo la lectura y creación de documentos incluso en el desarrollo de artículos científicos de ingeniería (Ramírez, 2024). El PLN es ampliamente usado en modelos que utilizan inteligencia artificial para lograr entender el lenguaje humano en instrucciones que le sean otorgadas. Además, el PLN es integrado en técnicas de sistemas que realizan tareas que requieren razonamiento para resolver problemas presentados por el usuario en este caso los "agentes", existen diversas bases de trabajo o frameworks que incluyen librerías para facilitar el uso de agentes en diversos ámbitos de los proyectos.

Smolagents, una librería de Python desarrollada por HuggingFace, facilita el uso de agentes de IA capaces de interactuar con diferentes herramientas a través de la ejecución de código Python (HuggingFace, 2025). También el mismo framework ofrece herramientas desarrolladas que pueden ser utilizadas con un sencillo llamado a dicha librería. Un punto clave es la utilización del framework HuggingFace o alguno similar durante la realización del estudio dada la versatilidad que entrega para proyectos de producto mínimo viable.

Otro framework importante a considerar durante el estudio es Langchain, ofreciendo herramientas pre creadas para utilizar con agentes, destacando la versatilidad de conexiones con API de OpenAI, Groq, grandes modelos de lenguaje locales, etc. (Langchain, 2025).

Por otro lado, la conexión API de Gemini de Google ofrece capacidades avanzadas en el

procesamiento del lenguaje natural y la generación de contenido a alta velocidad, además de contar con usos gratuitos sujeto a algunas restricciones (Google, 2025). Se destaca principalmente esta API porque puede aportar significativamente a validar funcionamiento inicial de generación de documentación en base a inteligencia artificial.

Groq entrega una conexión API gratuita para pruebas o usos no intensivos, con esta herramienta se utilizan grandes modelos de lenguaje con una generación suficientemente rápida para tareas diarias. Esta herramienta que otorga es muy relevante, ya que al disponer de llamadas a modelos de lenguaje que están precargados pueden ser utilizados para hacer pruebas de rendimiento o utilizar diferentes modelos que sean llamativos a la hora de desarrollar algún agente o utilizarlo para la generación de texto.

Esta propuesta se basa en cómo la combinación de frameworks de agentes que utilicen modelos grandes de lenguaje (LLM), el uso de llamadas a API con un LLM y un pipeline de Python puede optimizar o mejorar en términos de tiempo el proceso de creación de documentación técnica aportando documentos desarrollados por las tecnologías mencionadas para los product managers de la organización. La investigación se centrará en el desarrollo de un script en Python que aproveche estas tecnologías para generar documentación necesaria a la hora de crear productos, realizando mejoras respecto al proceso actual con la creación de documentos en base a inteligencia artificial.

1.1 Objetivos

1.2.1. Objetivo General

Optimizar el proceso de creación de documentación al disminuir el tiempo empleado utilizando un script de Python que integre herramientas de inteligencia artificial generativa para generar y validar documentos, otorgando feedback al product manager en la creación de soluciones de la empresa de telecomunicaciones.

1.2.2. Objetivos Específicos

- Investigar y comprender las capacidades de la API de IA, comprender el uso de los grandes modelos de lenguaje y los frameworks existentes para la generación de documentos y la automatización de tareas.
- Identificar los tipos de documentación técnica más relevantes y demandados por los product managers de la organización en sus procesos de trabajo.

- Diseñar la arquitectura y la lógica del script en Python que integre la API de inteligencia artificial generativa, los LLM y la librería que permita crear agentes para la creación automatizada de la documentación técnica identificada.
- Desarrollar el script que disminuya los tiempos necesarios a la hora de generar documentos completos en formato y contenido.
- Diferenciar LLM a través de una evaluación obtenida con las pruebas realizadas.
- Evaluar la confiabilidad, precisión y eficacia del script con los resultados entregados.
- Evaluar mejora del proceso de creación de productos.

1.2 **Justificación del Tema**

El tema central del estudio cobra real importancia por diversas razones. En primer lugar, responde directamente a uno de los KPI con el que miden el área durante el año 2025, este indicador solicita mejorar en un 15% el tiempo empleado para liberar una solución con respecto al año 2024. Este indicador motiva al área a introducir optimizaciones o mejoras en procesos claves del flujo correspondiente a la creación de productos, entre ellos la generación de documentación técnica.

Adicionalmente, se identifica un problema persistente en la organización que parte directamente con la documentación inexistente de productos y servicios que ya han sido liberado en años anteriores a la existencia del área actual. El bajo tiempo disponible de los product managers para crear dicha documentación agrava la situación, la operación en su mayoría funciona sin problemas, aun cuando la información es inexistente, siendo un ejemplo claro que el conocimiento se encuentra en las personas y no en la organización. Esto abre una ventana de mejora en el proceso que es documentar esta información a través técnicas lo suficientemente automatizadas para generar documentación de calidad para la operación correcta de los productos o servicios.

En términos de beneficios operativos, se anticipa una disminución de carga de horas empleadas en el procedimiento inicial de las soluciones y de generación de documentos inexistentes. Al obtener una eficiencia en el proceso existiría una mejora directamente en el indicador con el que miden el área. A nivel organizacional, esta eficiencia en los tiempos de liberación puede traducirse en plazos menores para liberar productos, generando impactos positivos en ingresos, esto ayudando a distintos indicadores de las diversas líneas de negocios con las que cuenta la empresa, indirectamente ayudando

a sus metas y rentabilidad operativa.

Respecto a beneficios académicos, se abre una nueva línea de investigación de aplicación de generación de texto sumado a agentes validadores de la información que se está entregando, validando empíricamente la aplicabilidad de dichas tecnologías en este proceso de creación de soluciones. Actualmente, existen diversos trabajos realizados para generar documentos de forma automatizada, utilizando modelos de lenguaje o técnicas de placeholders con información de una base de datos para rellenar documentos tipo que estructuralmente son iguales pero varían en pocos parámetros, a diferencia de estos estudios, este proyecto se basa en una combinación de tecnologías que se utilizan al día de hoy para entregar un documento estructurado con formato empresarial, con información que puede ser compleja y a la vez completamente distinta un documento de otro.

Este proyecto busca aportar principalmente para mejorar los fallos identificados en el proceso de la organización, a pesar de ello, este procedimiento puede aplicarse a distintas organizaciones con características similares a las presentadas en este estudio. Adicionalmente, se busca aportar con ejemplos e iteraciones de buenas prácticas a la hora de trabajar con grandes modelos de lenguaje (LLM) en procesos empresariales contribuyendo con metodologías que puedan ser aplicables en otros contextos distintos al actual.

El usuario que se estará beneficiando de este proyecto es la organización en general por las diversas métricas a las que, directa o indirectamente, se estarán afectando debido al aplicar el estudio realizado. Pese a eso, el usuario final de este proyecto son los product managers que podrán ejecutar el script para obtener documentos generados ahorrando tiempo en generar formato empresarial y en entender el producto o servicio a generar. Por otra parte, las tecnologías a utilizar cuentan con años de estudio aplicado en artículos y proyectos, siendo técnicamente viable aplicarlas en el contexto dado.

Revisión de literatura

Las nuevas tecnologías están revolucionando la manera de resolver problemas en la vida diaria de las personas. Por esta razón, la toma de decisiones en organizaciones tiende cada vez más a la integración de un gran volumen de datos, una correcta gestión de la información e incluir herramientas innovadoras que facilitan procesos integrando técnicas de inteligencia artificial.

Durante los últimos años, el alcance de las herramientas que integran el uso de inteligencia artificial generativa va en aumento. Lo que se menciona en (Babic et al, 2020) es un gran ejemplo de esto último, ya que según el artículo una gran cantidad de personas cree que estas herramientas son útiles para aumentar la productividad en los lugares de trabajo. Sin embargo, existe incertidumbre por el posible reemplazo de las máquinas o lenguajes de inteligencia artificial en los puestos de trabajo.

Artículos abordan la problemática planteada sobre la incertidumbre de la integración de inteligencia artificial en las organizaciones y en la relación máquina-humano. Esta incertidumbre es presentada por el autor en (Abbass, 2019) llegando a la conclusión de que más que un reemplazo será una herramienta que necesitará tarde o temprano de alguna instrucción humana en algún bucle para la toma de decisiones que influirá en el proceso de inteligencia artificial.

Por otra parte, la documentación a menudo suele ser bastante laboriosa ya que requiere de una gran cantidad de inversión de tiempo para lograr completar documentos con formatos y contenido requerido. En el trabajo realizado por (Lin & Cheng, 2024) se muestra una aplicación práctica para el problema mencionado, aplican el uso de un LLM entrenado con diversas fuentes de documentación legal, obteniendo un lenguaje pre entrenado que pueda generar diversos documentos legales (contratos, acuerdos legales, documentos judiciales, etc.), con el objetivo de ahorrar tiempo al solo necesitar revisión y adaptación a cada caso por parte del profesional a cargo de realizar estos documentos.

El procesamiento de lenguaje natural (PLN) es una herramienta que se utiliza en diferentes soluciones entregadas para facilitar tareas repetitivas. Un ejemplo de lo anterior es presentado por el autor (da Cunha, 2022), al utilizar PLN para realizar un redactor asistido de textos administrativos a lenguaje más sencillo y ayudar en el entendimiento. Lo desarrollado por los autores se basa en un estudio de las técnicas utilizadas de PLN, posteriormente crear algoritmos que realicen una mejora del texto haciéndolo más claro con estructuras predefinidas de títulos, sugerencias y ejemplos para lograr el objetivo.

Otro ejemplo práctico se presenta en el área de búsqueda de talento. Los autores (Chango & Varela, 2021) utilizan procesamiento de lenguaje natural, combinado con automatización de robótica de procesos. Lo anterior, con el objetivo de recopilar la información de manera no estructurada y posteriormente puntuar estos currículos con criterios valorativos, ahorrando tiempo y revisando todos los candidatos a los puestos de trabajo.

La inteligencia artificial generativa puede ser una herramienta útil para crear textos que sean lo suficientemente personalizados según la base de datos que contenga para la generación. En particular, existe un artículo que estudia la inteligencia artificial generativa de Google, titulado "Probando Bard: así funciona la Inteligencia Artificial Generativa de Google" realizado por los autores (Lopezosa & Codina, 2023) se estudia la efectividad del modelo Bard de Google (posteriormente Gemini) probando diversidad de prompts y analizando las respuestas. Notando que durante el año de la publicación se encontraba junto con OpenAI en el lanzamiento reciente de las herramientas y estaban en periodo inicial de entrenamiento con los datos proporcionados por los usuarios.

Así mismo, (Casar Corredera, 2023) explica que la inteligencia artificial data cerca de los años 50 en el siglo XX, ahora bien, últimamente ha tenido un crecimiento exponencial por la cercanía de los lenguajes como lo es ChatGPT o Bard (el chat de inteligencia artificial de Google). Lo interesante que menciona este autor es la variedad de aplicaciones que se les puede dar a estas nuevas tecnologías, menciona por ejemplo los chatbots, la creación de imágenes, en diagnósticos médicos, educación, entre otros. En este sentido, existe una enorme cantidad de aplicaciones que se puede integrar con la inteligencia artificial generativa utilizando procesamiento de lenguaje natural.

Por otra parte, en la actualidad los agentes autónomos utilizados para resolver tareas han mostrado ser una herramienta útil en diversos ámbitos. Un agente alimentado por un modelo grande de lenguaje puede lograr buenos resultados en tareas que son complejas o repetitivas. Los autores (Xu et al., 2025) presentan un interesante estudio, utilizan un sistema de agentes alimentados con un LLM que son aplicados a tareas del mundo cotidiano, utilizando los agentes impulsados con lenguajes obtenidos con conexiones a API y otras técnicas, obteniendo que el agente que es más competitivo puede completar el 30% de las tareas de manera autónoma. El resultado obtenido por los autores señala una mejoría en estas tareas, pero también entrega las posibilidades de integrar un agente en tareas simples que pueden llegar a ser automatizadas, de esta forma indican que con las tecnologías actuales las tareas de mayor complejidad escapan del alcance de estas soluciones.

Los agentes alimentados con un modelo grande de lenguaje despiertan el interés de muchas

investigaciones ya que con la evolución de los modelos durante los últimos años se han encendido muchas áreas que pueden beneficiarse del uso de agentes autónomos que realicen tareas con un nivel mayor de razonamiento al poder tomar decisiones, en (Cheng et al., 2024) los autores presentan diversos métodos para aplicar el uso de agentes, explican una clasificación de agentes, pero indican que la característica más única se debe a la posibilidad de tomar decisiones durante su ejecución y mejorar los resultados con el feedback que obtiene al realizar una tarea.

Una aplicación similar se desarrolló por los autores en (Huang et al., 2023), quienes crearon agentes que tuvieran herramientas para leer/escribir archivos, ejecutar código e inspección de resultados, al poder ejecutar dichas acciones se podían generar investigaciones por parte de los agentes. Los resultados que obtuvieron los autores presentaban una variación en las tasas de éxito establecidas, variando desde un 0% hasta un 90% según la existencia de información previa, lo que necesariamente generó dudas y oportunidades de mejora en un futuro sobre la planificación a largo plazo y alucinación del agente.

Marco teórico

3.1 Generación de documentos

La automatización de documentos es ampliamente utilizada en el ámbito legal, empresarial y financiero para realizar documentos tipo reemplazando valores iniciales para personalizar y adecuar los documentos. Ejemplo de lo anterior, es presentado por (Brooks et al, 2020) al destacar que la automatización de documentos ha sido una de las tecnologías fundamentales que han tenido, ya que ayuda a disminuir el tiempo que se tarda en generar documentos legales, mejorando los tiempos y cantidad de documentos generados, sin afectar a la calidad de estos.

En el entorno empresarial, la generación automatizada de documentos ha permitido bajar tiempos de desarrollo de contratos para trabajadores al reemplazar los datos necesarios para crear un nuevo documento que cumple los estándares y mantiene las mismas normas de los puestos de trabajo ya creados. Con todo ello, este tipo de generación de documentos se guía por un formato que solo cambia en un par de parámetros según el usuario requiera, automatizando la generación, pero limitando por completo la creación de nuevo contenido en base a los inputs que se reciban.

3.2 Inteligencia artificial

La inteligencia artificial se puede entender como las técnicas de aprendizaje automático (Machine learning), aprendizaje profundo (Deeplearning), Procesamiento de Lenguaje Natural (PLN) y visión por computadora (Getchell et al, 2022). El mismo artículo describe estas tecnologías con capacidades de realizar tareas que generalmente requieren de una acción humana, como la toma de decisiones, la traducción de idiomas, etc.

Otra forma de comprender la inteligencia artificial es como una capacidad que tiene un sistema para lograr objetivos a través de una sucesión de acciones que realiza interpretando datos externos (Caldeiro, 2024). En el mismo artículo se mencionan características y ramas que se relacionan con la inteligencia artificial, por ejemplo, la inteligencia artificial generativa, que busca generar contenido nuevo usando las mismas técnicas de basarse en datos y algoritmos sofisticados.

En otras palabras, las definiciones para inteligencia artificial tienen muchas variaciones y formas de entenderlas, aunque todas intentan apoyar en tareas que necesiten instrucciones, tomas de decisiones y generar contenido, ya sea completamente nuevo o con datos otorgados para realizar las respuestas necesarias a las instrucciones otorgadas.

3.2.1 **Inteligencia artificial generativa**

Como se mencionó anteriormente, la inteligencia artificial generativa es una rama de la inteligencia artificial que busca generar contenido completamente nuevo y original a partir de datos existentes y técnicas como el Procesamiento de Lenguaje Natural (PLN), algoritmos de aprendizaje automático, entre otras.

El autor (Casar Corredera, 2023), en su revisión y explicación sencilla de estos temas menciona a la inteligencia artificial generativa como el conjunto de métodos y aplicaciones capaces de generar contenido como texto, imágenes, software, etc. Con características indistinguibles de las que podría hacer un humano. En resumen, se trata de un subconjunto de la inteligencia artificial que tiene como objetivo crear contenido de diverso ámbito recibiendo una solicitud y dejando actuar diferentes técnicas de algoritmos para producir algo coherente que sea similar o indistinguible al trabajo de un humano.

3.2.2 **Procesamiento de lenguaje natural (PLN)**

Se entiende el lenguaje natural como la forma que evoluciona el idioma, es decir no necesariamente utilizando las estrictas reglas del lenguaje formal, sino la forma cotidiana del uso del lenguaje, de esta forma se describe el lenguaje natural en el artículo de los autores (Cortez Vásquez, 2009). Además, se describe que el procesamiento del lenguaje natural corresponde a la forma de comunicarnos con la computadora utilizando el lenguaje cotidiano. En este mismo artículo, se describe la arquitectura del PLN, incluyendo aplicaciones, ventajas y desventajas de utilizar PLN a la hora de comunicarse con una computadora. No obstante, para los fines de este estudio, la definición que entrega el autor es suficiente para el alcance que tiene este proyecto utilizando PLN.

3.2.3 **Grandes modelos de lenguaje (LLM: Large Language Model)**

Los grandes modelos de lenguaje o LLM por sus siglas en inglés son modelos que contienen una gran cantidad datos para generar y procesar requerimientos presentados, los LLM utilizan técnicas de deeplearning para mejorar el procesamiento natural (Rodríguez Fernández et al, 2024). En otras palabras, los LLM son modelos de lenguaje que procesan requerimientos utilizando como base de datos una gran cantidad de datos que previamente les han sido otorgados a través de un entrenamiento, ejemplos de aquello son los lenguajes GPT-4, Gemini, LlaMa, etc.

El preentrenamiento es relevante para los LLM, ya que se puede entrenar para predicción

autorregresiva, un modelado de lenguaje enmascarado, etc. El entrenamiento se logra a través de modelado semántico, probabilidades de texto contextualizado a partir de grandes volúmenes de datos (Yao, Y. et al, 2024). El artículo también menciona que un LLM debe ser capaz de 4 puntos claves: Comprender profundamente el lenguaje natural, debe ser capaz de generar texto similar a un humano, conciencia contextual y seguir instrucciones fuertemente.

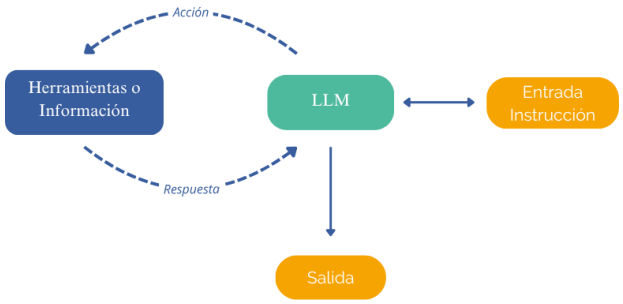
3.3 Agentes de IA

Una nueva forma de utilizar la inteligencia artificial son las técnicas de agentes autónomos que usan un modelo grande de lenguaje, la diferencia que existe con la inteligencia artificial es la mínima necesidad de intervención humana, utilizando una mayor adaptabilidad y capacidades de toma de decisiones (Acharya et al, 2025). Es decir, utilizan iteraciones incluyendo toma de decisiones para lograr los objetivos planteados, evaluando los resultados y entregando mejores respuestas que las técnicas utilizadas en los chatbots con IA.

Un agente de inteligencia artificial es muy útil a la hora de realizar tareas que son muy complejas y requieren de pasos intermedios para entregar una solución final. Según el artículo (Rovane, C., 2004) presenta sus características como dos suposiciones claves, una corresponde las varias opciones disponibles que elegir, la segunda es el control que debe tener a la hora de decidir que será ejecutado.

Los agentes en sencillas palabras son LLM que se les entregan herramientas personalizadas que le otorgan diferentes capacidades a estos lenguajes para que tomen decisiones y obtengan respuestas muy buenas sin grandes intervenciones en las instrucciones otorgadas. Una representación gráfica de un flujo de trabajo simplificada de un agente se presenta a continuación.

Figura 3.1: Flujo de trabajo agente genérico.



Fuente: Elaboración propia

3.3.1 Herramientas de agentes

Los agentes son impulsados con modelos grandes de lenguaje entrenados con muchos parámetros, a

pesar de ello, al utilizarlos se vuelven agentes que responden cosas sin sentido, ya que su base de información no tiene nada más que los parámetros entrenados en el LLM con el que se alimenta. No obstante, las herramientas otorgadas a un modelo grande de lenguaje o agente sirven para mejorar en eficiencia y precisión de los LLM en tareas específicas y complejas (Shen, Z., 2024). Según el mismo autor la utilización de herramientas externas sirve para elevar las capacidades de los agentes más allá de las bases que tienen con el modelo grande de lenguaje que utiliza.

Es importante comprender que las herramientas son una conexión que entrega una capacidad extra al agente, es decir le otorgan nuevas posibles tareas que pueden ser abordadas de diversas maneras, lo que aporta a una selección de herramientas según sea el caso y el ajuste dinámico del plan con las iteraciones del agente (Shen, Z., 2024).

Metodología

En este capítulo se presentará la metodología utilizada en el desarrollo del estudio, iniciando por la descripción de la problemática hasta la arquitectura y evaluación de la solución implementada.

4.1 Investigación flujo de proceso

El primer paso del proyecto será investigar características de las tareas que se realizan durante la creación de una nueva solución, obtener métricas existentes y KPI del área donde se está realizando el proyecto.

El levantamiento de información sobre el área se realizará con el director encargado de desarrollar las soluciones, a través de reuniones presenciales. Se obtendrán las métricas y se resolverá una serie de preguntas a través de las cuales se obtendrá la suficiente información del proceso.

Además, se establecerá el alcance del proyecto a desarrollar en objetivos y en tiempo. Para el levantamiento de información durante la reunión se almacenará la información relevante de los siguientes puntos:

- KPI del área: Una métrica que es muy útil posterior a la liberación de este proyecto es la comparación de un indicador clave de rendimiento para el área de desarrollo de soluciones de la empresa, por esta razón la entrevista con el director encargado será clave a la hora de evaluar la solución creada respecto al KPI que se tenga en el área.
- Tiempos y costos del desarrollo: Otra métrica interesante que se debe tener en cuenta es el tiempo que se tarda el área correspondiente en generar un documento completo para un nuevo producto, levantar si siempre es el mismo tiempo o puede variar según la dificultad de creación.
- Áreas participantes: La composición de la organización a la hora de realizar procesos es de alta relevancia, ya que revisando la comunicación entre las áreas puede ser clave y una oportunidad de mejora en el proceso. Por este motivo, es de suma relevancia levantar la información de las áreas que están ligadas directamente a la generación de un producto.
- Proyección del área: En la entrevista se debe obtener la información de como se ve el futuro del área ligada a los pilares organizacionales, es decir, cual es la proyección que se tiene del área para los próximos años, si se quieren implementar mejoras de procesos, disminución de tiempo, etc.
- Entradas y salidas del proceso: Para conocer a fondo el flujo de generación de un nuevo

producto/servicio es importante conocer cuál es el input que produce la cadena de actividades que deben desarrollarse y cuál es el output que se debe generar para dar por concluido el proceso.

4.2 **Elección de framework**

Las técnicas para proyectos con similares características están basadas normalmente en programación con lenguaje Python, por esta razón existen diversas librerías pertenecientes a distintos desarrolladores de código abierto que proporcionan una base de trabajo que se dispone para la comunidad y puede seguir creciendo junto con ella.

Se realizará un estudio sobre las bases de trabajo o "frameworks" que actualmente se utilizan para desarrollar herramientas utilizando inteligencia artificial generativa o modelos grandes de lenguaje en la literatura, de esta forma realizar un ranking por atributo detallando los beneficios y oportunidades que puedan generarse al utilizar cada uno. Para el objetivo que se quiere cumplir se seleccionó el framework de "LangChain".

Para la elección del framework se priorizaron las siguientes aristas:

- Conexión con API's: Capacidad para conectarse a servicios de proveedores como Groq, OpenAI, Google, HuggingFace, etc.
- Documentación: Claridad y cantidad de material disponible para crear utilizando el framework, ejemplos de código, etc.
- Herramientas y comunidad: Llamado sencillo a herramientas, herramientas existentes creadas por la comunidad, etc.
- Facilidad de uso: Curva de aprendizaje para implementar uso de framework en el tiempo de desarrollo de este proyecto.

Como anteriormente se mencionó la elección es el framework de LangChain, pues al ser de código abierto tiene un gran potencial para ser utilizado en el proyecto, ya que contiene integraciones generales con diversos sistemas, incluso con HuggingFace y CrewAI (otros frameworks que podrían reemplazar a langchain durante el desarrollo del proyecto), siendo posible trabajar con un abanico de opciones para crear un generador de documentos utilizando grandes modelos de lenguaje.

Cabe destacar que Smolagents de HuggingFace tiene una sintaxis que es muy sencilla de trabajar, ya que con unas pocas líneas de código se pueden obtener agentes funcionales y con una capacidad

increíble, aunque, al ser una librería emergente y bastante reciente se utilizará como integración de LangChain para casos como prueba de modelos precargados en HuggingFace.

4.3 **Elección Modelo Grande de Lenguaje**

Actualmente, existe una gran cantidad de modelos preentrenados de lenguaje que suelen ser muy potentes por la cantidad de parámetros utilizados en el entrenamiento, no obstante lo anterior, se han estudiado a través de pruebas evaluadas que los modelos "mini", es decir, modelos que tienen una cantidad menor de parámetros utilizados durante su entrenamiento, pueden obtener buenos resultados en una diversa cantidad de tareas, lo que implica que un modelo que tenga mayor cantidad de parámetros almacenados no necesariamente es mejor que uno bien optimizado o mejor entrenado. Un ejemplo de lo anterior es presentado en la comparación de Mistral 7B en (Jiang et al., 2025), donde se obtienen resultados mejores en diversas tareas en comparación con otros lenguajes que cuentan con una mayor cantidad de parámetros. Por esta razón, se debe utilizar un criterio de elección de modelo de lenguaje, que sea gratuito, bueno en generación de texto y fiable.

Para la elección del LLM se buscaron comparaciones de los modelos con tareas de generación de texto, además se realizaron pruebas empíricas al ejecutar el programa y medir los tiempos que se tardaba en el proceso de creación de documentos. El principal enfoque para evaluar y utilizar los grandes modelos de lenguaje fue su disponibilidad gratuita o ser de código abierto para realizar estas pruebas. Los modelos evaluados fueron los siguientes:

- Gemini-2.0-flash: Modelo facilitado con API de Google para pruebas, corresponde a IA generativa
- Llama3-8B: Modelo LLM utilizado con la API de Groq para pruebas no intensivas
- Llama3-70B: Al igual que el modelo con menos parámetros, se utiliza la llamada a la API de Groq para obtener este modelo de forma precargada.

El criterio de elección de estos modelos fue principalmente su disponibilidad gratuita para pruebas otorgado por plataformas como Google o Groq, y las herramientas claves entregadas por las comunidades de cada plataforma.

Por otra parte, el costo asociado a estos tres modelos durante el proyecto fue gratuito, aun así, para un posible escalamiento o pago de estas API se presenta el costo asociado y el cálculo que se realizará para mostrar los posibles costos asociados a generar los documentos

Tabla 4.1: Costos de uso LLM.

Modelo	Entrada (1M tokens)	Salida (1M tokens)
Gemini 2.0 Flash	$0.10 USD	$0.40 USD
LLaMA 3 – 70B (Groq)	$0.59 USD	$0.79 USD
LlaMA 3 – 8B (Groq)	$0.05 USD	$0.08 USD

Fuente: Elaboración propia (Google, 2025) y (Groq, 2025)

Para calcular el costo asociado a los grandes modelos de lenguaje se realizará de la siguiente forma:

$$G_i = T_{in} \cdot \frac{C_{inp,i}}{1M} + T_{out} \cdot \frac{C_{out,i}}{1M}$$

Donde:

$G_i =$ Costo en USD por uso de LLM i

$T_{in} =$ Tokens de entrada del LLM.

$T_{out} =$ Tokens de salida del LLM.

$C_{out,i} =$ Costo por millón de tokens de salida para el modelo i.

$C_{inp,i} =$ Costo por millón de tokens de entrada para modelo i.

$$i = \begin{cases} 1 \; si \; el \; modelo \; es \; Gemini - 2.0 - flash \\ 2 \; si \; el \; modelo \; es \; Llama3 - 70b \\ 3 \; si \; el \; modelo \; es \; Llama3 - 8b \end{cases}$$

Notar que G_i está en dólares, para realizar la conversión se utilizará un tipo de cambio de 1 USD = 961,80 CLP.

4.4 Creación de plantillas

Uno de los puntos de mejora de la solución es entregar un producto avanzado utilizando los requerimientos profesionales de la empresa. Por este motivo, posterior a elección del modelo se crearán plantillas personalizadas para cada documento generado y que posteriormente será completado con la información obtenida a través del modelo de lenguaje grande del cual se mencionó anteriormente.

La creación de plantillas será manual, inicialmente para que los datos confidenciales de la empresa y material gráfico que se incluya se realicen de manera local en la máquina donde se ejecutará el programa final. Para realizar la inserción de información a las plantillas creadas se utilizarán

placeholders o marcadores de posición, el propósito de estos marcadores será localizar de una forma óptima la posición que debe ocupar la información otorgada por el LLM anteriormente.

La forma de utilizar los marcadores de posición serán con un simple formato JSON, donde la llave estará marcada en las plantillas y las respuestas generadas por el modelo grande de lenguaje otorgará el complemento que posteriormente tendrá el documento en lugar de la llave que inicialmente tenía. Las librerías que se utilizan durante el trabajo con Python es Docx y Docxtpl.

4.5 **Creación del script**

La creación del script de Python irá definiendo la estructura y lógica siendo consultada en el repositorio facilitado por Langchain, dado que entrega la información necesaria para utilizar modelos grandes de lenguaje y creación de agentes.

La metodología utilizada para crear el script fue metodología ágil, iterando semanalmente durante las reuniones con el equipo involucrado, iniciando por un programa sencillo que utilizaba la comunicación con una API de Google Gemini y posteriormente fue evolucionando a trabajos locales, plantillas predefinidas, entre otras mejoras implementadas.

Para la generación inicial de los documentos se utilizarán funciones de Python que seguirán la siguiente lógica:

- Extracción de texto: El primer paso que tendrá que realizar el script desarrollado es ingresar a la base de datos entregada por el usuario y extraer el texto que allí se encuentre para almacenarlo de forma temporal para la ejecución del código.
- Envío de prompts: Posterior a la extracción de texto se debe realizar un envío de un prompt al LLM, la lógica de este prompt será entregarle contexto del producto o servicio el cual se está generando y realizar preguntas claves para la generación de cada sección que se requiere en el documento. En este punto el flujo de trabajo variará para incluir el agente que pueda redactar la información, de esta forma se incluirá un agente con herramienta de validación de texto que primero generará y luego validará lo que está generando en el documento.
- Conexión API: Para los modelos grandes procesados de manera rápida se utilizará la API de Groq con pruebas de diversos modelos ofrecidos gratuitamente, de esta forma el script tendrá conexión con el LLM precargado.
- Generar primer documento: Posterior que el LLM haya realizado el trabajo correspondiente

de obtener los párrafos que se necesiten para generar los ítems se creará un primer documento que incluirá toda la información generada anteriormente.

- Validación utilizando BETO: Para conocer la relevancia del texto se implementará una validación cuantitativa con la variación del lenguaje BERT entrenado con parámetros en español llamado BETO, con el motivo de saber si el texto que se generó tiene relevancia semántica con el contexto y necesidades de cada caso.
- Generación de documentos: El último proceso del script será posterior a la validación, generar los documentos enviando la información desde el documento generado inicialmente a las plantillas de los documentos que finalmente serán generados, marcando así el punto final del flujo.

Las librerías necesarias para realizar este modelo son:

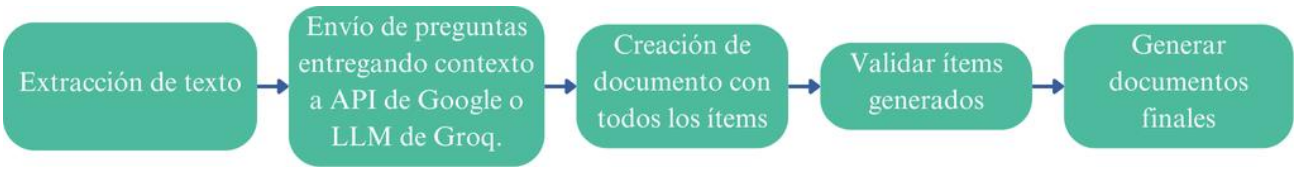
- Langchain: Este framework ofrece una gran cantidad de librerías útiles que se utilizarán para, herramientas claves para los agentes, obtener conexiones con API que faciliten el uso de modelos de lenguaje, etc.
- Python-docx: Manipulación de documentos Word con Python.
- Huggingfacehub: Para utilizar el token de acceso y conseguir diversidad de beneficios trabajando con esta comunidad de inteligencia artificial.
- Smolagents: Framework utilizado para generación de agentes que requieren pocas líneas de código.

Las especificaciones técnicas del equipo que se está utilizando para realizar este proceso de manera local son:

- CPU: AMD Ryzen 7 PRO 4750U 1.70 GHz
- GPU: AMD Radeon(TM) Graphics 496MB
- RAM: 16 GB
- Versión de Python 3.11.9
- Sistema operativo: Windows 11 Enterprise
- Tipo de sistema: 64 bits, procesador basado en x64

Un resumen del esquema que se estará utilizando en el caso de emplear la API de Google Gemini o API de Groq para acceder directamente a modelos grandes de lenguaje es extraer el texto, crear esquemas con respuestas, crear documento con todos los ítems y generar los documentos finales en base al documento inicial.

Figura 4.1: Resumen flujo de script sin agente.



Fuente: Elaboración propia.

4.6 **Creación de agente**

Para la generación el agente se debe definir en que parte actúa y las herramientas que tendrá a disposición para interceder en el desarrollo del script.

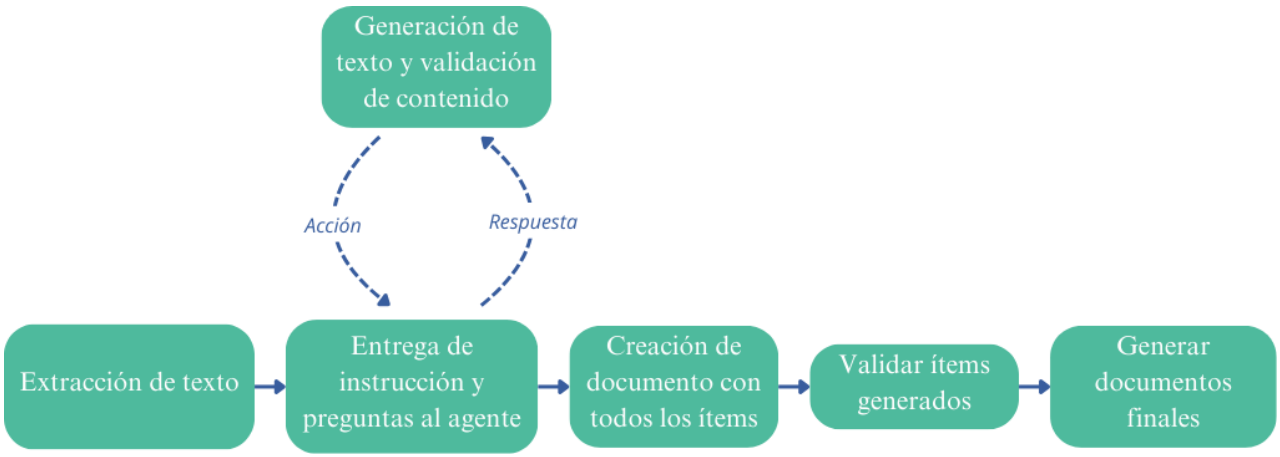
Como anteriormente se mencionó, el agente tendrá la tarea de validar y eventualmente evaluar o comentar situaciones especiales, para este proceso se le otorgarán herramientas que permitan realizar las tareas necesarias, dichas herramientas se detallan a continuación:

- Extracción de texto: El agente podrá acceder al documento creado primeramente con el LLM y almacenarlo temporalmente mientras realiza el flujo de trabajo que necesite, es decir siempre que valide el documento. Esta extracción será la función que servirá para extraer texto y concatenar para entregarle el contexto al agente. La herramienta que se le otorgará al agente será una consulta a esta extracción para cada vez que requiera responder una pregunta.
- Evaluación de relevancia: Tendrá una herramienta que pueda predecir las palabras que se necesitan en algunas frases según el contexto que pueda ir obteniendo en el documento, estas palabras que intenta predecir son Mask que al entregarle contexto intenta predecir con cierta cantidad de confianza, si pasa esta prueba el contexto estará bien. Este proceso lo hace usando el lenguaje BETO.

El esquema para el agente cambia respecto al proceso que se presentó en el script con API's de servicios externos sin implementar un agente, para este caso se mantiene la extracción de texto desde la base de datos, aunque ocurre un cambio, al agente se le asigna una tarea y comenzará a trabajar iterando para obtener una respuesta lo suficientemente satisfactoria antes de enviarla como output de

su proceso, posteriormente a esto se mantiene el flujo tal cual para validar y generar los documentos finales.

Figura 4.2: Resumen flujo agente.



Fuente: Elaboración propia.

El detalle de los scripts se encuentra en los anexos "**Anexo 1: Código estrategia para iteración con LLM de Groq Llama3 70b y 8b**", "**Anexo 2: Código estrategia para iteración con API de Google Gemini.**" y "**Anexo 3: Código estrategia para iteración con agente y LLM de Groq.**".

4.7 Evaluación de resultados

Una parte importante para el proyecto se basa en cómo se evaluarán los resultados, para la evaluación de la calidad de los resultados se utilizarán dos metodologías, una que será evaluación por parte del equipo de soluciones y otra será una automatizada utilizando técnicas de inteligencia artificial para evaluar posibles errores o incoherencias creadas durante la ejecución del script con todas las herramientas. Esta metodología está basada en la evaluación realizada por (López Gomez, 2025), el autor detalla su manera de evaluar siendo muy similar al caso del proyecto presente.

Equipo de soluciones: Para este caso se utilizará una evaluación de puntajes asignados a diferentes criterios relevantes en la entrega de un documento:

- Redacción: Evalúa la calidad gramatical, ortográfica y sintáctica del texto generado. Se espera un uso adecuado del lenguaje, sin errores formales ni estructuras confusas.
- Claridad: Mide qué tan comprensible es el mensaje. Una respuesta clara transmite la idea principal sin ambigüedades ni exceso de complejidad.
- Completitud: Considera si la respuesta aborda adecuadamente el contenido requerido en la

sección correspondiente, incluyendo todos los elementos relevantes esperados. La escala de evaluación va del 1 al 5, donde cada nivel representa un incremento del 20% en la completitud del contenido. Es decir, una evaluación de 1 corresponde a un 20% de cumplimiento y una puntuación de 5 a una sección 100% completa.

- Tono: Evalúa si el estilo de redacción es apropiado para el contexto profesional. Debe ser coherente, respetuoso y alineado con el uso empresarial o técnico según corresponda.

La escala estandarizada que se utilizará para la evaluación es una escala de los rangos del 1 al 5, donde cada una se define de la siguiente forma:

1. Inaceptable**:** El texto presenta errores graves, incoherencia o no cumple con el criterio.
2. Deficiente: Cumple mínimamente con el criterio, pero con múltiples deficiencias.
3. Aceptable**:** Cumple con lo básico del criterio, sin destacar. Puede mejorar.
4. Bueno**:** Buena ejecución del criterio con leves oportunidades de mejora.
5. Excelente: Cumplimiento destacado y sin observaciones importantes.

Los cuatro criterios escogidos para evaluar son de relevancia en el estudio por diversas razones, las principales son criticar los resultados de una manera profesional y lo más objetivo posible, ya que se mide la generación automatizada con generación histórica de documentos que han llevado iteraciones para lograr tener el documento final. La redacción es relevante porque se necesita un estándar profesional en la entrega de documentos, la claridad sirve para que el mensaje sea sencillo de entender por parte de toda la organización que requiera esta información, el tono debe adecuarse de la misma forma al lenguaje que se utiliza en la empresa o contar con un tono lo suficientemente neutro para ser aceptado en la revisión del texto. Finalmente, el ítem de completitud es una parte crucial para el análisis de resultados y evaluación de estos, ya que al tener una cuantificación del avance que se genera con los textos se puede simular cuanto tiempo y costos se estarían ahorrando al utilizar esta propuesta de mejora con el script de Python.

Para estimar la completitud global de tarea “C” se utilizarán los datos que se recolecten por los expertos en el área de productos, los product managers. El procedimiento para calcular “C” de cada uno de los modelos será obtener un promedio de la encuesta hacia los product managers, se calculará de la siguiente forma:

$$C = \frac{0{,}2 \cdot \sum_{i=1}^{j} N_{ni}}{k} \ \forall n \ \in k$$

Donde:

$k =$ Número total de modelos evaluados

$j =$ Número total de ítems evaluados

$N_{ni} =$ Puntaje otorgado por el product manager (PM) al modelo n *e ítem* i

$C =$ Completitud global de tarea

Evaluación cuantitativa: se utilizará lenguaje BETO para realizar la verificación utilizando Masked Language Model, de esta forma se puede entender el contexto de un documento siguiendo las inferencias que pueden producirse con las palabras aledañas a esa misma, de esta manera se puede analizar la coherencia de un texto siguiendo esta metodología.

BETO es conocido por su capacidad de captar relaciones semánticas y sintácticas para un texto en español, siendo una gran herramienta para evaluaciones textuales. La metodología se basa en la evaluación de predicción al enmascarar tokens generados por los modelos, la puntuación se asigna según la predicción que pueda hacer BETO con el contexto otorgado con las palabras que no han sido enmascaradas (ocultas).

Este procedimiento se vuelve atractivo para aplicarlo a la generación de los documentos con los diferentes lenguajes, así volviéndose una evaluación comparativa a través de la cuantificación que entrega BETO. Los criterios de aceptación en este caso se presentan en la **Tabla 4.2** adjunta:

Tabla 4.2: Criterio de evaluación con BETO.

Criterio	Rango
Alta relevancia	$0{,}90 \ \leq \bar{x}$
Relevancia media	$0{,}70 \ \leq \bar{x} < \ 0{,}90$
Poca relevancia	$\bar{x} < 0{,}70$
Posibles valores de “x”	$x \in [0{,}1]$

Fuente: Elaboración propia

En la **Tabla 4.2** se presenta el criterio de evaluación que se utilizará dado el puntaje que otorga de relevancia el modelo BETO, x será el valor que otorgue por sección BETO y el promedio de *x* es el que se evaluará, mínimamente se aceptaran valores con alta relevancia, siendo la relevancia media

una alerta para la necesidad de una revisión necesaria por parte de los usuarios finales.

Para calcular la mejora del proceso se empleará un cálculo con la completitud global de la tarea estimada con las respuestas de los expertos en documentación y encargados de evaluar los documentos en el flujo día a día, los product managers evaluarán la completitud de los documentos y se utilizará el siguiente procedimiento para calcular el ahorro empleado en el flujo

Por lo tanto, con la completitud del documento sumado al tiempo empleado en desarrollar el documento se puede estimar el ahorro en términos económicos del desarrollo del documento con la generación automatizada de documentos propuesta de la siguiente forma:

$$A = D_a - D_p$$

Donde:

$D_a =$ Costo actual desarrollo de documento

$D_p =$ Costo ahorrado estimado con uso de proyecto.

$A =$ Nuevo costo para completar el documento.

$$D_a = (HH_d \cdot DH) \cdot S \cdot V$$

Donde:

$D_a =$ Costo actual desarrollo de documento

$HH_d =$ Horas hombre disponibles al día.

$DH \quad =$ Días habiles empleados para el desarrollo.

$V =$ Costo estimado por el área para HH

$S =$ Porcentaje de horas efectivas dedicadas a la generación estimada por el área.

$$D_p = D_a \cdot C$$

Donde:

$D_p =$ Costo ahorrado estimado con uso de proyecto.

$D_a =$ Costo actual desarrollo de documento

$C =$ Completitud global de la tarea.

Por otra parte, para el tiempo que el proyecto estará ahorrando se utilizará la completitud global de la tarea estimada a través de la cuantificación otorgada por los expertos en la documentación multiplicada por la cantidad actual de tiempo empleado para crear el documento en estudio, en otras palabras el tiempo total de desarrollo del documento de interés por el porcentaje de completitud que se tiene con

el proyecto, con la suposición que el tiempo de ejecución del proyecto es lo suficientemente bajo respecto al tiempo empleado actual para desarrollar el documento. Esto se presenta de la siguiente forma:

$$T_d = C \cdot G$$

Donde:

$T_d =$ Ahorro de tiempo en días para crear producto usando proyecto.

$C =$ Completitud global de la tarea.

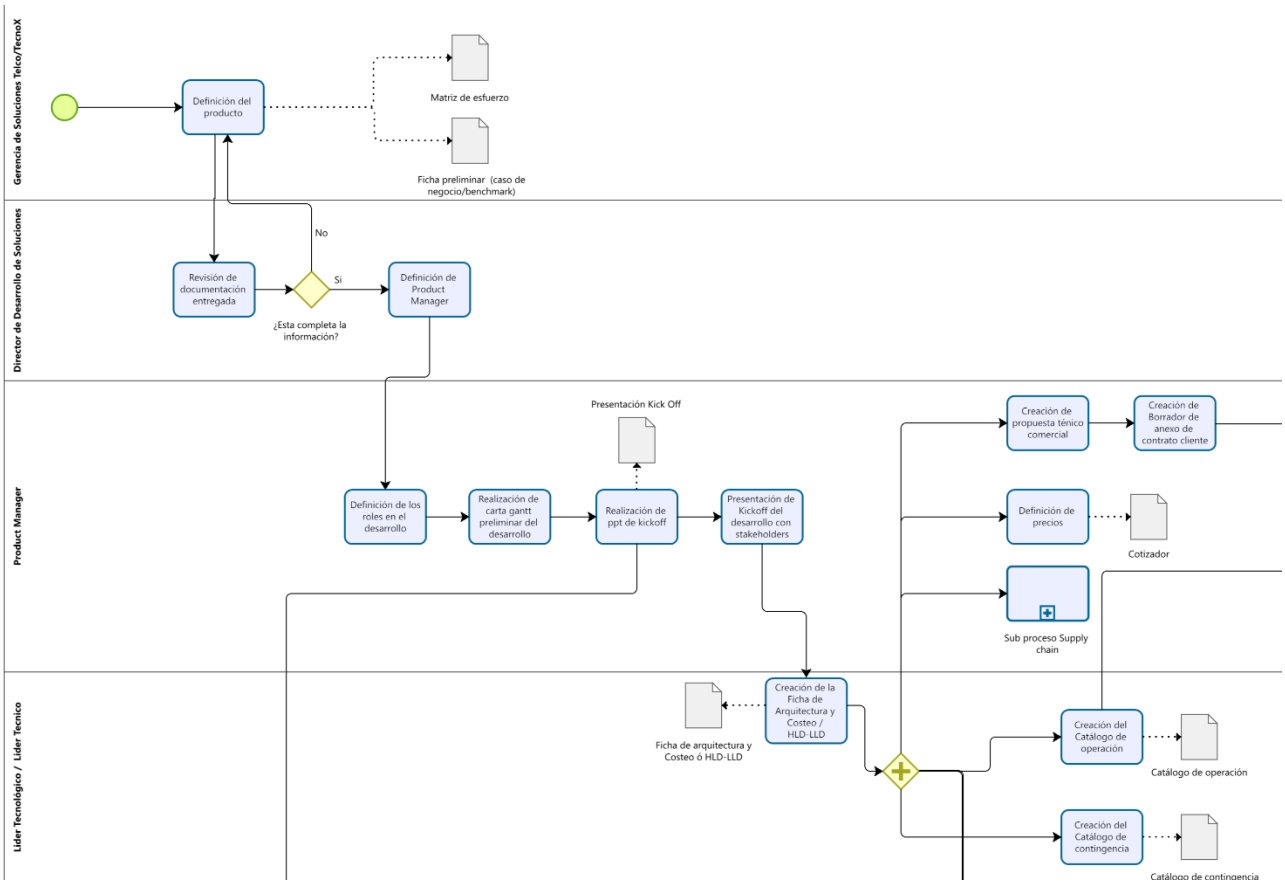
$G =$ Tiempo que tarda el documento en completarse actualmente.

Datos

En este capítulo se presenta información relacionada al área de enfoque del proyecto, desde flujos actuales de desarrollo de los productos hasta indicadores claves de rendimiento (KPI).

5.1 Proceso del área

Figura 5.1: Extracto de flujo "Desarrollo de soluciones".



Fuente: Área desarrollo de soluciones

En la **Figura 5.1** se puede apreciar un extracto del proceso de desarrollo de soluciones, se observa el inicio del proceso y las primeras actividades que se van coordinando para lograr la liberación final del producto. La actividad de “Creación de la Ficha arquitectura y Costeo / HLD-LLD” se observa que es partícipe y desencadenante de una gran cantidad de actividades para el desarrollo del producto. El documento de “Ficha de arquitectura y Costeo o HLD-LLD” es parte importante al igual que diversos otros involucrados en el proceso, sin embargo, al inicio del proceso es primordial conocer la arquitectura y los costos que generará la nueva solución a implementar. Por esta razón, existe la necesidad de obtener este documento lo antes posible para no retrasar el tiempo total de desarrollo (KPI que actualmente mide al área de desarrollo de soluciones), es una gran oportunidad de mejora

obtener el documento en el menor tiempo posible para que implique en una disminución en los tiempos del desarrollo y aportar en el indicador del área.

5.2 **Diagnóstico del área**

El estado actual de las soluciones y los documentos que se encuentran creados se investigó por segmento y la cantidad de soluciones activas. Los segmentos hacen referencia a como la empresa divide los clientes según el tamaño de la organización que representa. Entre ellas se destacan el segmento residencial, corporaciones y empresas:

- Residencial: Corresponde al mundo de telecomunicaciones de los clientes individuales, es decir, clientes de hogar que necesitan conectividad o soluciones ofertadas. Residencial comúnmente es el segmento que maneja la mayor cantidad de clientes, esto es explicado porque es una venta casi individual y muy masiva.
- Corporaciones: Corresponde según tamaño a la definición de grandes empresas, detallando que cuentan con capitales propios y grandes financiamientos. Generalmente cuentan con instalaciones propias y sus ventas son de varios millones de dólares. Es común que el mayor ingreso de la organización venga referenciado a este segmento.
- PYME y Empresas: El otro segmento importante es el de Empresas, se refiere a cualquier tipo de empresa que no cumpla para ser incluido en la definición del segmento corporaciones. Para este segmento es relevante realizar contratos específicos por posibles inestabilidades de estas empresas y por esta razón se tiene un segmento aparte.

A continuación, se presenta un resumen de las soluciones encontradas en la organización con el detalle de cuál es el porcentaje de cumplimiento de los documentos establecidos a incluir en cada solución.

Tabla 5.1: Resumen soluciones documentadas.

Documentación por solución		
Segmento	**Soluciones activas**	**Cumplimiento en %**
Residencial	9	22%
Corporaciones	21	67%
Empresas	10	60%

Fuente: Elaboración propia con datos del área.

Como se puede observar en la **Tabla 5.1**, existe una oportunidad de mejora en el porcentaje de cumplimiento de documentación. Se puede apreciar que el segmento residencial es el que menos productos o servicios (soluciones) tiene documentadas hoy en día. Se observa que los otros segmentos tienen un porcentaje de cumplimiento un poco mayor pero completamente mejorable en términos de aumentar el conocimiento dentro de la organización.

5.3 Documentos requeridos

Un hito importante a la hora del desarrollo de productos es la creación del primer documento. Ya que como se ve en el diagrama de flujo presentado en la **Figura 5.1** un desencadenante o miembro de la ruta crítica es la creación del documento titulado "Ficha de Arquitectura y costeo". Este documento debe detallar información importante para gatillar diferentes áreas de la organización al inicio del desarrollo de cada solución. Los ítems que se abordan en este documento es la información técnica que tiene el producto o servicio entregado por la empresa, además se debe detallar la inversión necesaria (CAPEX) o el costo operacional (OPEX) que involucrará darle marcha a el producto. A continuación, se presenta un breve resumen del contenido de los documentos requeridos por el área:

- "Ficha arquitectura y costeo": Este documento contiene información crucial para la operación del producto o servicio al detallar sus componentes que hacen funcional dicho producto y los costos asociados de los equipos e inversiones requeridas para la organización.
- "Gestión de servicio": En este documento se detallan los flujos que se emplean a la hora de realizar habilitaciones, bajas y modificaciones de los servicios, se detalla el paso a paso de cómo se gestiona el servicio.

- “Gestión de contingencia”: El documento de contingencia cubre el procedimiento que se debe realizar ante diferentes tipos de caídas del servicio respecto a contingencias, es información crucial para la operación en caso de algún desperfecto técnico o alguna situación fuera del alcance de la organización.

Estos son algunos de los documentos que se requieren generar durante el proceso de creación de soluciones, con estos archivos se trabajará en la generación de documentos para el proyecto de automatizar la producción con técnicas de inteligencia artificial.

5.4 **KPI del área**

Los indicadores claves de rendimiento son muy importantes a la hora de evaluar el estado de un área en algún periodo de tiempo por la gerencia correspondiente. En el área de desarrollo de productos históricamente existía un indicador clave que medía la cantidad de soluciones que se estaban liberando en un año.

El KPI que se implementó durante el último tiempo es la “ahorro de tiempo de desarrollo de soluciones en un 15% respecto al tiempo que se obtuvo durante el 2024”. Lo que quiere decir este indicador es una mejora de eficiencia respecto al trabajo que se realizó durante el 2024.

Para lograr con este indicador se deben ajustar varias tareas que se llevan a desarrollo durante la creación de nuevos productos, el proyecto que se integrará al desarrollo de soluciones busca apoyar al área en la optimización del proceso de generación de documentos, para que de esta forma se pueda reducir y cumplir con el KPI establecido para el área.

Los tiempos que se obtuvieron durante el 2024 fueron variables según la dificultad de la solución, a continuación, se muestra un ejemplo para un producto desarrollado durante el 2024 con la planificación de cronograma estimado y el real:

Tabla 5.2: Ejemplo de cronograma de producto.

Ítem	Cronograma Planificado	Cronograma Real
Fecha de inicio	30 de abril de 2024	30 de abril de 2024
Fecha de término	26 de julio de 2024	30 de agosto de 2024
Duración total	87 días	122 días
Diferencia (retraso)	—	+35 días

Fuente: Elaboración propia con datos del área.

Los tiempos de cada actividad son variables según los suministros previos que necesita cada una de ellas, en ciertos casos son documentos como lo es "Ficha de arquitectura y costeo o HLD-LLD" demoran más del tiempo previsto por razones fuera del alcance del área implicando un atraso directo en la liberación del producto.

Sin embargo, al implementar un desarrollo que pueda ayudar a las áreas técnicas encargadas del desarrollo de los documentos a tener un borrador avanzado del documento a completar puede disminuir los tiempos empleados afectando directamente a un menor tiempo en el cronograma real. En la **Tabla 5.3** se muestra el tiempo esperado de generación del documento de ejemplo "Ficha de arquitectura y costeo"

Tabla 5.3: Ejemplo plazo estimado de documento.

Documento	Plazo estimado
Ficha arquitectura y costeo	20 días hábiles

Fuente: Elaboración propia

5.5 **Tiempos y costos del área**

Para los últimos desarrollos se han obtenido tiempos que son variables según la dificultad de la solución que se está entregando. Existe una división de tres tipos de soluciones que se desarrollan, se agrupan en baja, mediana y alta complejidad, cada una de ellas con un tiempo promedio estimado para el desarrollo:

Tabla 5.4: Tiempo estimado por complejidad de solución en 2024.

Complejidad	Detalle de desarrollo	Tiempo estimado
Baja	Actualización de flujo o similar.	1 mes
Media	Nuevo producto de proveedor existente.	3 meses
Alta	Nuevo producto sin antecedentes en sistemas.	6 meses

Fuente: Elaboración propia.

La complejidad se determina por los directores de cada línea de negocio tal como se muestra en el diagrama de flujo en **Figura 5.1**, este es un requisito previo antes de comenzar con el desarrollo de un producto o servicio nuevo a entregar.

El costo es implícito, ya que no corresponde a una cantidad que se invierta para iniciar el desarrollo, aun así, existen valores estimados según el documento "Matriz de esfuerzo" donde se determina la complejidad del desarrollo medido en los casos presentados en la **Tabla 5.5**, se resume el costo según HH invertidas en el desarrollo del producto para esos casos.

Tabla 5.5: Costo estimado por complejidad de solución en 2024.

Complejidad	Detalle de desarrollo	Costo estimado
Baja	Actualización de flujo o similar.	$1.000.000 CLP
Media	Nuevo producto de proveedor existente.	$4.000.000 CLP
Alta	Nuevo producto sin antecedentes en sistemas.	$12.000.000 CLP

Fuente: Elaboración propia.

Por lo tanto, con la completitud global de la tarea sumado al tiempo empleado en desarrollar el documento se puede estimar el ahorro en términos económicos y temporales del desarrollo del documento con la generación automatizada de documentos propuesta.

5.6 Flujo de trabajo

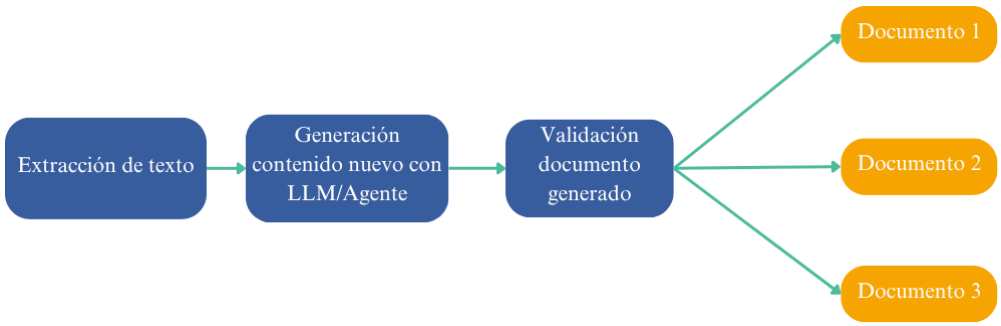
Para la generación de documentos se necesita definir un flujo de trabajo que sea sencillo para un posterior mantenimiento y un fácil entendimiento para el momento que se decida implementar este proyecto dentro del área de desarrollo de soluciones.

El flujo inicia tomando una base de datos que servirá como contexto para entregar ya sea a la API de inteligencia artificial generativa, a un modelo grande de lenguaje de manera local o a un agente alimentado por un modelo grande de lenguaje. Este primer paso es muy relevante, dado que según la cantidad de información que se entregue en la entrada del proceso generará un impacto directo a los resultados que se obtendrán como salida. El proceso actualmente cuenta con dos documentos iniciales que forman parte del comienzo del proceso de soluciones nuevas, estos documentos se titulan "Ficha preliminar desarrollo de soluciones" y "Matriz de esfuerzo", en este caso la ficha preliminar es un

archivo de formato word que contiene información del alcance y detalles de la solución a desarrollar, mientras que la matriz de esfuerzo corresponde a un documento en formato Excel que contiene una estimación del esfuerzo a emplear en el desarrollo de la solución, esto otorga información necesaria de la duración estimada para liberar el producto. Por esta razón, son dos archivos claves para identificar que solución se va a desarrollar y el tiempo que tendrá el product manager para liberar el producto o servicio.

El flujo del proceso de creación de documentos con grandes modelos de lenguaje inicia obteniendo la información de la "Ficha preliminar desarrollo de soluciones", posterior a esto se genera un cuestionario entregando el contexto que se obtiene del documento y almacenando las respuestas en Python para posteriormente enviar estas respuestas a un documento que contendrá todos los ítems necesarios de los demás documentos. En esta parte del proceso ya se tiene un documento generado y se procede a la validación con el lenguaje BETO para determinar si el texto generado es relevante con el contexto que se está generando por parte del modelo de lenguaje grande o el agente.
Todo el proceso se resume a este esquema:

Figura 5.2: Flujo de trabajo script de Python.



Fuente: Elaboración propia.

Resultados

En este capítulo se presentan los resultados obtenidos tras implementar el desarrollo del programa de la solución propuesta para la generación de documentación técnica mediante el uso de grandes modelos de lenguaje (LLM). El procedimiento se realiza en cuatro iteraciones para realizar una comparación posterior y poder obtener las mejores prácticas implementadas.

6.1 Estimación de costos para documento.

La **Tabla 5.2** es un ejemplo real de un cronograma para la generación de un producto completo, desde la planificación hasta la liberación misma del producto, ahora bien el proyecto busca disminuir los tiempos que tardan en obtener los documentos presentes en dicho proceso, por lo tanto, se obtuvo en la **Tabla 5.3** la proyección que tiene el área de desarrollo para generar un documento de arquitectura y costeo, siendo este un factor clave en la ruta crítica del proceso de desarrollo de productos. La duración es de 4 semanas laborales para crear este documento, por lo tanto, se puede obtener el tiempo que disminuye el proceso conociendo que con el proyecto este documento puede iniciar con una cierta "completitud" que será estimada según la opinión y cuantificación de los expertos del proceso. Por esta razón, se tiene que el desarrollo de este documento sería aproximadamente 4 semanas o 20 días laborales.

$$T_d = C \cdot 20$$

Por otra parte, el costo para generar el documento que se estará estudiando en este caso, es el documento arquitectura y costeo, todo esto es para simplificar cálculos respecto a la cantidad de documentos que pueden ser generados paralelamente, aun así, el documento de "ficha de arquitectura y costeo" forma parte del inicio del desarrollo, siendo parte de la ruta crítica del proyecto afectando directamente al tiempo de desarrollo. El costo tomando en cuenta 20 días hábiles de desarrollo se estima:

$$D_a = (HH_d \cdot DH) \cdot \mathrm{S} \cdot D = (8 \cdot 20) \cdot 10\% \cdot 9318{,}4 = 149094{,}4 \approx 149.094 CLP$$

$$D_a \approx \$149.094\ CLP$$

Donde el costo de la mano de obra estimado por el área es $9318,4 CLP, considerando un sueldo aproximadamente de $1.500.000 CLP. Las horas laborales diarias disponibles son 8 y el documento tarda 20 días en completarse en el caso estimado para un desarrollo de alta complejidad. Estos 20 días dan como resultado 160 horas continuas, sin embargo, se estima por el área de desarrollo de soluciones

que solo se utiliza el 10% del total de horas como dedicación completa para generar el documento por parte de las áreas involucradas.

6.2 **Generación con Llama3-70b.**

El primer documento que se generó con el script fue utilizando el servicio disponible de Groq con el lenguaje Llama3-70b, los parámetros obtenidos al correr este script se presentan a continuación:

- Tiempo de ejecución: 622,11 segundos
- Cantidad de ítems generados: 20 de 20 ítems y 4 documentos generados
- Tokens ingresados: 4511
- Tokens generados: 3956

El costo de generación de este LLM se calcula con los datos obtenidos en la **Tabla *4.1***.

$$G_2 = 4511 \cdot \frac{0{,}59}{1M} + 3956 \cdot \frac{0{,}79}{1M} \approx 0{,}00579\ USD \approx 5{,}65\ CLP$$

El validador del documento generado de BETO es introducido luego de que el documento inicial con todos los ítems es generado, se obtiene el puntaje para cada uno de los ítems:

Tabla 6.1: Resumen de puntaje obtenido con BETO Llama3-70b.

Nº	Sección	Puntaje BETO
1	Nombre Producto	0,9385
2	Alcance	0,9411
3	Definición Servicio	0,9259
4	Casos Uso Pertinentes	0,9281
5	Consideraciones Generales	0,9150
6	Diagrama Arquitectura	0,9174
7	Responsabilidad Administración	0,9340
8	Capacity Planning	0,9367
9	Costo Servicio	0,9314
10	Personal Requerido	0,9353
11	Procedimiento Contingencia	0,9357
12	Flujo Contingencia	0,9206
13	Escalamientos	0,9432
14	Características Producto	0,9378
15	Exclusiones Comerciales	0,9370
16	Método Aprovisionamiento	0,9329
17	Procedimiento Postventa	0,9282
18	Procedimientos Habilitación	0,9421
19	Flujo Gestión Incidentes	0,9246
20	Nivel Servicio SLA	0,9210
—	**Promedio BETO**	**0,9313**

Fuente: Elaboración propia.

La evaluación por parte de los usuarios finales para este caso se presenta a continuación

Tabla 6.2: Promedio feedback product managers para Llama3-70b.

Sección	Alcance del producto	Diagrama de arquitectura	Costo servicio	Consideraciones generales	Promedio
Redacción	2,7	3,3	3,3	3,3	3,2
Claridad	2,3	2,7	2,7	2,7	2,6
Completitud	1,7	3,0	2,0	2,3	2,3
Tono	3,0	3,3	3,0	3,3	3,2

Nota: El universo de respuestas consideradas es n = 192 y para cada modelo $n_i = 48$
Fuente: Elaboración propia.

La completitud global de la tarea estimada para el modelo de generación se calcula de la siguiente forma:

$$C = \frac{0{,}2 \cdot (1{,}7 + 3 + 2 + 2{,}3)}{4} \cdot 100 = 45\%$$

Con la estimación del porcentaje de documento que estaría completo podemos calcular el ahorro en tiempo y en costo de la siguiente forma:

$$T_d = C \cdot 20 = 0{,}45 \cdot 20 = 9$$

Por lo tanto, el tiempo nuevo para realizar el documento por completo sería de 11 días aproximadamente. Mientras que para el costo ahorrado se obtiene:

$$D_p = D_a \cdot C = \ 149.094 \cdot 0{,}45 = \$67.092{,}3 CLP$$

Finalmente, se tiene que el nuevo costo para generar el documento es:

$$A = D_a - D_p \ = \ 149.094 \ - \ 67.092{,}3 \approx \$82.000 \ CLP$$

6.3 **Generación con Llama3-8b.**

El segundo documento que se generó con el script fue utilizando el servicio disponible de Groq con el lenguaje Llama3-8b, los parámetros obtenidos al correr este script se presentan a continuación:

- Tiempo de ejecución: 548,34 segundos.
- Cantidad de ítems generados: 20 de 20 ítems y 4 documentos generados
- Tokens ingresados: 4511
- Tokens generados: 4232

El costo de generación de este LLM se calcula con los datos obtenidos en la **Tabla *4.1***.

$$G_3 = 4511 \cdot \frac{0{,}05}{1M} + 4232 \cdot \frac{0{,}08}{1M} \ \approx 0{,}000564 \ USD \ \approx 0{,}54 \ CLP$$

El validador del documento generado de BETO obtiene el puntaje para cada uno de los ítems:

Tabla 6.3: Resumen de puntaje obtenido con BETO para Llama3-8b.

Nº	Sección	Puntaje BETO
1	Nombre Producto	0,9327
2	Alcance	0,9377
3	Definición Servicio	0,9245
4	Casos Uso Pertinentes	0,9212
5	Consideraciones Generales	0,9033
6	Diagrama Arquitectura	0,9088
7	Responsabilidad Administración	0,9289
8	Capacity Planning	0,9343
9	Costo Servicio	0,9252
10	Personal Requerido	0,9337
11	Procedimiento Contingencia	0,9304
12	Flujo Contingencia	0,9165
13	Escalamientos	0,9407
14	Características Producto	0,9323
15	Exclusiones Comerciales	0,9305
16	Método Aprovisionamiento	0,9270
17	Procedimiento Postventa	0,9272
18	Procedimientos Habilitación	0,9369
19	Flujo Gestión Incidentes	0,9187
20	Nivel Servicio SLA	0,9149
—	**Promedio BETO**	**0,9263**

Fuente: Elaboración propia.

La evaluación por parte de los usuarios finales para este caso se presenta a continuación

Tabla 6.4: Promedio de feedback product managers para Llama3-8b.

Sección	Alcance del producto	Diagrama de arquitectura	Costo servicio	Consideraciones generales	Promedio
Redacción	2,7	4,0	3,3	2,7	3,2
Claridad	3,0	3,7	3,0	2,3	3,0
Completitud	2,3	3,7	2,7	2,3	2,8
Tono	3,7	4,0	3,3	3,3	3,6

Nota: El universo de respuestas consideradas es n = 192 y para cada modelo $n_i = 48$
Fuente: Elaboración propia.

La completitud global de la tarea estimada para el modelo de generación utilizando la conexión con Google Gemini se calcula de la siguiente forma:

$$C = \frac{0{,}2 \cdot (2{,}3 + 3{,}7 + 2{,}7 + 2{,}3)}{4} \cdot 100 = 55\%$$

Con la estimación del porcentaje de documento que estaría completo podemos calcular el ahorro en

tiempo y en costo de la siguiente forma:

$$T_d = C \cdot 20 = 0{,}55 \cdot 20 = 11$$

Por lo tanto, el tiempo nuevo para realizar el documento por completo sería de 9 días aproximadamente. Mientras que para el costo ahorrado tenemos:

$$D_p = D_a \cdot C = \ 149.094 \cdot 0{,}55 = \$82.001{,}7 CLP$$

Finalmente, se tiene que el nuevo costo para completar el documento es:

$$A = D_a - D_p \ = \ 149.094 \ - 82.001.7 \ \approx \$67.090 \ CLP$$

6.4 **Generación de documentos con API de Gemini.**

Para generar el tercer documento se estará utilizando la metodología de realizar una llamada a la API de Google Gemini para obtener respuestas directas del lenguaje de inteligencia artificial que proporciona el servicio, para ello se esquematiza el proceso que sigue el programa en la **Figura 5.2**. Los resultados obtenidos al correr este script se presentan a continuación:

- Tiempo de ejecución: 391,8 segundos, 6 minutos aproximadamente.
- Cantidad de ítems generados: 20 de 20 ítems y 4 documentos generados.
- Tokens ingresados: 3778
- Tokens generados: 3032

El costo de generación de este LLM se calcula con los datos obtenidos en la **Tabla *4.1***.

$$G_1 = 3778 \cdot \frac{0{,}1}{1M} + 3032 \cdot \frac{0{,}4}{1M} \ \approx 0{,}00159 \ USD \ \approx 1{,}53 \ CLP$$

El validador del documento generado de BETO es introducido luego de que el documento inicial con todos los ítems es generado con Gemini, se obtiene el puntaje para cada uno de los ítems:

Tabla 6.5: Resumen de puntaje obtenido con BETO para Gemini.

Nº	Sección	Puntaje BETO
1	Nombre Producto	0,9361
2	Alcance	0,9352
3	Definición Servicio	0,9247
4	Casos Uso Pertinentes	0,9269
5	Consideraciones Generales	0,9141
6	Diagrama Arquitectura	0,9172
7	Responsabilidad Administración	0,9202
8	Capacity Planning	0,9311
9	Costo Servicio	0,9349
10	Personal Requerido	0,9283
11	Procedimiento Contingencia	0,9262
12	Flujo Contingencia	0,9114
13	Escalamientos	0,9390
14	Características Producto	0,9342
15	Exclusiones Comerciales	0,9386
16	Método Aprovisionamiento	0,9396
17	Procedimiento Postventa	0,9326
18	Procedimientos Habilitación	0,9378
19	Flujo Gestión Incidentes	0,9113
20	Nivel Servicio SLA	0,9272
—	**Promedio BETO**	**0,9283**

Fuente: Elaboración propia.

La evaluación por parte de los usuarios finales para este caso se presenta a continuación:

Tabla 6.6: Promedio de feedback product managers para Gemini.

Sección	Alcance del producto	Diagrama de arquitectura	Costo servicio	Consideraciones generales	Promedio
Redacción	2,0	3,0	3,3	2,7	2,8
Claridad	2,0	2,3	2,3	2,7	2,3
Completitud	2,0	2,0	2,3	2,7	2,3
Tono	3,3	3,3	3,3	3,0	3,3

Nota: El universo de respuestas consideradas es n = 192 y para cada modelo $n_i = 48$
Fuente: Elaboración propia.

La completitud global de la tarea estimada para el modelo de generación utilizando la conexión con Google Gemini se calcula de la siguiente forma:

$$C = \frac{0{,}2 \cdot (2{,}0 + 2{,}0 + 2{,}3 + 2{,}7)}{4} \cdot 100 = 45\%$$

Con la estimación del porcentaje de documento que estaría completo podemos calcular el ahorro en

tiempo y en costo de la siguiente forma:

$$T_d = C \cdot 20 = 0{,}45 \cdot 20 = 9$$

Por lo tanto, el tiempo nuevo para realizar el documento por completo sería de 11 días aproximadamente. Mientras que para el costo ahorrado tenemos:

$$D_p = D_a \cdot C = \ 149.094 \cdot 0{,}45 = \$67.092{,}3 CLP$$

Finalmente, se tiene que el nuevo costo para completar el documento es:

$$A = D_a - D_p \ = \ 149.094 \ - 67.092.3 \ \approx \$82.000 CLP$$

6.5 **Generación de documentos con agente.**

Los últimos documentos que se generaron con el script fueron utilizando el servicio disponible de Groq con el lenguaje Llama3-8b sumado al agente que se alimenta de este modelo de lenguaje, los parámetros obtenidos al correr este script se presentan a continuación:

- Tiempo de ejecución: 1833,6 segundos.
- Cantidad de ítems generados: 20 de 20 ítems y 4 documentos generados.
- Tokens ingresados: 4557
- Tokens generados: 8165

El costo de generación de este LLM utilizado por el agente se calcula con los datos obtenidos en la **Tabla *4.1***.

$$G_{3a} = 4557 \cdot \frac{0{,}05}{1M} + 8165 \cdot \frac{0{,}08}{1M} \ \approx 0{,}000881 \ USD \ \approx 0{,}85 \ CLP$$

El validador del documento generado de BETO es introducido luego de que el documento inicial con todos los ítems es generado con el agente, se obtiene el puntaje para cada uno de los ítems:

Tabla 6.7: Resumen de puntaje obtenido con BETO para agente.

Nº	Sección	Puntaje BETO
1	Nombre Producto	0,9326
2	Alcance	0,9400
3	Definición Servicio	0,9240
4	Casos Uso Pertinentes	0,9197
5	Consideraciones Generales	0,9005
6	Diagrama Arquitectura	0,9073
7	Responsabilidad Administración	0,9290
8	Capacity Planning	0,9343
9	Costo Servicio	0,9216
10	Personal Requerido	0,9301
11	Procedimiento Contingencia	0,9303
12	Flujo Contingencia	0,9177
13	Escalamientos	0,9404
14	Características Producto	0,9331
15	Exclusiones Comerciales	0,9298
16	Método Aprovisionamiento	0,9248
17	Procedimiento Postventa	0,9262
18	Procedimientos Habilitación	0,9348
19	Flujo Gestión Incidentes	0,9209
20	Nivel Servicio SLA	0,9122
—	**Promedio BETO**	**0,9255**

Fuente: Elaboración propia.

La evaluación por parte de los usuarios finales para este caso se presenta a continuación:

Tabla 6.8: Promedio de feedback product managers para agente.

Sección	Alcance del producto	Diagrama de arquitectura	Costo servicio	Consideraciones generales	Promedio
Redacción	3,7	4,7	4,3	4,0	4,2
Claridad	3,7	4,3	4,7	4,3	4,3
Completitud	4,0	4,7	4,7	4,3	4,4
Tono	3,7	4,3	4,3	4,0	4,1

Nota: El universo de respuestas es n = 192 y para cada modelo $n_i = 48$
Fuente: Elaboración propia.

La completitud global de la tarea estimada para el modelo de generación utilizando la conexión con Google Gemini se calcula de la siguiente forma:

$$C = \frac{0{,}2 \cdot (4{,}0 + 4{,}7 + 4{,}7 + 4{,}3)}{4} \cdot 100 = 88{,}5\%$$

Con la estimación del porcentaje de documento que estaría completo podemos calcular el ahorro en

tiempo y en costo de la siguiente forma:

$$T_d = C \cdot 20 = 0{,}885 \cdot 20 = 17{,}7 \approx 18\ días$$

Por lo tanto, el tiempo nuevo para realizar el documento por completo sería de 2 días aproximadamente. Mientras que para el costo ahorrado tenemos:

$$D_p = D_a \cdot C = 149.094 \cdot 0{,}885 = \$131.948{,}19\ CLP$$

Finalmente, se tiene que el nuevo costo para completar el documento es:

$$A = D_a - D_p = 149.094 - 131.948.19 \approx \$17.150\ CLP$$

6.6 **Texto generado**

En esta sección se presentan los textos generados para la sección “Diagrama de arquitectura”, el propósito de esta sección es describir los componentes y equipos necesarios para ser utilizados por la organización o el cliente con el fin de entregar un producto o servicio cumpliendo los estándares organizacionales. A continuación, se presenta la generación de cada uno de los modelos para esta sección.

Figura 6.1: Sección “Diagrama de arquitectura” obtenido de modelo Llama3-70b.

"El producto "Plan Internet Grandes Clientes" se compone de varios componentes que funcionan en conjunto para proporcionar acceso a Internet dedicado y confiable a medianas y grandes empresas. Los componentes principales son:

- Lado ACCESO y CLIENTE: Componentes definidos por el área de implementación de servicios IDS.
- Lado Backbone Internet (ISP): Equipamiento L2/L3 Juniper, Switch modelo QFX5100 (Switch Grandes Clientes 10G) y Router MX480 (Router Empresas). Las conexiones directas incluyen el módulo óptico SFP-10G-LR.

Estos componentes funcionan de la siguiente manera: los clientes se conectan o configuran en el equipamiento L2/L3 Juniper, que a su vez se conecta al Backbone Internet de ---. El Backbone Internet utiliza tecnologías y plataformas como Juniper (QFX5100 y MX480) para proporcionar conectividad de alta velocidad y seguridad avanzada.
Además, existe una integración con la plataforma Netcracker para administrar segmentos de IP públicos y configurar recursos de red. Esta integración permite a --- ofrecer servicios de alta calidad y confiabilidad, con un enfoque en la satisfacción y experiencia del cliente."

Fuente: Elaboración propia

Figura 6.2: Sección “Diagrama de arquitectura” obtenido de modelo Llama3-8b.

"El producto Plan Internet Grandes Clientes se compone de varios componentes que se integran entre sí para proporcionar un servicio de Internet dedicado a medianas y grandes empresas. Los componentes clave son:

- Lado ACCESO y CLIENTE: Componentes definidos por el área de implementación de servicios IDS.
- Lado Backbone Internet (ISP): Equipamiento L2/L3 Juniper, Switch modelo QFX5100 (Switch Grandes Clientes 10G) y Router MX480 (Router Empresas). Las conexiones directas incluyen el módulo óptico SFP-10G-LR.

La arquitectura del producto se basa en la tecnología Juniper, con equipamiento L2/L3 Juniper que se utiliza para conectar a los clientes y proporcionar acceso a Internet. El equipo de Backbone Internet es responsable de la configuración y monitoreo del servicio.
Las integraciones con sistemas externos se realizan a través de la plataforma Netcracker, que se utiliza para administrar segmentos de IP públicos y configurar recursos de red. Además, el producto se integra con el sistema de gestión de incidentes JIRA, que se utiliza para gestionar los incidentes y requerimientos del cliente.

En resumen, el producto Plan Internet Grandes Clientes se compone de componentes que se integran entre sí para proporcionar un servicio de Internet dedicado a medianas y grandes empresas, utilizando tecnologías y plataformas de Juniper y Netcracker.
"

Fuente: Elaboración propia

Figura 6.3: Sección “Diagrama de arquitectura” obtenido de modelo Gemini-2.0-flash.

"El producto se compone de:

Lado ACCESO y CLIENTE: Componentes definidos por el área de implementación de servicios IDS.
Lado Backbone Internet (ISP): Equipamiento L2/L3 Juniper, Switch modelo QFX5100 (Switch Grandes Clientes 10G) y Router MX480 (Router Empresas). Las conexiones directas incluyen el módulo óptico SFP-10G-LR.
Funcionamiento: Los clientes se conectan o configuran en el equipamiento L2/L3 Juniper.
Tecnologías/Plataformas: Juniper (QFX5100 y MX480).
Integraciones: Integración con la plataforma Netcracker para administrar segmentos de IP públicos y configurar recursos de red."

Fuente: Elaboración propia

Figura 6.4: Sección “Diagrama de arquitectura” obtenido de modelo Llama3-8b utilizado por agente.

"El producto Plan Internet Grandes Clientes de --- es un servicio de acceso a Internet dedicado que se compone de varios componentes que trabajan juntos para proporcionar una conexión estable y segura a Internet. Los componentes clave del producto son:

- Acceso y cliente: se encarga de conectarse a la red de --- y recibir la conexión a Internet.
- Backbone Internet (ISP): es la infraestructura de red que se encarga de proporcionar la conexión a Internet a los clientes. Utiliza tecnologías de Juniper, como routers y switches, para enrutar el tráfico de Internet.
- Implementación de servicios IDS: se encarga de configurar y monitorear la conexión a Internet del cliente.

Entre estos componentes, el acceso y cliente se conecta a la red de --- a través de un enlace físico, como un cable de fibra óptica. El Backbone Internet (ISP) se encarga de enrutar el tráfico de Internet desde la red de --- hasta el cliente. El equipo de implementación de servicios IDS se encarga de configurar y monitorear la conexión a Internet del cliente.
En cuanto a tecnologías y plataformas, --- utiliza tecnologías de Juniper, como routers y switches, para enrutar el tráfico de Internet. También utiliza la plataforma Netcracker para administrar los segmentos de IP públicos disponibles y configurar los recursos de red necesarios para proveer el servicio de Internet.

En cuanto a integraciones con sistemas externos, el servicio de Internet de --- tiene una integración con la plataforma Netcracker, que se utiliza para administrar los segmentos de IP públicos disponibles y configurar los recursos de red necesarios para proveer el servicio de Internet."

Fuente: Elaboración propia

6.7 **Análisis de resultados**

En esta sección se presentan los resultados obtenidos a partir de la ejecución de las distintas estrategias para resolver la problemática planteada en el proyecto. Las métricas propuestas como metodología permiten evaluar la calidad semántica utilizando BETO y calidad técnica con el feedback obtenido de parte de los expertos en el proceso y documentación, además de la eficiencia operativa comparando en términos de costo y tiempo. La **Tabla 6.9** y **Tabla 6.10** sintetizan los resultados obtenidos.

Tabla 6.9: Resumen métricas para modelos

Iteración	**Método**	**Promedio BETO**	**PM puntuación (1-5)**	Completitud global de la tarea "C"
1	LLM Llama3-70b	**0,9283**	2,8	45%
2	LLM Llama3-8b	**0,9263**	3,1	55%
3	API Gemini	**0,9313**	2,6	45%
4	Agente impulsado con LLM Llama3-8b	**0,9255**	4,2	88,5%

Fuente: Elaboración propia

Los datos presentados en la **Tabla 6.9** permiten identificar datos muy relevantes en cuanto a desempeño de cada una de las estrategias utilizadas. En primer lugar, el "Promedio BETO" que evalúa la similitud semántica entre los textos que han sido generado y el contexto otorgado, en todos los métodos la puntuación promedio muestra valores consistentes de "Promedio BETO>0,92", lo que indica que todas las estrategias que se implementaron lograron mantener un alto grado de coherencia semántica. Si bien la API de Gemini obtuvo el puntaje más alto de esta evaluación la diferencia es marginal y no muestra un cambio significativo respecto a los resultados que se obtuvieron con el resto de las estrategias.

No obstante, al considerar la evaluación que entregan los product managers a través de las encuestas cuantitativas, el agente impulsado por el modelo Llama3-8b destaca por sobre los demás con una puntuación de 4,2 sobre 5, siendo el segundo lugar el mismo modelo sin el uso de un agente con una puntuación promedio de 3,1. Este resultado muestra que el agente genera contenido relevante, además de hacer este contenido lo más alineado a los objetivos y necesidades del usuario, probablemente esta capacidad de generar buen contenido se deba a las propias herramientas y estrategias de validación y refinación de resultados antes de entregarlos al usuario final.

Respecto a la completitud de los documentos, obtenida a partir de las mediciones que se entregaron por parte de los expertos en documentación, se confirma la superioridad del agente respecto a las otras estrategias empleadas, donde el agente alcanza una completitud global de la tarea de un 88,5%, a diferencia de los valores obtenidos de 45% y 55% de las otras estrategias. Este porcentaje nos demuestra que según el usuario final el agente obtiene textos más desarrollados, estructurados y cercanos al producto final esperado, minimizando las posibles intervenciones humanas posteriores.

Tabla 6.10: Resumen métricas.

Iteración	Método	Ahorro	Tiempo de ejecución (s)	Días para completar el documento	Costo del LLM
1	LLM Llama3-70b	$67.092 CLP	622,11	11	$5,65 CLP
2	LLM Llama3-8b	$82.001 CLP	548,34	9	$0,54 CLP
3	API Gemini	$67.092 CLP	391,6	11	$1,53 CLP
4	Agente impulsado con LLM Llama3-8b	$131.948 CLP	1833,6	2	$0,85 CLP

Fuente: Elaboración propia

En la **Tabla 6.10** se presenta un resumen de los datos obtenidos respecto a los tiempos de ejecución, ahorros de costo y tiempo por cada solución. A diferencia de los datos aparentemente superiores del agente respecto a las otras estrategias presentados en la **Tabla 6.9**, en los tiempos de ejecución que se resumen en la **Tabla 6.10** se muestra que el agente presenta el mayor tiempo de ejecución computacional con 1833,6 segundos, esto es debido a que su diseño implica validaciones y análisis intermedios, además de las consultas a las herramientas en caso de ser necesario, de esta forma lo vuelve más intensivo computacionalmente. Esta inversión de tiempo en la generación se ve compensada por los datos que comparamos en la **Tabla 6.9**, al obtener mejores puntuaciones y completitudes respecto a las demás estrategias.

Mientras que los métodos que utilizan directamente un modelo para generar los documentos requieren entre 9 y 11 días para completar un documento, el agente logra disminuir este tiempo de completar el documento a solo 2 días, estos días contienen 16 horas continuas, y considerando el 10% de dedicación total bastaría con una validación y aprobación final agregando o quitando ítems para obtener el documento en un tiempo de 1,6 horas de dedicación total por parte de las áreas involucradas. Esta notable mejoría y disminución impacta directamente con ahorros finales estimados de $131.947 CLP, siendo notablemente mayor que los ahorros que se obtienen con las otras estrategias que se encuentran

entre $67.000 CLP y $82.000 CLP, por otra parte, los costos asociados a la generación de texto varían según cada modelo, siendo el más costoso Llama3-70b con $5,65 CLP, pero el agente que logra una mayor completitud global de la tarea trae costos de generación de $0,85 CLP en estas pruebas, siendo un costo marginal respecto al ahorro que proporciona al utilizar esta estrategia.

Conclusiones

En el presente estudio se aplicaron herramientas de inteligencia artificial a un contexto real sobre generación de documentación con el objetivo de disminuir los tiempos al aprovechar dichas herramientas para generar documentos lo más completos y correctos posible. El desarrollo del proyecto permitió cumplir parcialmente el objetivo general de optimizar el proceso de creación de documentación técnica mediante un script de Python que integra herramientas de inteligencia artificial generativa. Los resultados demuestran que la solución propuesta, especialmente a través del uso de un agente que utiliza un LLM (Llama3-8b), permite generar documentos de una calidad alta al tener un grado de completitud global de la tarea de un 88,5%, lo que disminuye significativamente la carga y la cantidad de HH que se emplean para terminar de desarrollar el documento entregable.
Entre las principales conclusiones se destacan:

La incorporación de un agente que combina generación, validación semántica y las propias estrategias para entregar una respuesta permite obtener documentos más completos (completos al 88,5%) reduciendo los días de trabajo para completar dicho documento (2 días y 1,6 horas de dedicación total), con una reducción de los costos operativos requeridos para la generación de este documento, estimando un ahorro cercano a $131.948 CLP.

Los tiempos de ejecución que se obtuvieron muestran que el agente es el que mayor carga computacional exige respecto a las estrategias que utilizan los modelos de lenguaje directamente, sin embargo, este costo computacional se ve compensado con la disminución de carga de trabajo que se requiere para completar los documentos

Respecto a la evaluación semántica que otorga BETO todos los modelos y estrategias evaluadas presentaron niveles altos de coherencia semántica (Promedio BETO >0,92), pese a eso, la evaluación humana demuestra que el agente es el más completo respecto a la completitud, coherencia y percepción final de utilidad.

La estructura diseñada de Python fue versátil para evaluar con BETO e integrar los diferentes modelos que se utilizaron, además fue un éxito a la hora de integrar los lenguajes con las llamadas al agente con las instrucciones necesarias.

Si bien los costos asociados a la generación de estos textos son lo suficientemente bajos, siendo Llama3-70b el más costoso económicamente con $5,65 CLP, cualquiera de estas opciones estaría ahorrando tiempos y costos que superan con creces el costo de generación de texto.

7.1 Recomendaciones y limitaciones

Concluida la creación y validación del proyecto se recomienda a la organización lo siguiente:

- Implementación del proyecto: Se recomienda a la empresa adoptar este proyecto en el flujo de trabajo, dada su capacidad demostrada para reducir costos y tiempos, comenzar con documentos que necesitan pocos ajustes para estar completos y aportar directamente al KPI con el que miden el área desarrollo de soluciones.
- Monitorear y actualizar el flujo: A la hora de implementar o usar este proyecto es recomendable establecer mecanismos de monitoreo continuo del rendimiento y métricas sobre calidad de documentos generados, con el objetivo de ir ajustando dinámicamente los modelos, flujos y requisitos de los documentos según las necesidades reales de la organización.
- Extender el flujo del proyecto a todos los documentos del negocio: Al ser una opción muy relevante para obtener documentos casi completos una de las recomendaciones hacia la organización es aumentar el alcance de esta solución e integrarlo a todos los documentos que se puedan generar de esta forma, ya sea en el área de desarrollo y gestión de soluciones o en cualquiera otra área de la empresa, siguiendo la misma metodología, pero aplicada a otro contexto.

Las limitantes que se presentaron durante el proyecto se presentan a continuación:

- Utilización de infraestructura como un servicio (IaaS): El proyecto utiliza niveles computacionales altos, por esta razón una posibilidad es utilizar infraestructura con CPU/GPU que soporte los modelos de inteligencia artificial a ejecutar, a pesar de ello, para la validación de este proyecto no se contemplará el uso de IaaS debido a su alto coste para el uso dedicado de GPU.
- Costos de acceso a API: Para la validación del proyecto se utilizarán servicios con acceso 5 gratuito bajo ciertas restricciones de Google y de Groq, de esta forma los costos que se incurren en las API de inteligencia artificial serán una limitante para la presentación del proyecto, entre estas API's encontramos HuggingFace, Groq, Gemini, OpenAI, etc.
- Limitación de recursos disponibles: El trabajo diario de los product managers y los encargados de desarrollar documentación se realiza con un bajo poder de CPU, lo que genera una gran limitante a la hora de correr modelos de forma local, fine tunear, etc. Por este motivo, para

realizar validaciones el proyecto utilizará estrategias sencillas con llamadas a API's que ya contengan estos modelos de forma precargada.

- Bloqueos de ejecutables: Al ser una organización con altos estándares de seguridad y vulnerable a los ataques de programas externos, las limitantes que se generan para estos proyectos son los bloqueos a los ejecutables como el instalador pip, la ejecución de archivos ejecutables de Python, entre otros bloqueos limitantes para el desarrollo del proyecto.

7.2 Trabajos futuros

Trabajar con LLM de manera local

Esta estrategia permitiría una mayor autonomía, asegurando la seguridad de los datos y ahorrando posibles costos al no utilizar API's externas, esto es de suma relevancia en las empresas que tienen restricciones de privacidad.

Aplicar fine-tuning al proyecto

Realizar ajustes conocidos como "fine-tuning" a los modelos bases utilizando datos de documentación histórica creada por la organización. De este modo, se puede adaptar los modelos para obtener texto generado con terminología, estructura y coherencia similar a los textos que ya han sido generados con anterioridad por la empresa.

Otorgar otras herramientas al agente para mejorar resultados

Ampliar la funcionalidad del agente con nuevas herramientas que puedan generar e insertar gráficos a partir de datos, inserción de tablas para mejorar la entrega de información hacia el cliente final, etc.

Desarrollar sistema multi agente

Se pueden implementar mejoras respecto al flujo del agente, elevando el proceso a un sistema multi agente, incorporando agentes especializados para realizar tareas según el tipo de documento, validadores, generadores de texto y gráficos, etc.

Referencias

Abbass, H.A. (2019). Social Integration of Artificial Intelligence: Functions, Automation Allocation Logic and Human-Autonomy Trust. *Cogn Comput* **11**, 159–171 (2019). https://doi.org/10.1007/s12559-018-9619-0.

Ahmadi Achachlouei, M., Patil, O., Joshi, T., & Nair, V. N. (2021). Arquitecturas y tecnologías de automatización de documentos: una encuesta. *arXiv*. https://doi.org/10.48550/arXiv.2109.11603.

Babic, B., Chen, D. L., Evgeniou, T., & Fayard, A. L. (2020). A better way to onboard AI: Understand it as a tool to assist rather than replace people. *Harvard Business Review*, http://hdl.handle.net/10722/334716

Caldeiro, G. (2024). Inteligencia artificial generativa y educación: hacia un nuevo paradigma. *El Faro. Revista Digital De Docencia Universitaria*, *1*(1), 22-43. https://revistaelfaro.uflo.edu.ar/index.php/elfaro/article/view/15

Casar Corredera, J. R. (2023). Inteligencia artificial generativa. *Anales de la Real Academia de Doctores de España*, Vol 8(3), 475–489.

Chango, J. A. Z., & Varela, P. M. R. (2021). Automatización de preselección en el área de talento humano, utilizando tecnologías de NLP y RPA. *Dominio de las Ciencias*, *7*(4), 96.

Chay Brooks, Cristian Gherhes, Tim Vorley. (2020). Inteligencia artificial en el sector legal: presiones y desafíos de la transformación, *Cambridge Journal of Regions, Economy and Society*, Volumen 13, Número 1, marzo de 2020, páginas 135-152, https://doi.org/10.1093/cjres/rsz026.

Cheng, Y., Zhang, C., Zhang, Z., Meng, X., Hong, S., Li, W., Wang, Z., Wang, Z., Yin, F., Zhao, J., & He, X. (2024). Exploring large language model based intelligent agents: Definitions, methods, and prospects. arXiv. https://arxiv.org/abs/2401.03428

Cortez Vásquez, A. V. (2009). Procesamiento de lenguaje natural. *Revista de Ingeniería de Sistemas e Informática*, Vol6(2), 45–54.

da Cunha, I. (2022). Un redactor asistido para adaptar textos administrativos a lenguaje claro. *sepln*, 69, 39–49. https://doi.org/10.26342/2022-69-3.

D. B. Acharya, K. Kuppan and B. Divya. (2025). "Agentic AI: Autonomous Intelligence for Complex Goals—A Comprehensive Survey," in *IEEE Access*, vol. 13, pp. 18912-18936, 2025, doi: 10.1109/ACCESS.2025.3532853.

Getchell, K. M., Carradini, S., Cardon, P. W., Fleischmann, C., Ma, H., Aritz, J., & Stapp, J. (2022). Artificial Intelligence in Business Communication: The Changing Landscape of Research and Teaching. *Business and Professional Communication Quarterly*, *85*(1), 7-33. https://doi.org/10.1177/23294906221074311.

Google. (19 de Marzo de 2025). *GoogleAI*. Obtenido de Google AI for Developers: https://ai.google.dev/gemini-api/docs?hl=es-419

Groq. (2025). Groq Pricing. Obtenido de groq: https://groq.com/pricing.

Huang, Q., Vora, J., Liang, P., & Leskovec, J. (2023). Benchmarking large language models as AI research agents. *Foundation Models for Decision Making Workshop at NeurIPS 2023*.

HuggingFace. (19 de Marzo de 2025). *HuggingFace*. Obtenido de Smolagents: https://huggingface.co/docs/smolagents/index

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., de las Casas, D., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L. R., Lachaux, M.-A., Stock, P., Le Scao, T., Lavril, T., Wang, T., Lacroix, T., & El Sayed, W. (2025). Mistral 7B: Un modelo de lenguaje eficiente y de alto rendimiento. Conferencia de Modelos de Lenguaje.

Junaed Younus Khan y Gias Uddin. (2023). Generación automática de documentación de código mediante GPT-3. En Actas de la 37ª Conferencia Internacional IEEE/ACM sobre Ingeniería de Software Automatizado (ASE '22). Association for Computing Machinery, Nueva York, NY, EE.UU., Artículo 174, 1–6. https://doi.org/10.1145/3551349.3559548.

Lin, C.-H., & Cheng, P.-J. (2024). Redacción de documentos legales con un modelo de lenguaje grande preentrenado y afinado. *arXiv*. https://doi.org/10.48550/arXiv.2406.04202.

Lopezosa, C., & Codina, L. (2023). Probando Bard: así funciona la Inteligencia Artificial Generativa de Google. Anuario ThinkEPI. https://doi.org/10.3145/thinkepi.2023.e17a25.

López Gómez, J. J., García-Peñalvo, F. J., & García-Holgado, A. (2025). Finetune a un LLM para la creación de chats de apoyo al aprendizaje. Caso de estudio para la Ingeniería de Software. En J. A. Cordón Muñoz & J. García Pechero (Eds.), *Avances en Informática y Automática. Decimoséptimo Workshop* (pp. 264-285). Departamento de Informática y Automática. Universidad de Salamanca.

Ramírez, C. A. Y. (2024). Uso de herramientas de procesamiento de lenguaje natural para el análisis y desarrollo de artículos científicos de ingeniería. *Revista de investigación de sistemas e informática*, *17*(2), 5-15.

Rodríguez Fernández, L., Fernández Mena, A. L., Torres Magaña, M. P., Rodríguez Magaña, M. A., & Rodríguez Fernández, M. A. (2024). Inteligencia artificial en la educación: Modelo de lenguaje de gran tamaño (LLM) como recurso educativo. *Revista Ipsumtec*, 7(2), 157-164. https://doi.org/10.61117/ipsumtec.v7i2.321.

Rovane, C. (2004). ¿Qué es un agente? Synthese, 140(1/2), 181-198.

Shen, Z. (2024). *LLM With Tools: A Survey*. arXiv. https://doi.org/10.48550/arXiv.2409.18807

Xu, F. F., Song, Y., Li, B., Tang, Y., Jain, K., Bao, M., Wang, Z. Z., Zhou, X., Guo, Z., Cao, M., Yang, M., Lu, H. Y., Martin, A., Su, Z., Maben, L. M., Mehta, R., Chi, W., Jang, L., Xie, Y., Zhou, S., & Neubig, G. (2025). TheAgentCompany: Benchmarking LLM Agents on Consequential Real World Tasks. Carnegie Mellon University, Independent, Duke University.

Yao, Y., Duan, J., Xu, K., Cai, Y., Sun, Z., & Zhang, Y. (2024). A survey on large language model (LLM) security and privacy: The Good, The Bad, and The Ugly. High-Confidence Computing, 4, 100211.

Anexos

Anexo 1: Código estrategia para iteración con LLM de Groq Llama3 70b y 8b

Figura 9.1: Librerías utilizadas con Groq.

```python
import os
import time
import tkinter as tk
from tkinter import filedialog
from dotenv import load_dotenv
import docx
from docxtpl import DocxTemplate
from openai import OpenAI
```

Fuente: Elaboración propia

Figura 9.2: Inicio cliente Groq.

```python
# Cargar variables de entorno e iniciar cliente OpenAI
load_dotenv("variables.env")
client = OpenAI(
    api_key=os.getenv("GROQ_API_KEY"),
    base_url="https://api.groq.com/openai/v1"
)

modelo_llm = "llama3-70b-8192"  # Cambiar modelo a Groq Llama 3 8b
```

Fuente: Elaboración propia

Figura 9.3: Función de envío prompt.

```python
def enviar_prompt(prompt):
    try:
        response = client.chat.completions.create(
            model=modelo_llm,
            messages=[
                {"role": "system", "content": "Eres un experto product manager especializado en redacción técnica."},
                {"role": "user", "content": prompt}
            ],
            temperature=0.1
        )
        return response.choices[0].message.content.strip()
```

Fuente: Elaboración propia

Figura 9.4: Ejemplo de prompt de sección con preguntas.

```python
prompts = {
    "nombre_producto": "¿Cuál es el nombre oficial del producto o servicio? ¿Existe alguna versión o variación del nombre para diferentes mercados? ¿Cómo se espera que los clientes lo identifiquen?",
    "alcance": "¿A qué segmentos de clientes o industrias está dirigido el producto o servicio? ¿Cuál es su alcance geográfico? ¿Hay planes para expandir a otros mercados en el futuro? ¿Qué necesidades específicas de cada segmento se buscan cubrir?",
    "definicion_servicio": "¿Qué objetivos principales cumple para el cliente? ¿Cómo se define el producto o servicio? ¿Qué problemas específicos resuelve? ¿Cuál es su propuesta de valor frente a la competencia?",
    "casos_uso_pertinentes": "¿Cuáles son los principales casos de uso del producto o servicio? ¿En qué situaciones específicas los clientes lo utilizarán? ¿Hay casos de uso secundarios o menos comunes?",
    "...": "..."
}
```

Fuente: Elaboración propia

Figura 9.5: Ciclo de envío de preguntas por sección.

```python
for i, (campo, pregunta) in enumerate(prompts.items(), start=1):
        prompt_individual = (
            "Eres un experto (product manager) en análisis de información y documentación de productos.\n"
            "Usando el siguiente contexto, responde a la pregunta solo con texto simple (sin asteriscos, listas ni markdown).\n\n"
            f"Contexto:\n{source_text}\n\n"
            f"Pregunta:\n{pregunta}\n\n"
            "Respuesta:"
        )
```

Fuente: Elaboración propia

Figura 9.6: Renderizar y exportar documento.

```python
# Generación del documento
    template_docx_path = "C:/Users/Equipo/Desktop/MT Rodrigo/Templates/Documento madre.docx"
    template_doc = DocxTemplate(template_docx_path)

    try:
        template_doc.render(respuestas)
    except Exception as e:
        print(f"Error al renderizar la plantilla: {e}")
        return

    output_path = "C:/Users/Equipo/Desktop/MT Rodrigo/Documento madre_Groq_llama70.docx"
    template_doc.save(output_path)
```

Fuente: Elaboración propia

Anexo 2: Código estrategia para iteración con API de Google Gemini.

Figura 9.7: Librerías utilizadas con Gemini.

```python
import google.generativeai as genai
from docxtpl import DocxTemplate
import docx
from dotenv import load_dotenv
import os
import tkinter as tk
from tkinter import filedialog
import time
```

Fuente: Elaboración propia

Figura 9.8: Configurar API de Gemini y función de extraer texto.

```python
load_dotenv("variables.env")  # Cargar las variables de entorno desde el archivo .env
genai.configure(api_key=os.getenv("GEMINI_API_KEY"))
tiempo_inicial = time.time()  # Guardar el tiempo inicial para medir la duración del script
def extractor_texto_word(docx_path):
    """
    Extrae y devuelve el texto de todos los párrafos de un archivo DOCX.
    """
    document = docx.Document(docx_path)
    return "\n".join([para.text for para in document.paragraphs if para.text])
```

Fuente: Elaboración propia

Figura 9.9: Función de envío de prompt.

```python
def enviar_prompt(prompt):
    """
    Envía un prompt individual a la API de Gemini y devuelve la respuesta generada.
    """
    try:
        modelo = genai.GenerativeModel("gemini-2.0-flash", generation_config={"temperature": 0.1})
        respuesta = modelo.generate_content(prompt)
        return respuesta.text.strip()
    except Exception as e:
        print(f"Error al generar respuesta: {e}")
        return ""
```

Fuente: Elaboración propia

Figura 9.10: Ciclo para envío de prompts.

```
for i, (campo, pregunta) in enumerate(prompts.items(), start=1):
        # creamos el prompt individual para cada sección
        prompt_individual = (
            "Eres un experto (product manager) en análisis de información y documentación de productos.\n"
            "Utilizando el siguiente contexto, responde a la siguiente pregunta exclusivamente en formato de texto para Word, "
            "IMPORTANTE: no utilizar asteriscos, caracteres especiales ni agregar información adicional a la respuesta. "
            "Proporciona únicamente la respuesta basada en el contexto disponible.\n\n"
            f"Contexto:\n{source_text}\n\n" #enviamos el texto extraído del documento
            f"Pregunta:\n{pregunta}\n\n" #enviamos la pregunta de la sección correspondiente
            "Respuesta:"
        )
```

Fuente: Elaboración propia

Figura 9.11: Renderizar template y exportar documento.

```
template_docx_path = "C:/Users/Equipo/Desktop/MT Rodrigo/Templates/Documento madre.docx"
    template_doc = DocxTemplate(template_docx_path)
    try:
        template_doc.render(respuestas)
    except Exception as e:
        print(f"Error al renderizar la plantilla: {e}")
        return

    # guardar el documento generado
    output_path = "C:/Users/Equipo/Desktop/MT Rodrigo/Documento madre_Gemini.docx"
    template_doc.save(output_path)
```

Fuente: Elaboración propia

Anexo 3: Código estrategia para iteración con agente y LLM de Groq.

Figura 9.12: Librerías utilizadas con Agente.

```
import os
from dotenv import load_dotenv
import tkinter as tk
from tkinter import filedialog
import time
from docxtpl import DocxTemplate
import docx
from langchain.agents import AgentExecutor, create_tool_calling_agent
from langchain_core.tools import tool
from langchain_groq import ChatGroq
from langchain_core.prompts import ChatPromptTemplate
from langchain_core.messages import HumanMessage
```

Fuente: Elaboración propia

Figura 9.13: Función decorada con tool para entregar contexto a agente.

```python
def crear_herramienta_contexto(contexto: str, llm):
    @tool
    def consultar_contexto(pregunta: str) -> str:
        """

        Herramienta para consultar el contexto y responder preguntas específicas.

        """
        prompt = (
            "Eres un experto (product manager) en análisis de información y documentación de productos.\n"
            "Utilizando el siguiente contexto, responde a la siguiente pregunta exclusivamente en formato de texto para Word, \n"
            "Genera un párrafo estructurado de manera profesional que contenga todas las preguntas que contestaste. \n"
            "Proporciona únicamente la respuesta basada en el contexto disponible.\n\n"
            f"Contexto:\n{contexto}\n\n"
            f"Pregunta:\n{pregunta}\n\n"
            "Respuesta:"
        )
        respuesta = llm.invoke([HumanMessage(content=prompt)])
        return respuesta.content.strip()

    return consultar_contexto
```

Fuente: Elaboración propia

Figura 9.14: Definir el agente generador de texto.

```python
class AgenteGenerador:
    def __init__(self, context: str):
        self.context = context
        self.llm = ChatGroq(
            temperature=0.3,
            model_name="llama3-8b-8192",
            api_key=GROQ_API_KEY
        )
        self.tools = [crear_herramienta_contexto(self.context, self.llm)]
        self.agent = self._create_agent()
```

Fuente: Elaboración propia

Figura 9.15: Crear agente.

```python
def _create_agent(self) -> AgentExecutor:
    """
    Crear y configurar el agente reactivo.

    """
    prompt = ChatPromptTemplate.from_messages([
        ("system",
    "Eres un asistente experto en análisis de documentos. "
    "Tu única función es responder preguntas usando herramientas externas. "
    "IMPORTANTE: No respondas directamente a las preguntas. Responde en forma de párrafo estructurado de manera profesional y en español. "
    "Usa SIEMPRE la herramienta `consultar_contexto`."),
        ("placeholder", "{chat_history}"),
        ("human", "{input}"),
        ("placeholder", "{agent_scratchpad}")
    ])

    agent = create_tool_calling_agent(
        llm=self.llm,
        tools=self.tools,
        prompt=prompt
    )
    return AgentExecutor(agent=agent, tools=self.tools, verbose=True)
```

Fuente: Elaboración propia

Figura 9.16: Función de ejecución del agente.

```python
def responder_pregunta(self, pregunta: str) -> str:
        """Ejecutar el agente para responder una pregunta específica."""
        result = self.agent.invoke({"input": pregunta})
        return result["output"]
```

Fuente: Elaboración propia

Figura 9.17: Ciclo para procesar cada pregunta con función de ejecución.

```python
# Procesar cada pregunta
    for i, (campo, pregunta) in enumerate(prompts.items(), start=1):
        print(f"Procesando pregunta: {campo}")
        respuestas[campo] = agent.responder_pregunta(pregunta)
```

Fuente: Elaboración propia

Anexo 4: Encuesta realizada a product managers

Figura 9.18: Instrucciones de evaluación de encuesta.

Descripción de evaluación

En este formulario se presentan respuestas generadas por distintos modelos de lenguaje a partir de una misma instrucción (prompt) y base de datos. Se mostrarán 5 secciones que se incluyen en los documentos que ustedes conocen, cada respuesta debe ser evaluada de forma independiente y objetiva, según cuatro criterios de calidad textual:

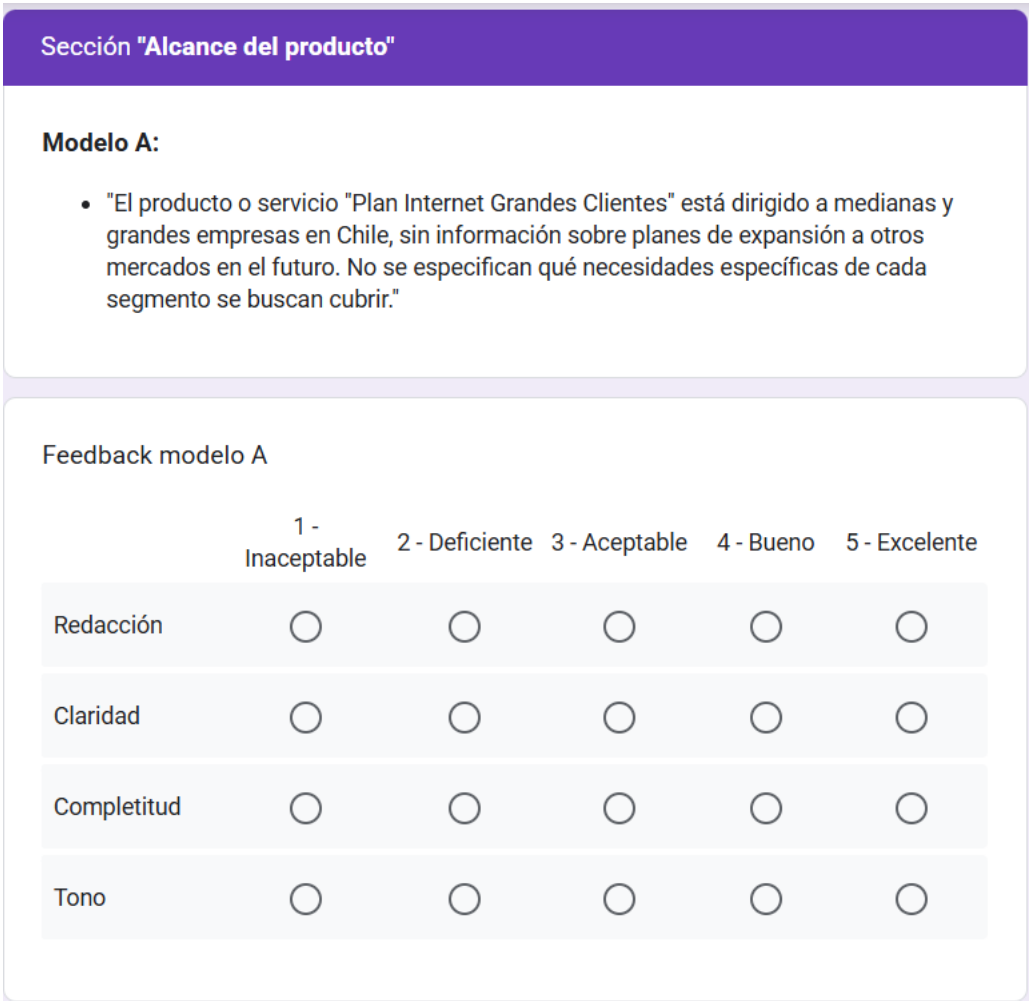
- **Redacción:** Evalúa la calidad gramatical, ortográfica y sintáctica del texto generado. Se espera un uso adecuado del lenguaje, sin errores formales ni estructuras confusas.
- **Claridad:** Mide qué tan comprensible es el mensaje. Una respuesta clara transmite la idea principal sin ambigüedades ni exceso de complejidad.
- **Completitud:** Considera si la respuesta aborda adecuadamente el contenido requerido en la sección correspondiente, incluyendo todos los elementos relevantes esperados. La escala de evaluación va del 1 al 5, donde cada nivel representa un incremento del 20% en la completitud del contenido. Es decir, una evaluación de 1 corresponde a un 20% de cumplimiento y una puntuación de 5 a una sección 100% completa.
- **Tono:** Evalúa si el estilo de redacción es apropiado para el contexto profesional. Debe ser coherente, respetuoso y alineado con el uso empresarial o técnico según corresponda.

Cada criterio debe evaluarse en una escala de 1 a 5 según el siguiente marco estandarizado

1. **Inaceptable:** El texto presenta errores graves, incoherencia o no cumple con el criterio.
2. **Deficiente**: Cumple mínimamente con el criterio, pero con múltiples deficiencias.
3. **Aceptable:** Cumple con lo básico del criterio, sin destacar. Puede mejorar.
4. **Bueno:** Buena ejecución del criterio con leves oportunidades de mejora.
5. **Excelente**: Cumplimiento destacado y sin observaciones importantes.

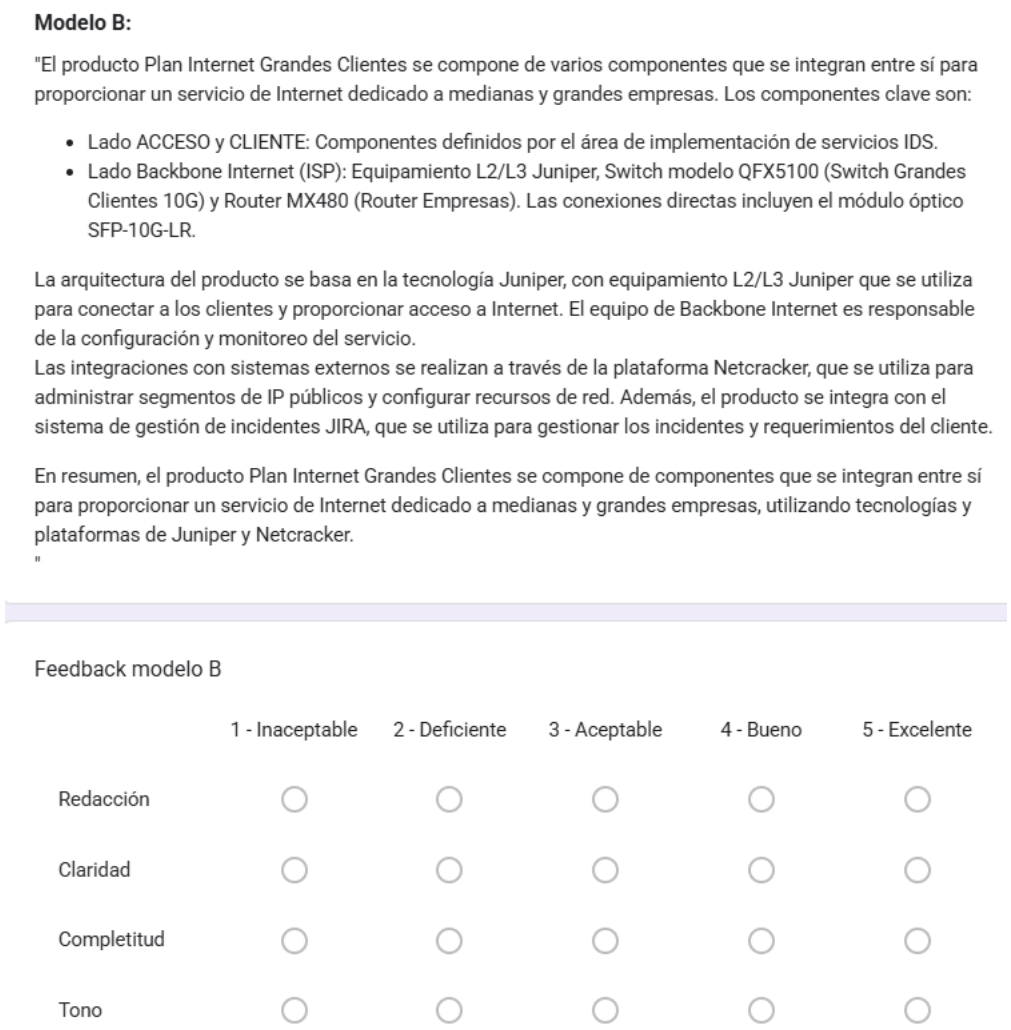
Fuente: Elaboración propia

Figura 9.19: Ítem "Alcance del producto" ejemplo de evaluación con modelo Llama3-70b.



Fuente: Elaboración propia

Figura 9.20: Ítem "Diagrama de arquitectura" ejemplo de evaluación modelo Llama3-8b.



Modelo B:

"El producto Plan Internet Grandes Clientes se compone de varios componentes que se integran entre sí para proporcionar un servicio de Internet dedicado a medianas y grandes empresas. Los componentes clave son:

- Lado ACCESO y CLIENTE: Componentes definidos por el área de implementación de servicios IDS.
- Lado Backbone Internet (ISP): Equipamiento L2/L3 Juniper, Switch modelo QFX5100 (Switch Grandes Clientes 10G) y Router MX480 (Router Empresas). Las conexiones directas incluyen el módulo óptico SFP-10G-LR.

La arquitectura del producto se basa en la tecnología Juniper, con equipamiento L2/L3 Juniper que se utiliza para conectar a los clientes y proporcionar acceso a Internet. El equipo de Backbone Internet es responsable de la configuración y monitoreo del servicio.
Las integraciones con sistemas externos se realizan a través de la plataforma Netcracker, que se utiliza para administrar segmentos de IP públicos y configurar recursos de red. Además, el producto se integra con el sistema de gestión de incidentes JIRA, que se utiliza para gestionar los incidentes y requerimientos del cliente.

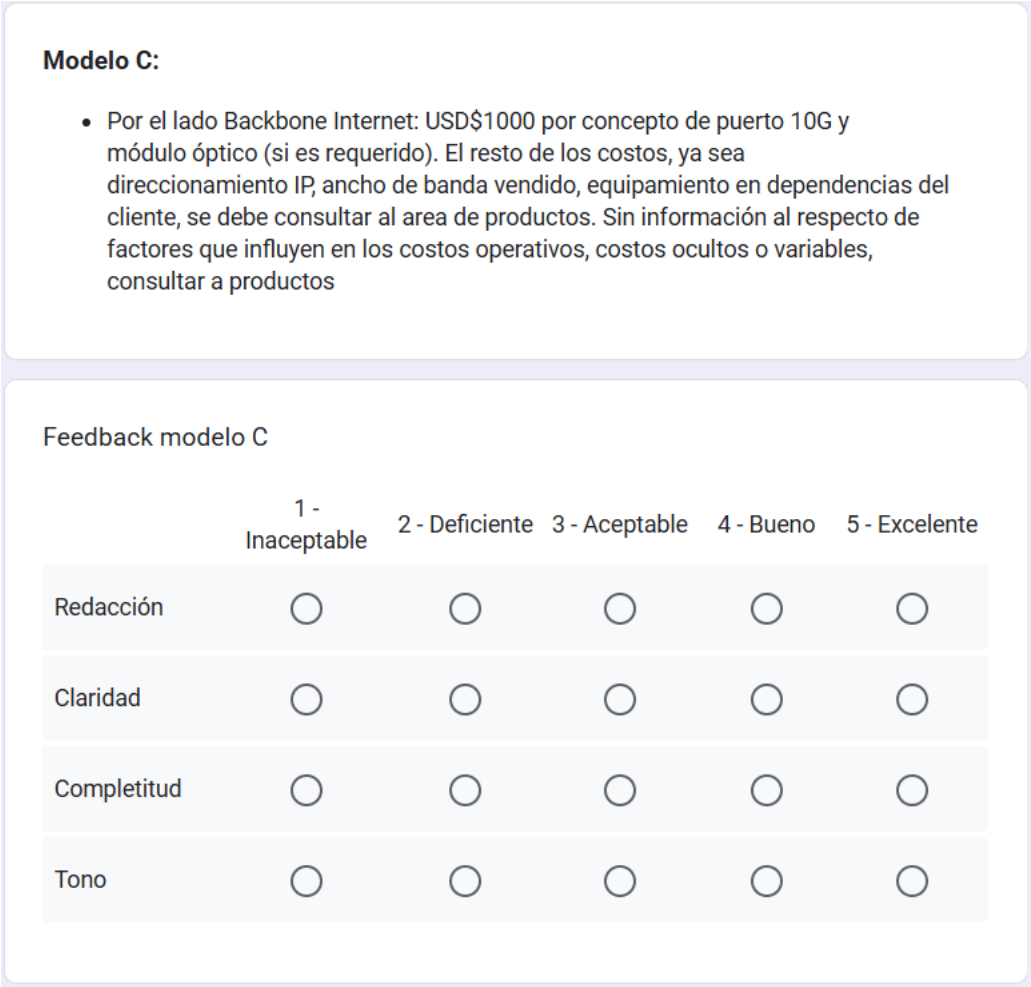
En resumen, el producto Plan Internet Grandes Clientes se compone de componentes que se integran entre sí para proporcionar un servicio de Internet dedicado a medianas y grandes empresas, utilizando tecnologías y plataformas de Juniper y Netcracker.
"

Feedback modelo B

	1 - Inaceptable	2 - Deficiente	3 - Aceptable	4 - Bueno	5 - Excelente
Redacción	○	○	○	○	○
Claridad	○	○	○	○	○
Completitud	○	○	○	○	○
Tono	○	○	○	○	○

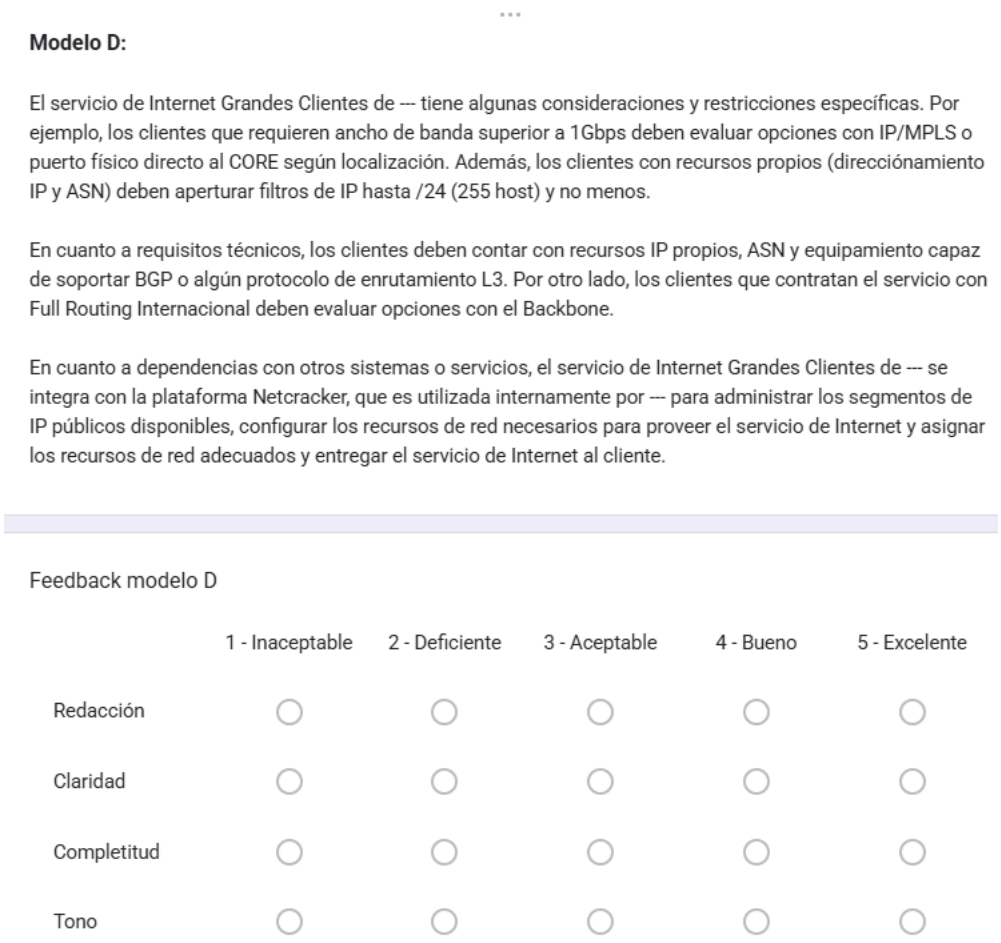
Fuente: Elaboración propia

Figura 9.21: Ítem "Costo del servicio (CAPEX/OPEX)" ejemplo de evaluación Gemini.



Fuente: Elaboración propia

Figura 9.22: Ítem "Consideraciones generales" ejemplo de evaluación agente.



Modelo D:

El servicio de Internet Grandes Clientes de --- tiene algunas consideraciones y restricciones específicas. Por ejemplo, los clientes que requieren ancho de banda superior a 1Gbps deben evaluar opciones con IP/MPLS o puerto físico directo al CORE según localización. Además, los clientes con recursos propios (direcciónamiento IP y ASN) deben aperturar filtros de IP hasta /24 (255 host) y no menos.

En cuanto a requisitos técnicos, los clientes deben contar con recursos IP propios, ASN y equipamiento capaz de soportar BGP o algún protocolo de enrutamiento L3. Por otro lado, los clientes que contratan el servicio con Full Routing Internacional deben evaluar opciones con el Backbone.

En cuanto a dependencias con otros sistemas o servicios, el servicio de Internet Grandes Clientes de --- se integra con la plataforma Netcracker, que es utilizada internamente por --- para administrar los segmentos de IP públicos disponibles, configurar los recursos de red necesarios para proveer el servicio de Internet y asignar los recursos de red adecuados y entregar el servicio de Internet al cliente.

Feedback modelo D

	1 - Inaceptable	2 - Deficiente	3 - Aceptable	4 - Bueno	5 - Excelente
Redacción	○	○	○	○	○
Claridad	○	○	○	○	○
Completitud	○	○	○	○	○
Tono	○	○	○	○	○

Fuente: Elaboración propia

UNIVERSIDAD DE CONCEPCIÓN – FACULTAD DE INGENIERÍA
RESUMEN DE MEMORIA DE TÍTULO

Departamento	: Departamento de Ingeniería Industrial
Carrera	: Ingeniería Civil Industrial
Nombre del memorista	: Rodrigo Santiago Espinoza Bustamante
Título de la memoria	: Optimización en proceso de creación de productos usando grandes modelos de lenguaje (LLM) y agentes con aplicación a productos de una empresa de telecomunicaciones.
Fecha de la presentación oral	:
Profesor Guía	: Juan Carlos Caro Seguel
Profesor(es) Revisor(es)	: Carlos Camilo Navarrete Lizama
Concepto	:
Calificación	:

Resumen (máximo 200 palabras)

La generación automática de documentos utilizando inteligencia artificial generativa se presenta como una herramienta para optimizar tiempos y recursos en organizaciones. Este trabajo desarrolló y validó una solución basada en grandes modelos de lenguaje (LLM) para automatizar la creación de documentos técnicos mediante plantillas personalizadas para la organización. La solución se presenta en Python integrando el framework Langchain y llamadas a las API's de Google Gemini y Groq. Se realizaron diferentes configuraciones con los modelos "Gemini-2.0-flash", "Llama3-70b", "Llama3-8b" y un agente inteligente impulsado con "Llama3-8b". Cada iteración fue evaluada con evaluación humana, relevancia semántica medida con BETO, tiempos de ejecución y ahorros de costos y tiempos para el área. Los resultados muestran que todos los modelos alcanzaron una relevancia semántica alta (Promedio BETO > 0,92), sin embargo, el agente utilizando "Llama3-8b" mostró el mejor desempeño con puntuación de 4,2/5 de los product managers, 88,5% de completitud global de la tarea, ahorro de 18 días de trabajo y un ahorro monetario estimado en $131.948 CLP. No obstante, presentó el mayor tiempo de ejecución (1833 s), frente a menos de 10 minutos para los demás modelos. Se recomienda incorporar herramientas que permitan personalizar los documentos con gráficos, tablas y personalización de prompts.