



Universidad Nacional Mayor de San Marcos

Universidad del Perú. Decana de América

Facultad de Ingeniería de Sistemas e Informática

Escuela Profesional de Ingeniería de Software

Plataforma digital con IAG y RAG para la automatización de valoración de autos

TRABAJO DE SUFICIENCIA PROFESIONAL

Para optar el Título Profesional de Ingeniero de Software

AUTOR

Jenny Lucia VEGA GUERRERO

ASESOR

Dr. Javier Arturo GAMBOA CRUZADO

Lima, Perú

2025



Reconocimiento - No Comercial - Compartir Igual - Sin restricciones adicionales

<https://creativecommons.org/licenses/by-nc-sa/4.0/>

Usted puede distribuir, remezclar, retocar, y crear a partir del documento original de modo no comercial, siempre y cuando se dé crédito al autor del documento y se licencien las nuevas creaciones bajo las mismas condiciones. No se permite aplicar términos legales o medidas tecnológicas que restrinjan legalmente a otros a hacer cualquier cosa que permita esta licencia.

Referencia bibliográfica

Vega, J. (2025). *Plataforma digital con IAG y RAG para la automatización de valoración de autos*. [Trabajo de suficiencia profesional de pregrado, Universidad Nacional Mayor de San Marcos, Facultad de Ingeniería de Sistemas e Informática, Escuela Profesional de Ingeniería de Software]. Repositorio institucional Cybertesis UNMSM.

Metadatos complementarios

Datos de autor	
Nombres y apellidos	JENNY LUCIA VEGA GUERRERO
Tipo de documento de identidad	DNI
Número de documento de identidad	70126281
URL de ORCID	https://orcid.org/0009-0003-2184-4800
Datos de asesor	
Nombres y apellidos	JAVIER ARTURO GAMBOA CRUZADO
Tipo de documento de identidad	DNI
Número de documento de identidad	17906323
URL de ORCID	https://orcid.org/0000-0002-0461-4152
Datos del jurado	
Presidente del jurado	
Nombres y apellidos	LUZMILA ELISA PRÓ CONCEPCIÓN
Tipo de documento	DNI
Número de documento de identidad	08862360
Miembro del jurado 1	
Nombres y apellidos	NORBERTO ULISES ROMAN CONCHA
Tipo de documento	DNI
Número de documento de identidad	08799230
Datos de investigación	
Línea de investigación	No aplica
Grupo de investigación	No aplica
Agencia de financiamiento	Sin financiamiento
Ubicación geográfica de la investigación	País: Perú Departamento: Lima Provincia: Lima

	Distrito: Cercado de Lima Jr. Carlos Amezaga No. 375 Universidad Nacional Mayor de San Marcos Latitud: -12.0564232 Longitud: -77.0843327
Año o rango de años en que se realizó la investigación	2024
URL de disciplinas OCDE	2.02.04 -- Ingeniería de sistemas y comunicaciones https://purl.org/pe-repo/ocde/ford#2.02.04



UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS
FACULTAD DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
Escuela Profesional de Ingeniería de Software

**Acta Virtual de Sustentación
del Trabajo de Suficiencia Profesional**

Siendo las 16:00 horas del día 15 de enero del año 2025, se reunieron virtualmente los docentes designados como Miembros de Jurado del Trabajo de Suficiencia Profesional, presidido por la Dra. Pró Concepción Luzmila Elisa (Presidente), Lic. Román Concha Norberto Ulises (Miembro) y el Dr. Gamboa Cruzado Javier Arturo (Miembro Asesor), usando la plataforma Meet (<https://meet.google.com/seg-hqdz-yxf>), para la sustentación virtual del Trabajo de Suficiencia Profesional intitulado: **“PLATAFORMA DIGITAL CON IAG Y RAG PARA LA AUTOMATIZACIÓN DE VALORACIÓN DE AUTOS”**, por la Bachiller **Vega Guerrero Jenny Lucia**; para obtener el Título Profesional de Ingeniero de Software.

Acto seguido de la exposición del Trabajo de Suficiencia Profesional, el Presidente invitó a la Bachiller a dar las respuestas a las preguntas establecidas por los miembros del Jurado.

La Bachiller en el curso de sus intervenciones demostró pleno dominio del tema, al responder con acierto y fluidez a las observaciones y preguntas formuladas por los señores miembros del Jurado.

Finalmente habiéndose efectuado la calificación correspondiente por los miembros del Jurado, la Bachiller obtuvo la nota de **18 (DIECIOCHO)**.

A continuación la Presidente de Jurados la Dra. Pró Concepción Luzmila Elisa, declara a la Bachiller **Ingeniero de Software**.

Siendo las 16:55 horas, se levantó la sesión.

Presidente

Dra. Pró Concepción Luzmila Elisa

Miembro

Lic. Román Concha Norberto Ulises

Miembro Asesor

Dr. Gamboa Cruzado Javier Arturo



CERTIFICADO DE SIMILITUD

Yo, GAMBOA CRUZADO JAVIER ARTURO en mi condición de asesor acreditado con la Resolución Decanal N° 001059-2024-D-FISI/UNMSM y Oficio N° 0015-PTTSP-FISI-2024 del Trabajo de Suficiencia Profesional cuyo título es Plataforma Digital con IAG y RAG para la Automatización de valoración de autos, presentado por el bachiller VEGA GUERRERO JENNY LUCIA para optar el título de Ingeniero de Software CERTIFICO que se ha cumplido con lo establecido en la Directiva de Originalidad y de Similitud de Trabajos Académicos, de Investigación y Producción Intelectual. Según la revisión, análisis y evaluación mediante la plataforma PIDS, el documento evaluado cuenta con el 7% de similitud, nivel **PERMITIDO** para continuar con los trámites correspondientes y para su **publicación en el repositorio institucional**.

Se emite el presente certificado en cumplimiento de lo establecido en las normas vigentes, como uno de los requisitos para la obtención del título.

Lima, 8 de enero de 2025

Firma del Asesor

Nombres y Apellidos del asesor: GAMBOA CRUZADO JAVIER ARTURO

DNI 17906323

AGRADECIMIENTOS

Agradezco a Dios por siempre estar conmigo,
por darme fortaleza para seguir adelante
todo el tiempo.

A mi madre, por siempre apoyarme y confiar en
mis sueños.

UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS
FACULTAD DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
ESCUELA PROFESIONAL DE INGENIERÍA DE SOFTWARE

Plataforma Digital con IAG y RAG para la Automatización de valoración de autos

Autora: Bach. Vega Guerrero, Jenny Lucia

Asesor: Dr. Ing. Gamboa Cruzado, Javier Arturo

Título: Trabajo de Suficiencia Profesional para optar el Título Profesional de Ingeniería de Software

Fecha: Diciembre de 2024

RESUMEN

Las ventas de autos usados están aumentando exponencialmente, y con esto surge la necesidad del uso de plataformas digitales para hacer del proceso de venta una experiencia online rápida y segura. En este contexto, existen dos actores: el usuario que desea vender su auto, y el usuario que quiere comprarlo. Asignar el precio correcto considerando las características del vehículo, así como los daños declarados es esencial para mantener la confianza de ambas partes. Los sistemas tradicionales de valoración de autos que utilizan inteligencia artificial se centran en evaluar únicamente características como marca, modelo, año de manufactura, kilometraje, tipo de combustible, sistema de frenos, etc., y algunos más avanzados analizan imágenes usando deep learning. Sin embargo, es difícil encontrar un precio correcto sin conocer las condiciones en las que el auto se encuentra, ya que estos sistemas no evalúan los daños mecánicos o eléctricos que el vehículo podría presentar. La plataforma digital de la empresa permite a los usuarios que desean vender su vehículo, declarar mediante texto en lenguaje natural los daños mecánicos y/o eléctricos que el automóvil presenta. El desafío ahora reside en entender lo que el usuario ha ingresado y asignar un precio en base a la descripción del daño vehicular. La ayuda de un agente humano es primordial y la primera respuesta a este reto, sin embargo, esto genera una carga de trabajo mayor para ellos. Para resolver este nuevo problema, la solución planteada en este estudio utiliza inteligencia artificial generativa con técnicas como RAG y fundamentación para analizar el texto ingresado por el usuario, encontrar la clase o tipo de daño en una base de datos, y asignar el precio correspondiente a uno o más artículos declarados. La implantación del proyecto y las primeras proyecciones sostienen que la solución propuesta tiene una alta rentabilidad, logrando reducir el tiempo promedio de valoración en un 200%, manteniendo un nivel de precisión del 87% con respecto a la evaluación realizada por agentes humanos.

Palabras clave: Mercado online de autos, IA Generativa, bases de datos de vectores, modelos grandes de lenguaje

UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS
FACULTY OF SYSTEMS AND INFORMATICS ENGINEERING
PROFESSIONAL

PROFESSIONAL SCHOOL OF SYSTEMS ENGINEERING

Using Generative Artificial Intelligence and Retrieval Augmented Generation for
automating a car valuation process on a digital platform

Author: Bach. Vega Guerrero, Jenny Lucia

Advisor: Dr. Gamboa Cruzado, Javier Arturo

Title: Professional Sufficiency Work to opt for the Professional Title of Software Engineer

Date: December 2024

ABSTRACT

Used car sales are increasing exponentially, and with this arises the need to use digital platforms to make the sales process a fast and safe online experience. In this context, there are two actors: the user who wants to sell their car, and the user who wants to buy it. Assigning the correct price considering the characteristics of the vehicle, as well as the declared damages, is essential to maintain the trust of both parties. Traditional car valuation systems that use artificial intelligence focus on evaluating only characteristics such as brand, model, year of manufacture, mileage, fuel type, brake system, etc., and some more advanced ones analyze images using deep learning. However, it is difficult to find a correct price without knowing the condition of the car, since these systems do not evaluate the mechanical or electrical damages that the vehicle could present. The company's digital platform allows users who want to sell their vehicle to declare, using text in natural language, the mechanical and/or electrical damages that the car presents. The challenge now lies in understanding what the user has entered and assigning a price based on the description of the vehicle damage. The help of a human agent is essential and the first response to this challenge. However, this generates a greater workload for them. To solve this new problem, the solution proposed in this study uses generative artificial intelligence with techniques such as RAG and substantiation to analyze the text entered by the user, find the class or type of damage in a database, and assign the corresponding price to one or more declared items. The implementation of the project and the first projections maintain that the proposed solution has a high profitability, managing to reduce the average assessment time by 200%, maintaining a level of precision of 87% with respect to the assessment carried out by human agents.

Keywords: Car marketplace, Generative AI, vector databases, large language models

TABLA DE CONTENIDOS

INTRODUCCIÓN	1
CAPÍTULO I – TRAYECTORIA PROFESIONAL.....	3
CAPÍTULO II – CONTEXTO EN EL QUE SE DESARROLLÓ LA EXPERIENCIA	8
2.1 EMPRESA – ACTIVIDAD QUE REALIZA.....	8
2.2 VISIÓN	8
2.3 MISIÓN.....	8
2.4 ORGANIZACIÓN DE LA EMPRESA	9
2.5 ÁREA, CARGO Y FUNCIONES DESEMPEÑADAS.....	9
2.5.1 ÁREA	9
2.5.2 CARGO	10
2.5.3 FUNCIONES DESEMPEÑADAS	10
CAPÍTULO III - ACTIVIDADES DESARROLLADAS	11
3.1 SITUACIÓN PROBLEMÁTICA	11
3.1.1 DEFINICIÓN DEL PROBLEMA	12
3.2 SOLUCIÓN.....	13
3.2.1. OBJETIVOS.....	13
3.2.2. ALCANCE	14
3.2.3. ETAPAS Y METODOLOGÍAS	14
3.2.4 FUNDAMENTOS UTILIZADOS	15
3.2.5 IMPLEMENTACIÓN DE LAS ÁREAS, PROCESOS, SISTEMAS Y BUENAS PRÁCTICAS	37
3.3 EVALUACIÓN ECONÓMICA.....	76
3.3.1 ANÁLISIS DE COSTOS	76
3.3.2 ANÁLISIS DE BENEFICIOS	77
3.3.3 EVALUACIÓN DEL RESULTADO.....	78
3.3.4 INTERPRETACIÓN DEL VAN Y DEL TIR.....	78
CAPÍTULO IV - REFLEXIÓN CRÍTICA DE LA EXPERIENCIA.....	80
CAPÍTULO V – CONCLUSIONES Y RECOMENDACIONES	82
5.1 CONCLUSIONES	82
5.2 RECOMENDACIONES.....	83
5.3 FUENTES DE INFORMACIÓN.....	84
ANEXOS.....	86
<i>Anexo 1: Cronograma a alto nivel.....</i>	<i>86</i>
<i>Anexo 2: Requerimiento creado en Salesforce para iniciar el proyecto.....</i>	<i>87</i>

ÍNDICE DE FIGURAS

Figura 1	<i>Esquema organizativo de la empresa</i>	9
Figura 2	<i>Flujo de conversión de vehículos</i>	12
Figura 3	<i>Metodología Kanban</i>	17
Figura 4	<i>Visualización cronológica de los lanzamientos de LLM</i>	19
Figura 5	<i>Word embeddings, representación de palabras en vectores multidimensionales</i>	21
Figura 6	<i>Codificación posicional</i>	23
Figura 7	<i>Arquitectura de un Transformer</i>	25
Figura 8	<i>Generación de texto usando autorregresión</i>	26
Figura 9	<i>Componentes de una instrucción</i>	28
Figura 10	<i>Evaluación de LLMs</i>	31
Figura 11	<i>Flujograma que muestra como interactúan los actores principales en el proceso de venta de vehículos</i>	38
Figura 12	<i>Etapas del Modelo de Contratación</i>	39
Figura 13	<i>Tablero Kanban en MIRO</i>	42
Figura 14	<i>Notas diarias durante el desarrollo del proyecto</i>	43
Figura 15	<i>Arquitectura conceptual de la solución</i>	44
Figura 16	<i>Arquitectura tecnológica de la solución</i>	44
Figura 17	<i>Pasos seguidos para completar Entregable 1</i>	45
Figura 18	<i>Estructura del proyecto en BigQuery</i>	46
Figura 19	<i>Código para crear embeddings usando VertexAI</i>	50
Figura 20	<i>Código para cargar los embeddings a Pinecone</i>	50
Figura 21	<i>Análisis del índice calidad de modelos</i>	52
Figura 22	<i>Análisis de tokens de salida por segundo</i>	52
Figura 23	<i>Análisis de precios</i>	53
Figura 24	<i>Flujo de razonamiento que debe ser seguido por el LLM</i>	56
Figura 25	<i>Código que crea y almacena los vectores en la base de datos de vectores</i>	58
Figura 26	<i>Código que utiliza un LLM para determinar si existe declaración de daño vehicular</i>	59
Figura 27	<i>Código que utiliza un LLM para determinar encontrar cualquier daño declarado en un texto</i>	60
Figura 28	<i>Código que utiliza embeddings de texto y una base de datos de vectores para encontrar categorías</i>	61
Figura 29	<i>Pasos seguidos para completar Entregable 2</i>	62
Figura 30	<i>Código que estima los costos de reparación de un vehículo</i>	63
Figura 31	<i>Código que permite fundamentar la respuesta de un LLM</i>	65
Figura 32	<i>Código que permite extraer preguntas</i>	66
Figura 33	<i>Código que genera un presupuesto total de reparación de daños vehiculares declarados</i>	67
Figura 34	<i>Código que permite extraer el precio final de reparación de los daños vehiculares declarados</i>	68
Figura 35	<i>API para carga en batch</i>	69
Figura 36	<i>YAML utilizado por Github Actions</i>	70
Figura 37	<i>Configuración para el uso de Cloud Run</i>	71
Figura 38	<i>Vista “Debug”</i>	72
Figura 39	<i>Vista “Review”</i>	74

Figura 40	<i>Vista “Agent”</i>	75
Figura 41	Ventas mensuales en el último año	78
Figura 42	<i>Relación VAN y TIR</i>	79

ÍNDICE DE TABLAS

Tabla 1 Experiencias laborales	3
Tabla 2 Educación Académica	5
Tabla 3 Cursos realizados	6
Tabla 4 Certificaciones	6
Tabla 5 Idiomas estudiados por la autora	7
Tabla 6 Roles que participaron en el proyecto	14
Tabla 7 Etapas del Proyecto.....	39
Tabla 8 Listado de entregables del proyecto	40
Tabla 9 Descripción de las tablas utilizadas en el Entregable 1	47
Tabla 10 Descripción de campos relevantes de las tablas utilizados en el Entregable 1	47
Tabla 11 Listado de modelos de Gemini y características principales	53
Tabla 12 Parámetros de la base de datos de vectores en Pinecone	57
Tabla 13 Costo total de implementación del proyecto	76

INTRODUCCIÓN

La forma en la que vendemos objetos de valor en una plataforma digital ha cambiado de manera importante en los últimos años alrededor del mundo y el mercado automovilístico está creciendo rápidamente en el medio digital. La confianza que generan estas plataformas y la experiencia que brindan a sus usuarios es un factor clave para aumentar la cantidad de ventas de vehículos de segunda mano.

La empresa objeto de la presente investigación necesitaba rediseñar su flujo de perfilamiento de clientes para mejorar la experiencia de sus usuarios y reducir la carga operativa que generan los procesos manuales en los agentes de la empresa cuya función es valorizar los vehículos.

Con esta prioridad, se desarrolló una solución que analiza texto en lenguaje natural ingresado por un usuario a través de la plataforma digital y permite extraer la información correspondiente a los daños vehiculares declarados en dicho texto. Esta solución mejora de manera significativa la experiencia del usuario que desea vender su vehículo y reduce la cantidad de llamadas telefónicas que realiza el agente para obtener más información sobre la declaración de daños.

Para la implementación de este proyecto se realizaron 3 entregables, la primera permitió clasificar el texto ingresado por el usuario en una o más categorías definidas previamente por la empresa. La segunda entrega correspondió a un modelo de lenguaje natural con instrucciones que permite calcular el costo de arreglar los artículos declarados como dañados y/o el problema eléctrico o mecánico del texto de entrada. Y la tercera entrega consistió en ajustar el modelo de lenguaje natural de la entrega anterior y hacerlo más “sensible” a los precios considerando información adicional.

A continuación, se describe brevemente los capítulos que organizan el contenido del trabajo desarrollado:

- En el Capítulo I se describe la trayectoria profesional de la autora, los conocimientos adquiridos y su experiencia laboral.
- En el Capítulo II se presentan los antecedentes y el contexto en el que se desarrolla esta investigación.
- En el Capítulo III se detalla el proceso actual de valoración de autos, los

retos que presenta, así como los objetivos, las bases teóricas revisadas, el desarrollo e implementación de la solución planteada en este estudio.

- En el Capítulo IV se ofrece una reflexión crítica sobre la carrera profesional y sus contribuciones en esta investigación, las herramientas y conocimientos obtenidos, junto con un análisis de las estadísticas que respaldan el éxito del uso de inteligencia artificial generativa para automatizar procesos de valoración de vehículos.
- Finalmente, en el Capítulo V, se proporcionan las conclusiones y sugerencias.

CAPÍTULO I – TRAYECTORIA PROFESIONAL

La autora se desempeña como arquitecta senior de inteligencia artificial, con experiencia en el diseño y desarrollo de soluciones que requieren el uso de técnicas avanzadas de procesamiento y análisis de datos, modelos de inteligencia artificial y puesta en producción de estos.

Actualmente desarrolla sus labores en Google liderando proyectos con clientes de la compañía en el Reino Unido.

Anteriormente la autora trabajó para empresas de diferentes tamaños e industrias. Los detalles se describen en la Tabla 1:

Tabla 1

Experiencias laborales

Empresa/Periodo	Descripción de actividades realizadas
Google (Julio 2024-Actualidad)	Cargo: Sr. Field Solutions Architect – Generative AI Actividades: Diseñar y construir prototipos rápidos de aplicaciones y soluciones end-to-end de IA generativa para clientes prioritarios de la compañía.
Amazon Web Services (Abril 2022 – Junio 2024)	Cargo: Sr. Solutions Architect – Core AI Startups Actividades: Proporcionar asesoría técnica a Startups que tienen Inteligencia Artificial como componente principal. Desarrollo de código, blogs, implementaciones de referencia y presentaciones para permitir que los clientes y socios de AWS aprovechen al máximo el aprendizaje automático y la inteligencia artificial. Logros: - Arquitecta de alto rendimiento 2023 Q3-Q4. - Líder de laboratorio práctico en ReInvent 2023. - Ponente en conferencia “EMEA Innovate Q1” 2024 - Ponente en AWS Cloud Day Zurich 2023

- Ponente en conferencia “EMEA Innovate Q3” 2023
- Evangelización técnica: “AWS Cost Optimisation Series”
- Ponente en Africa Tech Summit 2023
- Ponente en AWS London Summit 2023.
- Reconocimiento: Best deal Startups EMEA Q2 Startups EMEA.
- Ponente en sesión técnica en ReInvent 2022

Globant
(Marzo 2021 – Enero 2022)

Cargo: Sr. Data Architect

Actividades: Analizar diferentes casos de negocio, proponer soluciones basadas en datos, crear productos o construir plataformas de datos.

Logros:

- Implementación de catálogo de datos (Cliente: MetLife US)
- Migración GCP a Azure (Cliente: PeopleReady US)

Rappi
(Agosto 2020 – Febrero 2021)

Cargo: Sr. Machine Learning Engineer

Actividades: Automatización de los procesos de extracción, procesamiento y análisis de datos, desarrollo de modelos utilizando Airflow, Kubernetes y Python para machine learning.

Logros:

- Implementación de proyecto: ETA restaurantes. Reducción de latencia de 3s a 0.07s.
- Líder de proyecto: Sistema de recomendación de restaurantes.
- Implementación de proyecto: análisis de imágenes para clasificación y optimización de catálogos de proveedores.

Rimac Seguros y Reaseguros
(Febrero 2018 – Febrero 2020)

Cargo: Machine Learning Engineer

Actividades: Migraciones de datos de centro de datos locales a servicios de nube (AWS). Creción de flujos de procesamiento de datos en AWS. Implementación de soluciones con técnicas de machine learning para proyecto de visión por computadora y procesamiento de lenguaje natural.

Logros:

- Diseño, desarrollo y prueba de un flujo de extremo a extremo para modelos de aprendizaje automático y aprendizaje profundo (visión artificial). Creación de estándares en el desarrollo del modelo para el equipo de científicos de datos. Investigación e implementación de nuevos enfoques de aprendizaje automático y aprendizaje profundo relacionados con la industria de seguros.
- Desarrollo de un modelo de clustering utilizando K-Means que permitió encontrar el(los) candidato(s) más adecuado(s) para la fuerza de ventas de la empresa y la reducción de la rotación de personal en un 45%.
- Análisis y desarrollo de un modelo de churn que permitió la creación de estrategias comerciales para retener el 80% de los clientes de la compañía en un año.
- Proyecto Datalake: migración de datos

La educación académica de la autora se detalla en la Tabla 2:

Tabla 2*Educación Académica*

Período	Educación Académica
2019-2022	Máster Universitario en Inteligencia Artificial Universidad Internacional de la Rioja
2013-2017	Grado Académico de Bachiller en Ingeniería de Software Escuela Académico Profesional de Ingeniería de Software Facultad de Ingeniería de Sistemas e Informática Universidad Nacional Mayor de San Marcos

Los cursos realizados por la autora se muestran en la Tabla 3:

Tabla 3

Cursos realizados

Institución	Curso
BlueDot Impact	AI Safety
Stanford University	Machine Learning CS229
DeepLearning AI	Large Language Models with Semantic Search
DeepLearning AI	Machine Learning in Production
IBM	Qiskit Global Summer School on Quantum Machine Learning
MITx	Micromaster in Data Science

Las certificaciones logradas por la autora se reseñan en la Tabla 4:

Tabla 4

Certificaciones

Periodo	Certificación	Institución
2022	Machine Learning Specialist	AWS
2022	SysOps Administrator Associate	AWS
2022	Developer Associate	AWS
2022	Security, Compliance, and Identity Fundamentals	Microsoft
2021	Power Platform Fundamentals	Microsoft
2021	Azure Data Fundamentals	Microsoft
2021	Azure AI Fundamentals	Microsoft
2021	Azure Fundamentals	Microsoft

2021	Cloud Practitioner	AWS
2020	Solutions Architect Associate	AWS

Los estudios de idiomas realizados por la autora se listan en la Tabla 5:

Tabla 5

Idiomas estudiados por la autora

Periodo	Idioma	Nivel	Institución
2008-2010	Inglés	C1	ICPNA

CAPÍTULO II – CONTEXTO EN EL QUE SE DESARROLLÓ LA EXPERIENCIA

2.1 EMPRESA – ACTIVIDAD QUE REALIZA

La compañía donde se desarrolló la experiencia es un mercado de autos usados exclusivamente online con sede en el Reino Unido. Esta empresa cuenta con una plataforma digital (incluido móvil y web) que funciona para clientes particulares que venden automóviles y distribuidores verificados que compran dichos automóviles.

El portal del cliente permite a los usuarios recibir una valoración del vehículo compartiendo la matrícula y el kilometraje de su automóvil. La plataforma proporciona esta cifra basándose en datos del mercado y las tendencias actuales del mercado. Cuando el automóvil aparece en la plataforma de ventas, los concesionarios (distribuidores) verificados pueden realizar sus ofertas. Luego, el automóvil se recoge para su entrega al concesionario que realiza la oferta más alta.

El portal de concesionarios está restringido a perfiles profesionales. Los distribuidores reciben una notificación de los automóviles una vez que están disponibles y realizan ofertas en línea diariamente. Cuando los distribuidores compran vehículos, pueden organizar la recogida de este con el servicio de transporte de la empresa, así como también comprar mediante la plataforma de pago en tiempo real.

2.2 VISIÓN

La empresa de este estudio define su visión como la de construir un mejor mercado automovilístico para todos, aprovechando el poder de la tecnología para ofrecer la mejor experiencia en el sector.

2.3 MISIÓN

Para lograr con la visión descrita anteriormente, la empresa busca facilitar la venta de automóviles de manera rápida, sencilla y al mejor precio, permitiendo

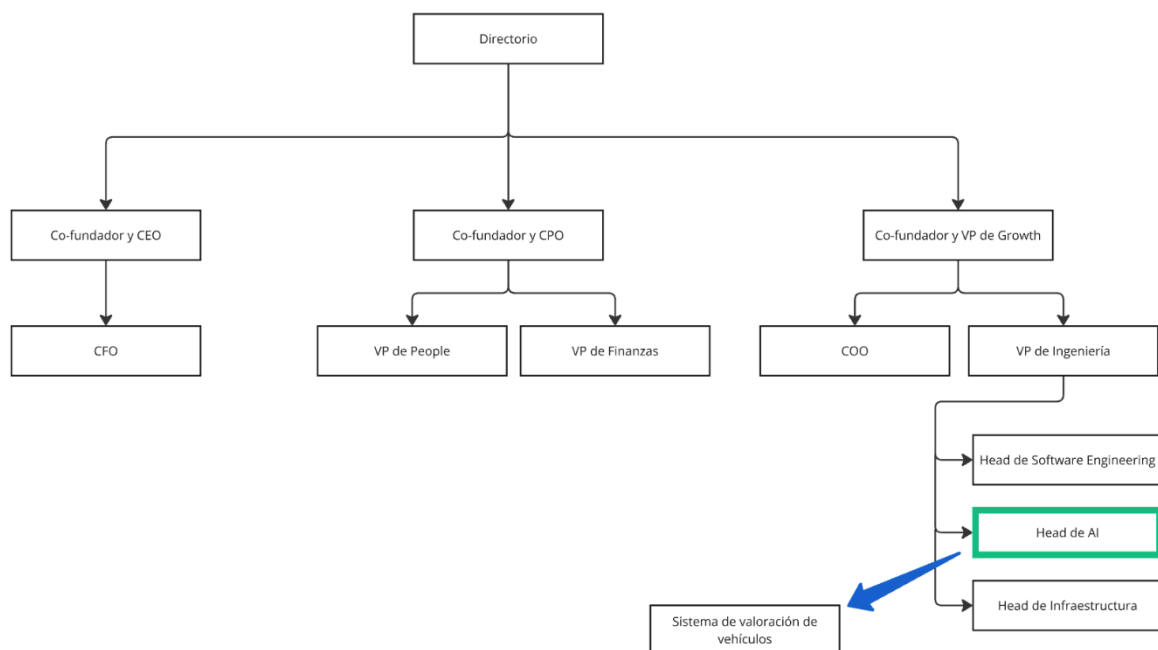
a los usuarios gestionar todo el proceso desde la comodidad de su hogar, utilizando únicamente su teléfono móvil.

2.4 ORGANIZACIÓN DE LA EMPRESA

El esquema que muestra la organización de la empresa estudiada se presenta en la Figura 1. El objetivo del esquema es ubicar el área donde se ha desarrollado el proyecto TSP. El área de Ingeniería, específicamente en el equipo de Inteligencia Artificial es donde se implementó el proceso de automatización de valoración de autos.

Figura 1

Esquema organizativo de la empresa



2.5 ÁREA, CARGO Y FUNCIONES DESEMPEÑADAS

2.5.1 ÁREA

Durante el transcurso del proyecto, la autora estuvo empleada por Google, donde se dedicó a diseñar e implementar soluciones innovadoras para abordar problemas complejos mediante el uso de Inteligencia Artificial, enfocándose en empresas del Reino Unido. En este proyecto específico, trabajó en estrecha colaboración con el equipo de Inteligencia Artificial del departamento

de ingeniería, asegurando un enfoque integral y efectivo.

2.5.2 CARGO

Como Arquitecta de Soluciones de Inteligencia Artificial, la autora desempeñó un rol clave como Líder de Proyecto e ingeniera de software. Sus responsabilidades incluyeron dirigir el proyecto y aplicar sus habilidades técnicas para desarrollar soluciones efectivas, combinando liderazgo y experiencia técnica.

2.5.3 FUNCIONES DESEMPEÑADAS

La autora desempeñó las siguientes funciones:

- Definir el alcance del proyecto.
- Dirigir las reuniones diarias y semanales con el equipo técnico del cliente (empresa en la que se desarrolló este estudio), así como con los actores principales (stakeholders).
- Determinar los requisitos funcionales y no funcionales, determinando el tiempo requerido para el desarrollo del proyecto.
- Desarrollar los componentes en cada etapa del proyecto.
- Diseñar e implementar pruebas técnicas y funcionales.
- Crear el documento técnico de entrega del proyecto

CAPÍTULO III - ACTIVIDADES DESARROLLADAS

3.1 SITUACIÓN PROBLEMÁTICA

En la actualidad, tecnologías como el aprendizaje automático o machine learning están transformando diversas industrias, incluida la automotriz. Esta herramienta desempeña un papel fundamental en la estimación precisa de los precios de los vehículos. Al procesar grandes volúmenes de datos, los modelos de machine learning son capaces de predecir los precios de los vehículos considerando factores como el modelo, el año, el kilometraje y otros elementos. Gracias a esta tecnología, tanto compradores como vendedores de vehículos pueden tomar decisiones más informadas, optimizando tiempo y recursos.

La compañía estudiada tiene un historial comprobado de uso de Inteligencia Artificial para impulsar el éxito en áreas como visión artificial (AI Image Assist), fijación de precios de vehículos, sistemas de recomendación y transporte utilizando técnicas de machine learning. Su estrategia “AI First” tiene como objetivo utilizar IA en todos los equipos para mejorar la productividad y la innovación de productos, con un fuerte énfasis en la IA generativa. Las principales iniciativas impulsadas por IA incluyen la creación de perfiles de vehículos con tecnologías como IA Generativa, agentes de IA y herramientas personalizadas para concesionarios.

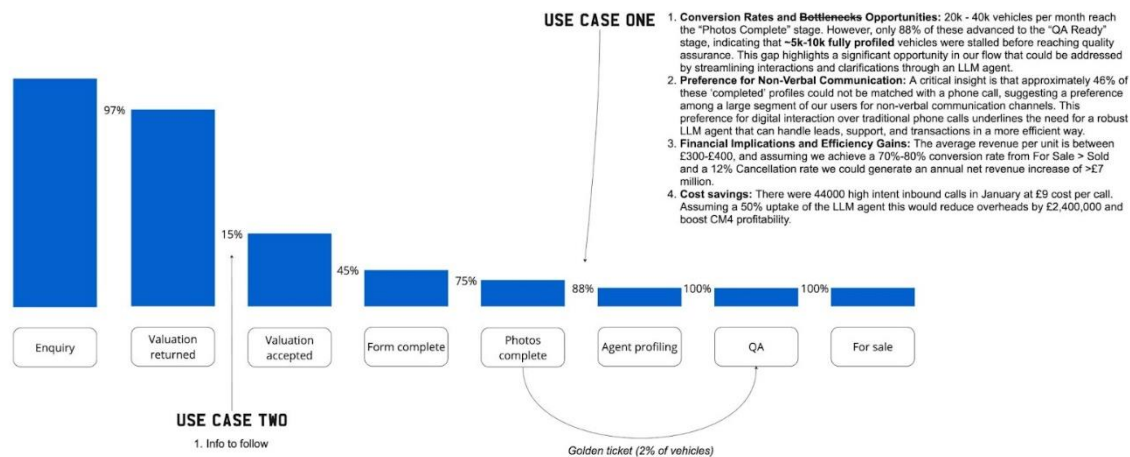
Para seguir mejorando sus capacidades, la empresa reconoció la oportunidad de revolucionar su proceso de recopilación de información sobre el estado y daños de los vehículos de sus clientes. La empresa determinó un total de 30.674 vehículos que alcanzaron la etapa de “Formulario y fotos completos” en enero. Sin embargo, solo 23.671 de ellos avanzaron a la etapa de “QA Ready”, lo que indica que 7.003 vehículos se detuvieron antes de llegar al control de calidad. Esta brecha resalta una oportunidad significativa en su flujo que podría abordarse agilizando las interacciones y aclaraciones a través de un agente de modelo grande lenguaje (LLM).

En la Figura 2 se explica el flujo de conversión de venta de vehículos, indicando el porcentaje de usuarios que permanecen o continúan en el proceso después de cada fase. El presente TSP aborda la problemática que surge en la fase llamada “Agent profiling”. Cuando un usuario vendedor completa información en la plataforma de la compañía, además de ingresar datos del

vehículo (marca, modelo, año, entre otros), necesita declarar (si existieran) los daños de su vehículo. Cuando el usuario envía toda la información, si ha declarado daño alguno en el vehículo, un agente de la empresa se contacta telefónicamente para obtener más detalles sobre los daños y reparaciones necesarias en el vehículo. Esto reduce drásticamente la experiencia digital del cliente que la empresa quiere ofrecer. Con la información obtenida, el agente clasifica los problemas que el vehículo presenta y añade un precio por cada artículo o reparación declarada con el fin de obtener un precio más justo cuando el vehículo se ofrece en la plataforma de usuarios concesionarios.

Un dato crítico es que aproximadamente el 46% de estos perfiles “completados” no pudieron ser emparejados con una llamada telefónica, lo que sugiere una preferencia entre un gran segmento de nuestros usuarios por los canales de comunicación no verbal. Esta preferencia por la interacción digital en lugar de las llamadas telefónicas tradicionales subraya la necesidad de un agente LLM que pueda manejar clientes potenciales, brinde soporte y gestione transacciones de una manera más eficiente.

Figura 2
Flujo de conversión de vehículos



3.1.1 DEFINICIÓN DEL PROBLEMA

El problema que enfrenta la empresa se centra en la necesidad de mejorar su proceso de perfilamiento de clientes, particularmente en el flujo relacionado con la declaración de daños mecánicos y/o eléctricos en los vehículos. Este

proceso no solo afecta la eficiencia, sino que también impacta negativamente la personalización y la interacción con el cliente, objetivos clave que la empresa busca mejorar mediante la implementación de un flujo conversacional generativo.

3.2 SOLUCIÓN

Para abordar el desafío del trabajo de clasificación manual que realizan los agentes de la empresa objeto de este estudio y la asignación de precio para cada artículo dañado, la solución propuesta fue crear un algoritmo que pueda hacer coincidir la descripción de daños de un vehículo de un usuario con una clase específica utilizando un enfoque de IA generativa y RAG (LLM y vectores de texto) así como encontrar un precio para cada artículo que necesite reparación.

3.2.1. OBJETIVOS

3.2.1.1. OBJETIVO GENERAL

Optimizar el tiempo de valoración de vehículos realizados por los agentes, mediante una solución de IA y RAG.

3.2.1.2. OBJETIVOS ESPECÍFICOS

- a) Planificar los entregables siguiendo la metodología Kanban en coordinación con la empresa objeto del estudio.
- b) Diseñar la arquitectura de la solución a nivel de componentes y a nivel de tecnologías y herramientas a utilizar.
- c) Utilizar técnicas de Inteligencia Artificial Generativa para analizar el texto ingresado por el usuario.
- d) Utilizar datos almacenados en fuentes de información como base de datos y documentos de la empresa para aterrizar los precios generados por el modelo de lenguaje natural.

3.2.2. ALCANCE

El alcance del desarrollo cubre la entrega de un producto mínimo viable que permita integrarse rápidamente con la plataforma de la empresa y que cumpla con los objetivos expuestos anteriormente.

3.2.3. ETAPAS Y METODOLOGÍAS

Para el desarrollo de la solución de este proyecto se utilizó la metodología Kanban gestionada por la herramienta MIRO. Debido a la agilidad requerida y a la constante iteración y experimentación la metodología elegida permitió un desarrollo de proyecto eficiente involucrando a todas las partes interesadas. En la etapa de diseño, se plantearon 3 etapas con 1 entregable por cada etapa.

ROLES

Los roles participantes del proyecto están resumidos en la Tabla 6:

Tabla 6

Roles que participaron en el proyecto

Rol	Descripción
Service Delivery Manager	Encargado de supervisar y garantizar el flujo constante de trabajo, la satisfacción del cliente y la entrega oportuna de valor.
Service Request Manager	Encargado de gestionar y priorizar el trabajo que entra en el sistema, asegurando que se atiendan las solicitudes de servicio de manera eficiente y alineada con las necesidades del negocio.
Team Member	Equipo que identifica y prioriza los entregables según los lineamientos. Confirma la implementación de los entregables en el proyecto

Nota. Fundamentado en el manifiesto Agile.

USO DE LA METODOLOGÍA

Kanban es una metodología ampliamente utilizada en proyectos de ingeniería de software. En este proyecto se decidió utilizar la herramienta MIRO, la cual es bastante flexible e intuitiva.

3.2.4 FUNDAMENTOS UTILIZADOS

3.2.4.1. DEFINICIÓN DE METODOLOGÍAS ÁGILES

Las metodologías ágiles son un conjunto de técnicas de gestión de proyectos que tienen como características:

- Dividir el proyecto en fases
- Enfatizar la colaboración y la mejora continua
- Seguir un ciclo de planificación, ejecución y evaluación
- Usar ciclos de trabajo cortos
- Entregar avances en cada etapa completada
- Promover la cooperación entre equipos multidisciplinares y autogestionados
- Involucrar a los usuarios o clientes
- Mantener una comunicación fluida
- Realizar revisiones constantes de los avances del proyecto
- Adaptarse a los cambios que vayan surgiendo
- Las metodologías ágiles son muy comunes en el desarrollo de software porque permiten desarrollar productos y servicios de calidad que se adaptan a las necesidades de los clientes y a las preferencias de un mercado que cambia constantemente.

Las metodologías ágiles presentan diferentes variaciones. Las más conocidas se mencionan a continuación (Asana, 2024):

1. Scrum: en esta metodología se introducen perfiles como “Scrum master”, quien se encarga de permitir que el equipo ejecutando un “Sprint” pueda desarrollar tareas sin problemas.
2. Extreme Programming (XP): metodología enfocada en el desarrollo de software que prioriza la adaptabilidad y la calidad del producto mediante prácticas colaborativas y entregas frecuentes. Presenta ciclos de

desarrollo cortos, pruebas continuas, programación en pareja (Pair Programming), retroalimentación constante y refactorización continua.

3. Adaptive Project Framework (APF): diseñada para entornos complejos y dinámicos donde los requisitos y objetivos pueden cambiar con frecuencia. APF se centra en la adaptabilidad y la flexibilidad, permitiendo a los equipos ajustar los planes del proyecto de manera iterativa a medida que se recibe nueva información o se identifican necesidades del cliente.

3.2.4.2. DEFINICIÓN DE LA METODOLOGÍA ÁGIL KANBAN

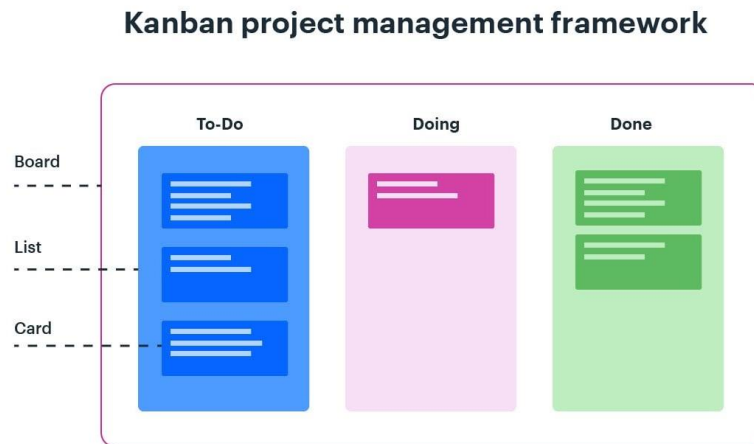
La estrategia Kanban conocida como “Tarjeta Visual” es muy útil para los responsables de proyectos. Esta consiste en la elaboración de un cuadro o diagrama en el que se reflejan tres columnas de tareas; pendientes, en proceso o terminadas como se describe en la Figura 3. Este cuadro debe estar al alcance de todos los miembros del equipo, evitando así la repetición de tareas o la posibilidad de que se olvide alguna de ellas. Por tanto, ayuda a mejorar la productividad y eficiencia del equipo de trabajo.

Las ventajas que proporciona esta metodología son:

- Planificación de tareas
- Mejora en el rendimiento de trabajo del equipo
- Métricas visuales
- Los plazos de entregas son continuos

Figura 3

Metodología Kanban



Nota. Tomado de kissflow.com

3.2.4.3. DEFINICIÓN DE LOS MODELOS GRANDES DE LENGUAJE NATURAL

Un modelo de lenguaje es un modelo de machine learning que puede generar lenguaje plausible (Humza Naveed, 2024). Un ejemplo de estos modelos es el autocompletado de los navegadores o los teléfonos móviles que utilizamos a diario. Estos modelos funcionan estimando la probabilidad de que un token o una secuencia de tokens aparezcan dentro de una secuencia más larga de tokens. Una secuencia de tokens podría ser una oración completa o una serie de oraciones. Es decir, un modelo de lenguaje podría calcular la probabilidad de diferentes oraciones completas o bloques de texto. Para medir la calidad de estos modelos, es importante determinar qué tan cerca está su distribución de probabilidad de la distribución empírica del corpus. Por esta razón se considera a los modelos grandes de lenguaje como modelos estadísticos de lenguaje (Michael Douglas, 2023).

El tamaño y la capacidad de los modelos de lenguaje han crecido exponencialmente en los últimos años a medida que aumentan la memoria de las computadoras, el tamaño de los conjuntos de datos y la potencia de procesamiento, y se desarrollan técnicas más efectivas para modelar secuencias de texto más largas. Se consideran modelos grandes lenguaje natural si contienen una cantidad de parámetros mayor a 100 millones (Google Developers, 2024).

3.2.4.3.1. Tipos de Modelos Grandes de Lenguaje Natural

Existen varios tipos de modelos de lenguaje grandes, cada uno con sus propias ventajas y desventajas.

1. Modelos de lenguaje autorregresivos: Estos modelos generan texto prediciendo la siguiente palabra en función de las palabras anteriores en la secuencia. Un ejemplo es GPT-3. Se entrenan para maximizar la probabilidad de predecir correctamente la próxima palabra con base en el contexto previo. Son efectivos en la generación de texto coherente y relevante, aunque pueden ser computacionalmente costosos y a veces producen respuestas repetitivas o irrelevantes. (Zhilin Yang, 2019)
2. Modelos basados en transformadores: Los transformadores son una arquitectura clave en muchos modelos de lenguaje grandes, introducidos por Vaswani et al. en 2017. Permiten procesar y generar texto de manera eficiente, capturando dependencias a largo plazo y el contexto en las secuencias de texto.
3. Modelos codificador-decodificador: Estos modelos, utilizados en tareas como traducción automática, resumen y respuesta a preguntas, tienen dos componentes: el codificador que procesa la secuencia de entrada y el decodificador que genera la secuencia de salida. El codificador convierte la entrada en una representación de longitud fija, que el decodificador usa para producir la salida.
4. Modelos preentrenados y ajustados: Los modelos preentrenados son entrenados en grandes conjuntos de datos, lo que les permite aprender patrones generales del lenguaje. Luego, se pueden ajustar con datos específicos de una tarea para adaptarse a necesidades concretas, como análisis de sentimientos o reconocimiento de entidades. Este método optimiza recursos y tiempo en comparación con entrenar un modelo desde cero.
5. Modelos multilingües: Estos modelos se entrenan con datos en múltiples idiomas, lo que les permite procesar y generar texto en varios lenguajes. Son útiles para tareas como la traducción automática, chatbots

multilingües o recuperación de información en diferentes idiomas, transfiriendo conocimientos entre lenguas.

3.2.4.3.2. Modelos de Código Abierto (Open Source) y Modelos Propietarios

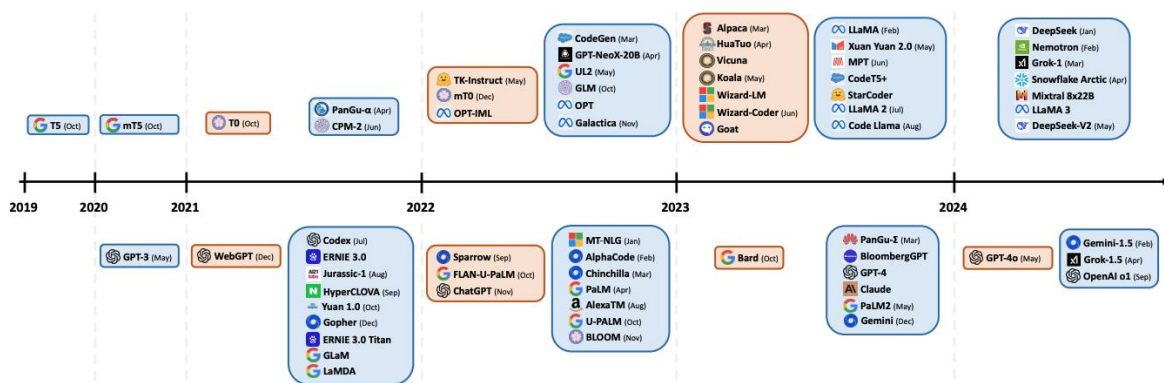
Los LLM de código abierto, como los desarrollados por Meta AI o Mistral, permiten el acceso público tanto a su código fuente como a su arquitectura, lo que facilita que desarrolladores, investigadores y empresas los utilicen, adapten y distribuyan sin restricciones.

Los modelos de código cerrado o propietarios, como GPT-3.5 de OpenAI, Claude de Anthropic o Gemini de Google, incorporan tecnologías de inteligencia artificial avanzadas capaces de analizar y generar texto similar al humano mediante el aprendizaje a partir de amplios conjuntos de datos. A diferencia de sus contrapartes de código abierto, el código fuente y la arquitectura de los modelos de lenguaje de código cerrado son exclusivos y solo se puede acceder a ellos bajo términos específicos definidos por sus creadores. Esta exclusividad permite un desarrollo, una distribución y un uso controlados. (Hailin Chen, 2024).

En la Figura 4 se muestran diferentes LLM creados desde 2019 hasta la actualidad.

Figura 4

Visualización cronológica de los lanzamientos de LLM



Nota. Tomado de “A Comprehensive Overview of Large Language Models” (Humza Naveed, 2024). Los modelos de la mitad superior son open source, mientras que los de la mitad inferior son de código cerrado o propietarios.

3.2.4.3.3. Arquitectura de un LLM

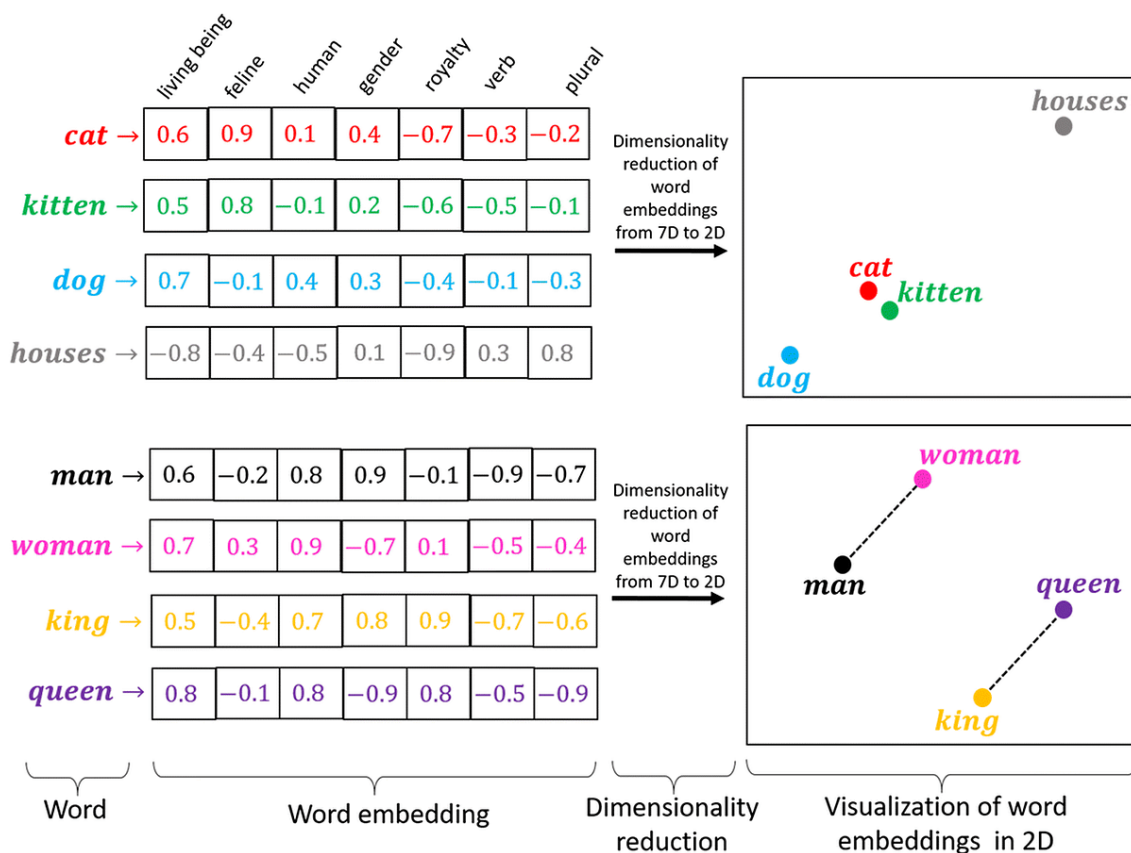
Como se ha mencionado anteriormente, existen diferentes tipos de modelos grandes de lenguaje. Cada tipo hace uso de una arquitectura diferente. La arquitectura más popular y conocida es la arquitectura de transformadores. Un modelo “Transformer” o transformador consta de diferentes componentes. Para explicar mejor dichos componentes podemos definir los pasos principales que se siguen para procesar los datos de entrada. El proceso comienza cuando el modelo convierte cada palabra en vectores numéricos multidimensionales, lo que permite representar su significado matemáticamente. A continuación, esta información vectorial atraviesa diversos niveles de procesamiento y transformación. Durante este proceso, un componente esencial es el sistema de autoatención, que permite analizar y establecer conexiones significativas entre los diferentes elementos de la secuencia de texto. Al finalizar el procesamiento en las capas transformadoras, el modelo utiliza el contexto que ha aprendido para determinar y generar la siguiente palabra o token que tiene mayor probabilidad de aparecer en la secuencia.

1. Incorporación de palabras (Word Embedding)

El desarrollo de un modelo de lenguaje grande comienza con la incorporación de palabras o el proceso conocido como word embedding. Este proceso consiste en representar las palabras como vectores en un espacio multidimensional, donde las palabras con significados similares se agrupan. Esto permite al modelo captar el significado de las palabras y realizar predicciones basadas en ese entendimiento.

Figura 5

Word embeddings, representación de palabras en vectores multidimensionales



Nota. Tomado de "Word Embedding: Basics. Medium. 2020. Hariom Gautam"

Consideremos como ejemplo las palabras "gato" y "perro". En el espacio vectorial, estas palabras tienden a encontrarse muy cerca, a diferencia de palabras más distantes semánticamente como "gato" y "casa". Esta cercanía refleja que ambos términos comparten características comunes - son animales domésticos que la gente suele tener como mascotas, tienen pelaje, se consideran amigables, etc. Al convertir estas palabras en vectores, el modelo puede capturar estas similitudes conceptuales y reconocer que pueden usarse en situaciones parecidas.

Para lograr estas representaciones vectoriales, es necesario entrenar una red neuronal utilizando grandes volúmenes de texto, como pueden ser artículos de noticias o libros digitalizados. Durante el proceso de entrenamiento, el modelo va aprendiendo a anticipar qué palabras tienen más probabilidad de aparecer en un contexto específico, basándose en las palabras que aparecen antes y después de dichas palabras dentro de cada oración. A través de este proceso,

los vectores resultantes logran capturar los vínculos de significado entre las distintas palabras del corpus. Este mismo principio se aplica a otros grupos de palabras relacionadas, como ocurre con "Rey", "Reina", "Hombre" y "Mujer", donde los vectores reflejan tanto las similitudes como las relaciones entre estos conceptos.

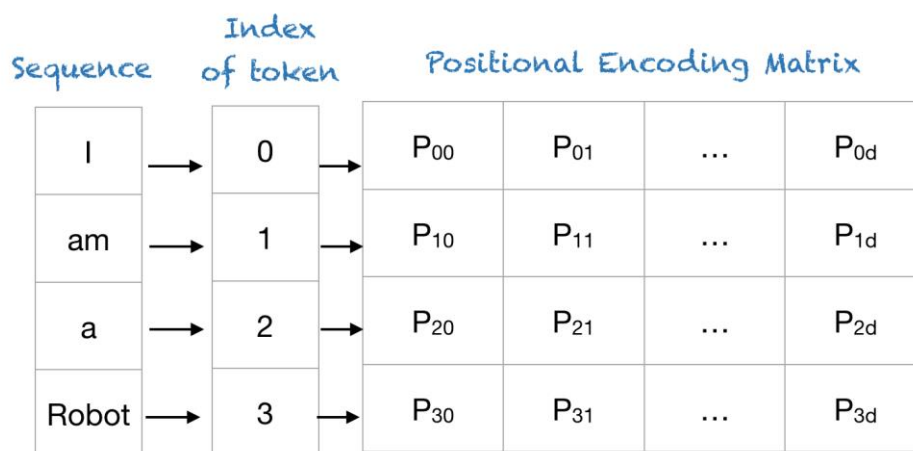
2. Codificación Posicional (Posicional Encoding)

La codificación posicional busca ayudar al modelo a identificar la posición de las palabras dentro de una secuencia (Amirhossein Kazemnejad, 2023). Su propósito no es capturar el significado de las palabras ni sus similitudes, como el hecho de que "gato" y "perro" sean términos que se encuentran cercanos en el espacio vectorial (son similares), sino en mantener el orden de las palabras. Por ejemplo, al traducir una oración como "El gato está en casa" a otro idioma, es esencial reconocer que "gato" precede a "casa". Este orden es muy importante para tareas de traducción, resumen y respuesta a preguntas (conocido como Q&A – Question and Answer).

La arquitectura de Transformers implementa la codificación posicional, donde cada elemento o índice en la secuencia recibe su propio vector representativo. Como resultado, cuando los datos salen de la capa de codificación posicional, forman una matriz donde cada fila combina dos elementos: la representación del objeto de la secuencia más la información sobre su ubicación específica. Se puede visualizar esto en la Figura 6 donde se muestra un ejemplo de la matriz que codifica únicamente la información posicional.

Figura 6

Codificación posicional



Positional Encoding Matrix for the sequence 'I am a robot'

Nota. Tomado de "A Gentle Introduction to Positional Encoding in Transformer Models, Part 1. Machine Learning Mastery. 2023. Mehreen Saeed"

Durante el entrenamiento, el modelo se expone a un extenso corpus de texto y aprende a realizar predicciones basadas en esos datos. A través de un proceso iterativo de ajuste de los pesos neuronales, utilizando un algoritmo de retropropagación o backpropagation, el modelo minimiza la discrepancia entre sus predicciones y las salidas reales.

3. Transformadores (Transformers)

Es común encontrar la capa "Transformer" como un elemento adicional en las arquitecturas de redes neuronales tradicionales, debido a que mejora significativamente la capacidad del modelo de lenguaje para detectar y procesar conexiones distantes dentro de textos. A diferencia del procesamiento secuencial como se presentaba en las capas de redes neuronales recurrentes, esta capa tiene la particularidad de analizar toda la secuencia de entrada de manera simultánea. Su funcionamiento se basa en dos elementos fundamentales: un mecanismo que permite la autoatención (self-attention mechanism) y una red neuronal que proporciona retroalimentación (feedforward neural network).

Consideremos que cada palabra en una oración tiene un significado básico, como una definición de diccionario. La autoatención es como añadir

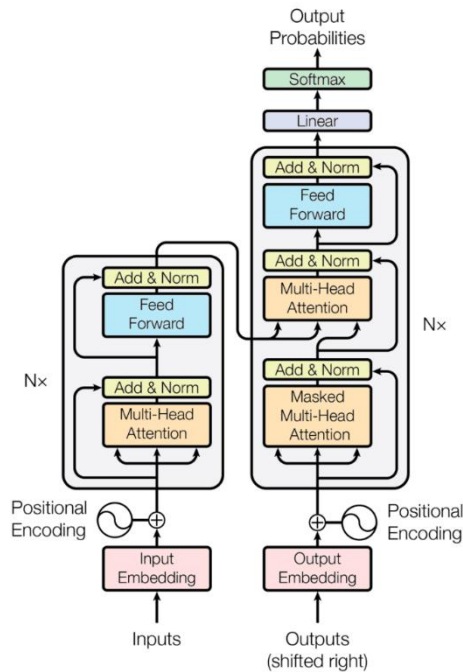
capas de significado a esa palabra, considerando todas las demás palabras a su alrededor dentro de una oración. En vez de tratar cada palabra por separado, este mecanismo permite al modelo entender las relaciones entre ellas y cómo se influyen mutuamente. Es por ello que la misma palabra puede tener un peso diferente en la interpretación final dependiendo del contexto. Si tomamos como ejemplo la palabra "banco", esta puede referirse a un lugar para sentarse o a una institución financiera. La autoatención ayuda al modelo a decidir cuál de los dos significados es el correcto al analizar las palabras que la rodean en la frase.

Una vez que la capa de autoatención ha analizado la secuencia, la capa de retroalimentación de posición se encarga de procesar cada elemento de la secuencia por separado. Para ello, utiliza una red neuronal que transforma la representación vectorial de cada elemento (obtenida de la capa de autoatención) en una nueva representación. Esta transformación ayuda al modelo a identificar patrones complejos y conexiones entre palabras.

El proceso de aprendizaje implica ajustar continuamente los parámetros de esta red neuronal para que las predicciones del modelo se acerquen cada vez más al resultado deseado. Este ajuste se realiza mediante un algoritmo de retropropagación, similar al utilizado en el entrenamiento de redes neuronales convencionales.

Figura 7

Arquitectura de un Transformer



Nota. Tomado de Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30.

4. Generación de Texto

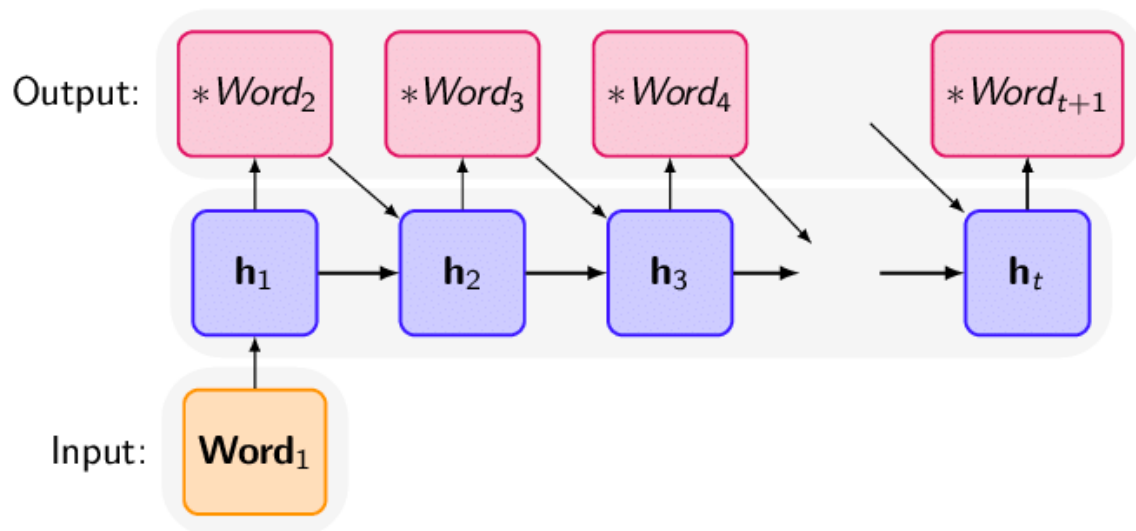
Este es el paso final que realiza un modelo grande lenguaje. Una vez entrenado y ajustado, el modelo puede emplearse para producir texto de alta calidad en respuesta a una instrucción o pregunta. El proceso comienza con una entrada de texto inicial, que puede variar desde unas pocas palabras hasta una oración completa o incluso un párrafo. A partir de esta entrada, el modelo utiliza los patrones adquiridos durante su entrenamiento para generar una respuesta que sea coherente y relevante en el contexto dado.

El texto de salida se genera mediante una técnica conocida como autorregresión. En este enfoque, el modelo produce cada palabra o token de manera secuencial, considerando las palabras previas que ha generado. Para determinar la siguiente palabra o token, el modelo utiliza a los parámetros aprendidos durante su entrenamiento, calcula la distribución de probabilidad

correspondiente y selecciona la opción más probable como el siguiente elemento en la secuencia.

Figura 8

Generación de texto usando autorregresión



Nota. Kelleher, J., Dobnik, S. (2018). What is not where: The challenge of integrating spatial representations into deep learning architectures. ResearchGate.

3.2.4.3.4. Uso de LLMs

Los modelos grandes de lenguaje tienen una variedad de usos, pero todos ellos tienen en común la generación de nuevo contenido basado en parámetros que son configurables por los usuarios. Los LLMs se pueden utilizar para realizar muchas tareas que normalmente llevarían mucho tiempo a los humanos, como generación de texto, traducción, resumen de contenido, reescritura, clasificación y análisis de sentimientos. Los modelos de lenguaje de gran tamaño también pueden impulsar chatbots, que permiten a los clientes hacer preguntas y buscar ayuda o recursos sin ingresar a una cola de soporte. (Petter Törnberg, 2023)

3.2.4.3.5. Ingeniería de Instrucciones (Prompt Engineering)

El diseño de instrucciones consiste en elaborar indicaciones que generen

la respuesta esperada por parte de los modelos de lenguaje. Crear instrucciones bien organizadas puede ser clave para asegurar respuestas precisas y de alta calidad en estos modelos. El proceso continuo de ajustar las instrucciones y revisar repetidamente las respuestas del modelo se conoce a menudo como ingeniería de instrucciones. (Alec Radford, 2019)

Los componentes de una instrucción (prompt) incluyen por lo general en uno de los siguientes componentes (Jason Wei, 2022):

- Tarea (obligatoria)
- Instrucciones del sistema (opcional)
- Ejemplos (opcional)
- Información contextual (opcional)

La Figura 9 presenta los componentes esenciales y opcionales de una instrucción.

Figura 9

Componentes de una instrucción

Componente	Descripción	Ejemplo
Objetivo	Qué quieres que logre el modelo. Sé específico y, además, incluye todos los objetivos generales. También se denomina "objetivo" o "misión".	Tu objetivo es ayudar a los estudiantes con problemas matemáticos sin darles la respuesta directamente.
Instrucciones	Instrucciones paso a paso sobre cómo realizar la tarea en cuestión. También se denominan "tarea", "pasos" o "indicaciones".	1. Comprende qué plantea el problema. 2. Comprende dónde está detenido el alumno. 3. Proporciona una pista para el siguiente paso del problema.
Componentes opcionales		
Persona ficticia	Como quién o qué actúa el modelo. También se conoce como "rol" o "visión".	Eres un tutor de matemáticas que ayuda a los alumnos con sus tareas de matemáticas.
Restricciones	Restricciones sobre lo que el modelo debe cumplir cuando se genera una respuesta, incluido lo que el modelo puede y no puede hacer. También se denominan "barreras", "límites" o "controles".	No le des la respuesta al alumno directamente. En cambio, brinda pistas en el siguiente paso para resolver el problema. Si el estudiante está completamente perdido, proporciónale los pasos detallados para resolver el problema.
Tono	El tono de la respuesta. También puedes influir en el estilo y el tono si especificas una persona. También se denominan "estilo", "voz" o "modo".	Responde de manera informal y técnica.
Contexto	Cualquier información que el modelo necesite consultar para realizar la tarea en cuestión. También se denominan "segundo plano", "documentos" o "datos de entrada".	Una copia de los planes de las lecciones del alumno para matemáticas.
Ejemplos con muestras	Ejemplos de cómo debería ser la respuesta para una instrucción determinada. También se denominan "ejemplares" o "muestras".	input : Estoy intentando calcular cuántas pelotas de golf pueden caber en una caja que tiene un volumen de un metro cúbico. Convertí un metro cúbico en centímetros cúbicos y lo dividí por el volumen de una pelota de golf en centímetros cúbicos, pero el sistema indica que mi respuesta es incorrecta. output : Las bolas de golf son esferas y no se pueden empaquetar en un espacio con una eficiencia perfecta. Tus cálculos tienen en cuenta la eficiencia máxima del empaquetado de las esferas.
Pasos de razonamiento	Indícale al modelo que explique su razonamiento. A veces, esto puede mejorar la capacidad de razonamiento del modelo. También se conoce como "pasos del pensamiento".	Explica tu razonamiento paso a paso.
Formato de la respuesta	El formato en el que deseas que tenga la respuesta. Por ejemplo, puedes indicarle al modelo que genere la respuesta en JSON, tabla, Markdown, párrafo, lista con viñetas, palabras clave, presentación breve y demás. También se denomina "estructura", "presentación" o "diseño".	Da formato a tu respuesta en Markdown.

Nota. Tomado de la documentación de Google – Vertex AI.

3.2.4.3.6. Control de los Resultados Generados por LLMs

Es fundamental controlar la salida de los modelos de lenguaje de gran tamaño (LLMs) para cumplir con requisitos específicos o ajustarse a un estilo deseado. Hay tres parámetros clave que influyen en la salida de estos modelos:

top-k, top-p y temperatura.

Antes de profundizar en estos parámetros, es importante entender la diferencia entre el muestreo voraz (greedy sampling) y el muestreo aleatorio (random sampling). El muestreo voraz selecciona siempre el token con la mayor probabilidad, lo que produce una salida más precisa y directa. En cambio, el muestreo aleatorio (utilizando top-k o top-p) introduce variabilidad, generando una salida más diversa y creativa (Tom B. Brown, 2020). La mayoría de los LLMs actuales (como GPT, Llama-2, Claude, Gemini, etc.) utilizan muestreo voraz, por lo que se requieren los parámetros top-p y top-k para gestionar esa aleatoriedad.

El parámetro top-k limita las predicciones del modelo a los k tokens más probables en cada paso de la generación. Al establecer un valor para k, se le indica al modelo que considere solo los k tokens más probables. Esto puede ayudar a ajustar el resultado generado y garantizar que se adhiera a patrones o restricciones específicos.

Top-p, o muestreo de núcleo, regula la probabilidad acumulada de los tokens generados. El modelo continúa generando tokens hasta que la probabilidad acumulada excede el umbral establecido (p). Este método ofrece un control más flexible sobre la longitud del texto generado y promueve la diversidad al permitir la inclusión de tokens menos probables cuando es necesario.

El parámetro de temperatura se utiliza para ajustar la aleatoriedad en la salida de un modelo de lenguaje, pero lo hace modificando la forma en que se calcula la distribución de probabilidad de los posibles tokens siguientes, en lugar de restringir directamente la selección de tokens. La temperatura actúa como un factor de escala aplicado en la capa softmax final del modelo, lo que altera la distribución de probabilidad para el siguiente token.

Cuando la temperatura es alta (alrededor de 1), el modelo introduce más variabilidad y explora una mayor variedad de tokens posibles, lo que genera respuestas más creativas o impredecibles. En cambio, con una temperatura baja (menos de 1), el modelo favorece los tokens más probables, lo que produce un texto más preciso y enfocado.

Por ejemplo, si un modelo está generando la siguiente palabra en la frase "El cielo está..." y la temperatura es alta (como 1), el modelo podría elegir entre varias opciones menos predecibles, como "brillante", "nublado" o incluso

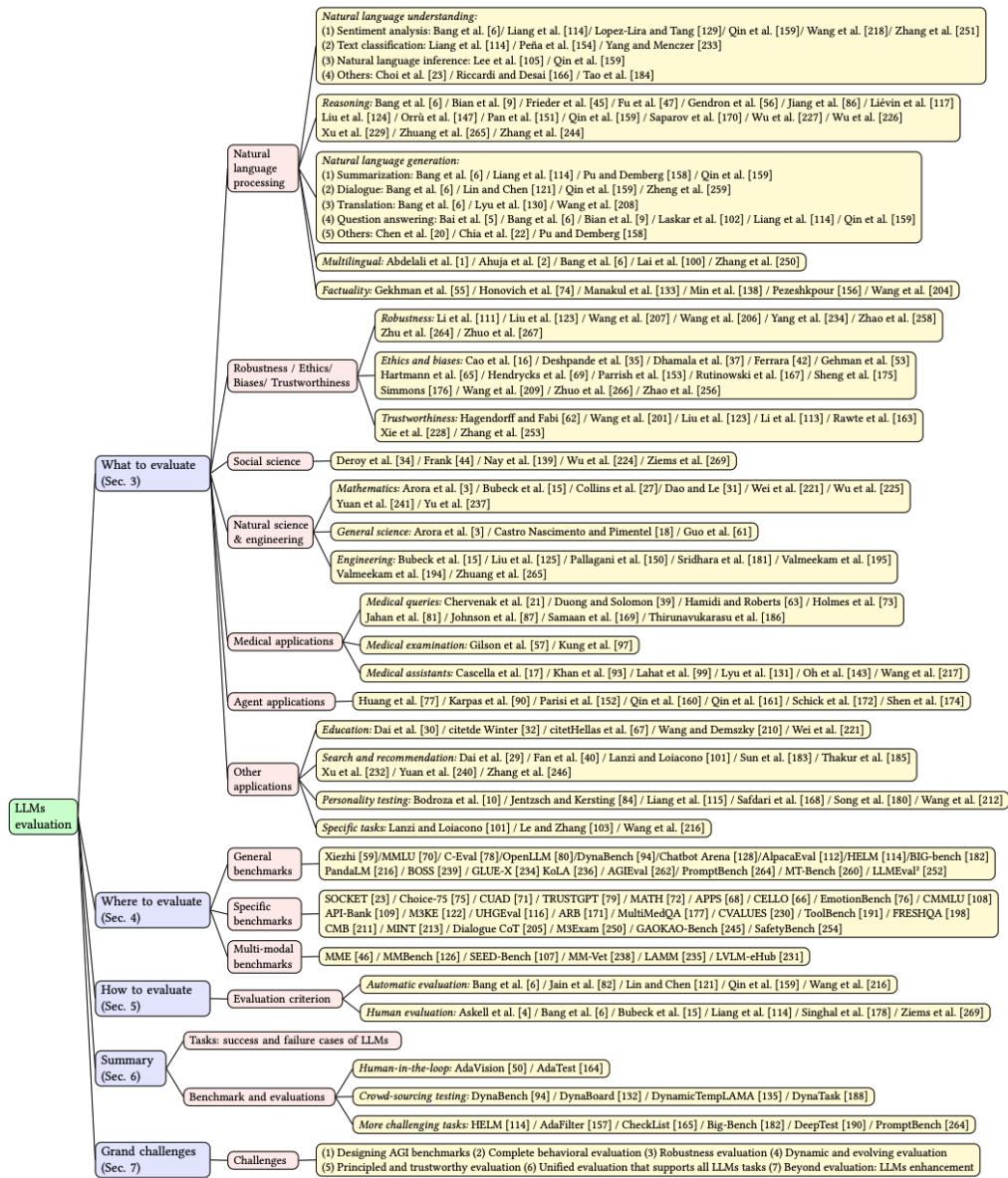
"enrojecido". Con una temperatura baja (como 0.2), el modelo probablemente seleccionará la opción más común o esperada, como "azul".

3.2.4.3.7. Evaluación de LLMs

La evaluación de los modelos de lenguaje natural es crucial para determinar tanto su capacidad como sus limitaciones. Este proceso examina la habilidad del modelo para entender, generar e interactuar con el lenguaje humano en diversas tareas. Existen diferentes formas y áreas para evaluar un modelo (Yupeng Chang, 2023) como se muestra en la Figura 10, pero para este estudio se ha simplificado la evaluación de un modelo de lenguaje (LM) clasificándolo en dos grandes áreas: 1) comprensión del lenguaje natural (NLU) y 2) generación del lenguaje natural (NLG).

Figura 10

Evaluación de LLMs



Nota. Tomado de “A Survey on Evaluation of Large Language Models” – 2023

1. Comprensión del lenguaje natural (NLU): evalúa la capacidad de los modelos de lenguaje para entender el lenguaje, abarcando diversas tareas como el análisis de sentimientos, clasificación de textos, inferencia de lenguaje natural (NLI), respuesta a preguntas (QA), razonamiento de sentido común (CR), razonamiento matemático (MR) y comprensión lectora (RC), entre otras.
2. Generación del lenguaje natural (NLG): mide la habilidad de los modelos

de lenguaje para generar texto basándose en el contexto proporcionado. Incluye tareas como la síntesis de resúmenes, finalización de oraciones, traducción automática (MT) y generación de diálogos, entre otras.

3.2.4.3.8. Fundamentación (Grounding)

En la IA generativa, fundamentar es la capacidad de conectar el resultado del modelo con fuentes de información verificables. Si se proporciona a un modelo acceso a fuentes de datos específicas, el proceso de fundamentación conecta el resultado a estos datos y reduce las posibilidades de inventar contenido. Esto es muy importante en situaciones en las que la precisión y la confiabilidad son significativas.

Usando los modelos de Google y la técnica de fundamentación con búsquedas en el buscador de Google, se puede vincular al modelo con Internet. Gracias a que Internet proporciona acceso a información existente y nueva, un prompt fundamentado puede proporcionar respuestas matizadas y específicas, con pocos errores y con fuentes citadas.

3.2.4.3.9. Almacenamiento en caché (Caching)

Los LLM enfrentan un reto computacional debido a que su arquitectura de transformador de solo decodificador (decoder-only) presenta una complejidad temporal cuadrática al procesar secuencias de texto. Este problema se agrava durante la inferencia por el mecanismo de decodificación autorregresiva, que repite el proceso para cada token generado. El uso del KV Cache (key-value cache), que almacena las claves y los valores del tensor de atención generados por tokens anteriores, permite reducir la complejidad temporal a lineal, lo que mejora notablemente la eficiencia de la inferencia. No obstante, el KV Cache tiene sus limitaciones, ya que su tamaño crece linealmente con la longitud de la secuencia, aumentando el uso de memoria, especialmente en modelos grandes como GPT-3 (Floridi & Chiriatti, 2020). Además, cada conversación debe mantener su propio KV Cache, lo que dificulta su reutilización entre diálogos, creando un obstáculo para la velocidad de generación en hardware moderno de inferencia, como las GPU, que suelen tener un ancho de banda de memoria limitado en comparación con su capacidad de cómputo (Yu et al., 2022). Existen diferentes métodos para optimizar KV Cache. Estos métodos abarcan la fase de

preentrenamiento, la fase de implementación y la fase de inferencia. (Shi Luohe et al., 2024).

Google ha implementado algunos de estos métodos para optimizar la latencia de sus modelos, especialmente de Gemini. Con la función de almacenamiento en caché de contexto de la API de Gemini, se puede pasar algo de contenido al modelo una vez, almacenar en caché los tokens de entrada y luego hacer referencia a los tokens almacenados en caché para solicitudes posteriores. En ciertos volúmenes, el uso de tokens almacenados en caché tiene un costo menor que pasar el mismo corpus de tokens repetidamente.

Con el almacenamiento en caché, se puede elegir cuánto tiempo es necesario que exista el caché antes de que los tokens se eliminen automáticamente. Esta duración del almacenamiento en caché se denomina tiempo de vida (TTL) y el valor predeterminado es 1 hora. El costo del almacenamiento en caché depende del tamaño del token de entrada y de cuánto tiempo desea que persistan los tokens.

3.2.4.4. DEFINICIÓN DE LOS MODELOS DE EMBEDDINGS DE TEXTO

Los embeddings de texto son una técnica de procesamiento del lenguaje natural (NLP) que transforma el texto en vectores numéricos que pueden ser representados en un espacio n-dimensional. Esta técnica permite tratar fragmentos de texto como datos relacionales, los cuales pueden utilizarse para entrenar modelos. (L. Wang, 2022)

Los embeddings capturan tanto el significado semántico como el contexto del texto, de manera que los fragmentos con significados similares tienen representaciones más cercanas entre sí en el espacio vectorial. Por ejemplo, las frases "Llevé a mi perro al veterinario" y "Llevé a mi gato al veterinario" tendrían embeddings cercanos, ya que comparten un contexto similar.

Esta técnica permite comparar diferentes textos y entender su relación. Por ejemplo, si los embeddings de las palabras "gato" y "perro" están próximos, se puede deducir que tienen similitudes en su significado o contexto. Esta capacidad abre la puerta a una variedad de aplicaciones, como se detalla en la siguiente sección. (Google AI for Developers, 2024)

3.2.4.5. DEFINICIÓN DE LAS BASES DE DATOS VECTORIALES

Una base de datos vectorial es un tipo de base de datos que almacena datos en forma de vectores de alta dimensión, los cuales son representaciones matemáticas de características o atributos. Cada vector tiene un número determinado de dimensiones, que puede variar desde decenas hasta miles, dependiendo de la complejidad y detalle de los datos. (Yikun Han, 2023)

Estos vectores se generan aplicando alguna función de transformación o embedding a los datos originales, como texto, imágenes, audio o video. Estas funciones de “embedding” pueden basarse en diversos métodos, como modelos de machine learning, embeddings de palabras (word embeddings) o algoritmos de extracción de características.

Las bases de datos vectoriales presentan varias ventajas en comparación con las bases de datos tradicionales. Una de ellas es la capacidad para realizar búsquedas y recuperaciones rápidas y precisas basadas en similitudes vectoriales, lo que es crucial en aplicaciones como el procesamiento del lenguaje natural, la visión por computadora y los sistemas de recomendación. A diferencia de las bases de datos tradicionales, que solo pueden buscar coincidencias exactas o usar criterios predefinidos, las bases de datos vectoriales permiten capturar el significado semántico o contextual de los datos. (Schwaber-Cohen)

3.2.4.6. DEFINICIÓN DE LOS SERVICIOS DE GOOGLE CLOUD PLATFORM (GCP)

3.2.4.6.1 BigQuery

BigQuery es servicio de almacén de datos (data warehouse) completamente administrado (es decir que no es necesario para el usuario administrar la infraestructura) y sin servidor desarrollado por Google. A continuación, se describen algunas de sus principales características (BigQuery, Google Cloud):

1. **Análisis de datos a gran escala:** Permite analizar grandes cantidades de datos mediante consultas SQL, pudiendo procesar terabytes o petabytes de información en segundos.

2. Integración nativa con otras herramientas de Google Cloud.
3. Capacidad de analizar datos en tiempo real.
4. Soporte para datos estructurados y semiestructurados.
5. Sistema de precios basado en el uso (pagas por los datos procesados).
6. Casos de uso comunes:
 - Análisis de datos empresariales (Inteligencia de negocios)
 - Machine Learning
 - Procesamiento de logs
 - Análisis de datos de IoT

3.2.4.6.2 Cloud Run

Cloud Run es un servicio de Google Cloud que permite ejecutar aplicaciones en contenedores de forma completamente administrada y sin servidor. Es especialmente útil para desplegar aplicaciones basadas en contenedores que se ejecutan en un entorno altamente escalable, respondiendo dinámicamente a las necesidades de tráfico. A continuación, se listan algunas características de Cloud Run (Cloud Run, Google Cloud):

1. Basado en contenedores:
 - a. Permite ejecutar aplicaciones empaquetadas en contenedores Docker.
 - b. Es compatible con cualquier lenguaje de programación o framework que pueda ejecutarse en un contenedor.
2. Completamente administrado:
 - a. No es necesario configurar ni mantener servidores.
 - b. Google Cloud se encarga de la escalabilidad, seguridad y administración del entorno.
3. Escalabilidad automática:
 - a. Escala automáticamente según la demanda, desde cero (sin costos cuando no hay tráfico) hasta miles de solicitudes por segundo.
 - b. Es un servicio ideal para aplicaciones con tráfico variable.
4. Pago por uso:
 - a. Los usuarios solo pagan por los recursos que sus aplicaciones

- utilizan mientras se procesan solicitudes.
 - b. Esto incluye CPU, memoria y duración de ejecución de las solicitudes.
5. Soporte para HTTP/HTTPS y eventos:
- a. Diseñado para responder a solicitudes HTTP/HTTPS.
 - b. También puede integrarse con otros servicios de Google Cloud para manejar eventos (usando Pub/Sub, por ejemplo).
6. Flexibilidad en el despliegue:
- a. Puede ejecutarse de manera completamente administrada en Google Cloud o en Kubernetes a través de Cloud Run for Anthos.
7. Compatibilidad con estándares abiertos:
- a. Basado en Knative, una tecnología de código abierto que permite portabilidad entre diferentes proveedores de nube y entornos locales.

3.2.4.6.3 Vertex AI

Vertex AI es una plataforma de Google Cloud diseñada para simplificar todo el ciclo de vida de machine learning e inteligencia artificial. Esta herramienta permite a equipos técnicos y no técnicos trabajar juntos en el desarrollo e implementación de soluciones de IA, aprovechando la robusta infraestructura de Google Cloud.

La plataforma destaca por su versatilidad en el entrenamiento de modelos, ofreciendo opciones como AutoML para usuarios sin experiencia en programación y herramientas avanzadas para expertos que requieren mayor control. Su “Model Garden” facilita el acceso a diversos modelos preentrenados, incluyendo acceso a IA generativa para texto, imágenes y más.

En cuanto a operaciones de ML (MLOps), Vertex AI automatiza flujos de trabajo completos mediante el uso de pipelines y proporciona un “Feature Store” o almacén de features, que permite gestionar eficientemente los features de los modelos. VertexAI se integra fácilmente con otros servicios de Google Cloud como BigQuery y es compatible con frameworks populares como TensorFlow y PyTorch.

Los beneficios principales incluyen la simplificación del desarrollo de IA, capacidad de escalar según las necesidades del proyecto, y herramientas robustas para monitorear y mantener modelos en producción. Las aplicaciones prácticas son diversas, desde predicción de demanda y análisis textual hasta sistemas de recomendación personalizados y procesamiento de imágenes. (Vertex AI, Google Cloud)

3.2.5 IMPLEMENTACIÓN DE LAS ÁREAS, PROCESOS, SISTEMAS Y BUENAS PRÁCTICAS

3.2.5.1. ÁREAS

El área de Inteligencia Artificial de la empresa utiliza Cloud para proporcionar valoraciones de automóviles precisas e instantáneas a los potenciales usuarios vendedores y mostrar a los concesionarios de automóviles registrados qué vehículos en la plataforma son los más relevantes para sus negocios. A nivel interno, esta área utiliza IA para agilizar los procesos de validación de documentos complejos que requieren mucho tiempo, lo que permite a los agentes dedicar más tiempo a las valiosas interacciones con los clientes.

Actualmente, la empresa crea sus propios modelos de machine learning en la plataforma Vertex AI (Google) para perfeccionar las estimaciones de precios, considerando todos los detalles que pueden influir en la valoración de un vehículo. Esto incluye el análisis de los datos proporcionados por el vendedor (como la marca, el kilometraje, el historial de mantenimiento y los defectos del vehículo), además de estudios de mercado que evalúan la oferta y la demanda de cada automóvil en el momento de su valoración. Asimismo, la compañía utiliza endpoints de Vertex AI para integrar datos en tiempo real de las subastas realizadas el día anterior, lo que les permite entender el nivel de interés actual de los concesionarios en su plataforma y determinar el precio más probable que el vehículo pudiese alcanzar en una subasta.

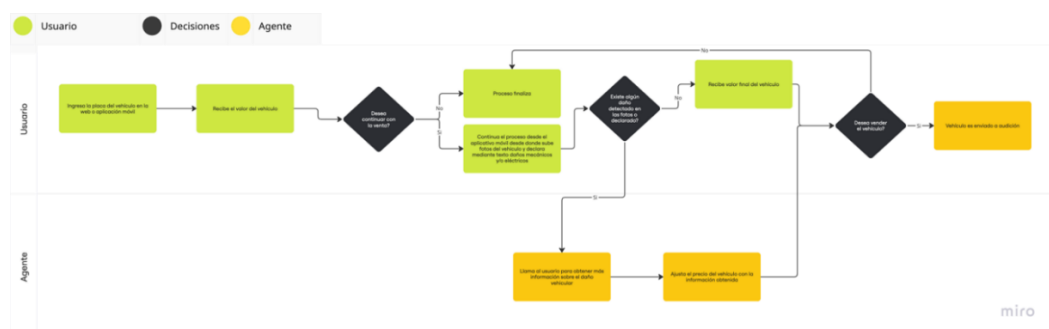
3.2.5.2. PROCESOS

En la Figura 11 se muestra el flujo de procesos de evaluación, aceptación y venta de vehículos. A continuación, se detallan los procesos representados en el gráfico:

1. **Enquiry (Consulta):** Es el primer paso del proceso, donde el 100% de los clientes inicia con una consulta.
2. **Valuation returned (Valoración devuelta):** Las consultas realizadas por usuarios que desean vender sus vehículos reciben una valoración del vehículo.
3. **Valuation accepted (Valoración aceptada):** Los usuarios aceptan la valoración ofrecida.
4. **Form complete (Formulario completado):** Usuarios que avanzan a completar el formulario.
5. **Photos complete (Fotos completadas):** Usuarios completan la carga de fotos del vehículo.
6. **Agent profiling (Perfilado por agente):** Se evalúan los datos proporcionados. Es aquí donde el proyecto busca generar un impacto automatizando el proceso y reduciendo los tiempos de espera por parte de los usuarios.
7. **QA (Control de calidad):** Se evalúa la calidad del vehículo y la veracidad de la información recibida.
8. **For sale (Para la venta):** En esta etapa se encuentran los vehículos que están listos para la venta.

Figura 11

Flujograma que muestra como interactúan los actores principales en el proceso de venta de vehículos



3.2.5.3. SISTEMAS Y BUENAS PRÁCTICAS

ETAPAS DEL PROYECTO

Para el diseño y desarrollo de la solución se siguieron las etapas del Modelo de Contratación establecidos por Google los cuales se visualizan en la Figura 12. El cronograma de las etapas que se usaron en el desarrollo se detalla en el Anexo 1. El proyecto inicia formalmente con un “Expert Request” (ER) creado en la plataforma Salesforce. Un ER cuenta con el detalle del cliente, la solicitud y el propietario o arquitecto encargado de la solución como se muestra en el Anexo 2.

Figura 12

Etapas del Modelo de Contratación



Nota. Basado en el documento interno “Field Solutions Architecture EMEA AI CE Practice 2024”

En cada entregable a desarrollar se emplearon las etapas detalladas en la Tabla 7. La metodología Kanban se utilizó en la etapa de desarrollo.

Tabla 7

Etapas del Proyecto

Etapas del Proyecto	
Etapas	Descripción
Calificar la oportunidad	Se evaluó el tamaño de la oportunidad e impacto en la organización y el cliente.
Descubrimiento	Se capturaron los retos y problemática que el cliente afrontaba.
Preparación	Se crearon acuerdos de confidencialidad y documentos

necesarios para dar inicio al proyecto.

Diseño	Se diseñó la solución técnica utilizando los documentos e información recabada en la etapa de descubrimiento.
Desarrollo	Se desarrolló la solución utilizando la metodología ágil Kanban.
Entrega de proyecto	Se programó la entrega del proyecto con los pasos a seguir necesarios para la puesta en producción.

El proyecto tiene 3 entregables mostrados en la Tabla 8 además del tiempo que tomó el desarrollo de cada uno. En cada etapa de los 3 entregables del proyecto se añadieron tareas en la columna “To do” como se visualiza en la Figura 13. Cuando las tareas se encontraban siendo ejecutadas por algún miembro se movían a la columna “In progress”. Finalmente, al completarse, se movieron a la columna “Done”.

Tabla 8

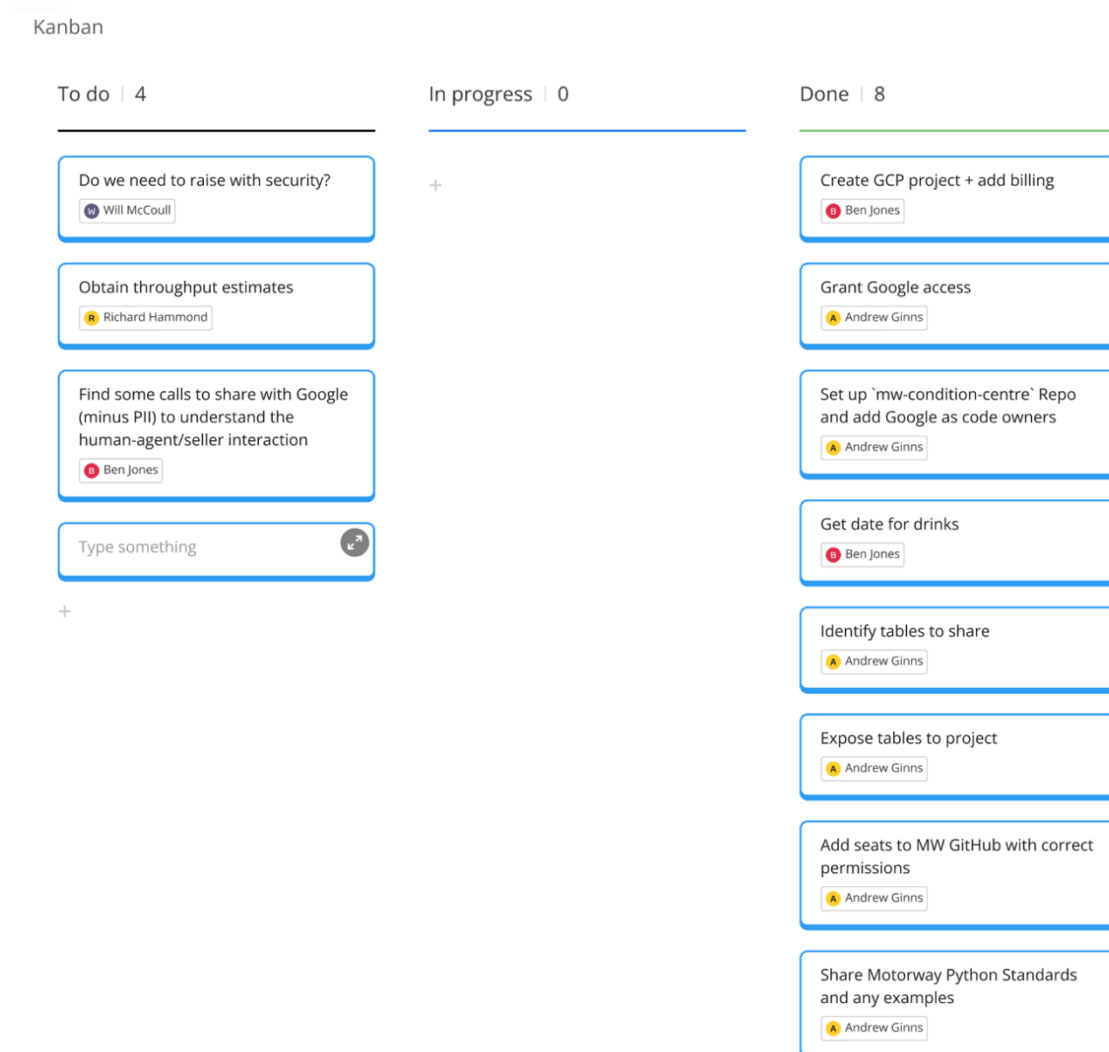
Listado de entregables del proyecto

Entregable	Descripción	Período de Tiempo
Prepltem Match	Mapear el texto ingresado por el usuario en lenguaje natural con una categoría de daño vehicular especificada en la base de datos de la empresa.	3 semanas
Prepltem Cost	Si existe un match, retornar el costo de arreglar dicho problema en el vehículo. Si no existe, utilizar un modelo grande de lenguaje natural para encontrar el precio usando técnicas como “grounding”.	2 semanas
Cost sensibility	Ajustar al modelo para que sea capaz de identificar las características particulares de cada vehículo	3 semanas

antes de asignar un precio.

Figura 13

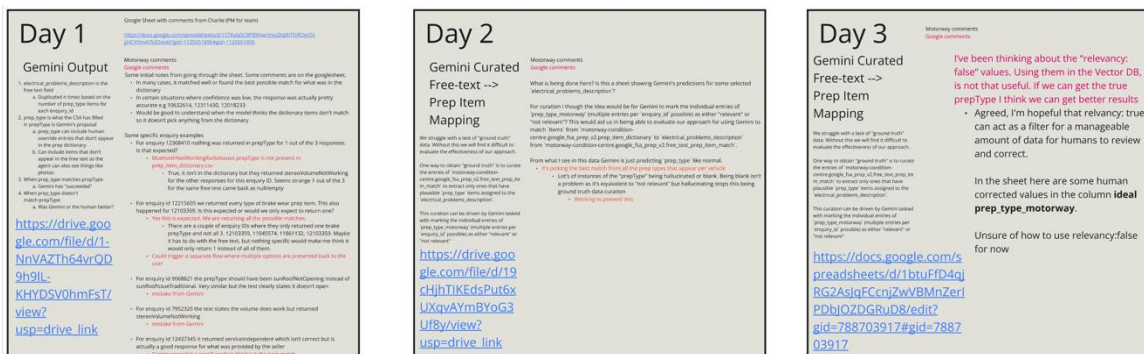
Tablero Kanban en MIRO



Nota. Tomado de la herramienta MIRO durante el desarrollo del proyecto.

De forma diaria y cuando era necesario, se crearon tarjetas adicionales en la herramienta MIRO para capturar observaciones relevantes y objeto de discusión entre los miembros del proyecto. Un ejemplo de estas tarjetas se puede observar en la Figura 14.

Notas diarias durante el desarrollo del proyecto



Nota. Tomado de la herramienta MIRO durante el desarrollo del proyecto.

La primera tarea luego del análisis y definición el alcance del proyecto fue la de diseñar la arquitectura conceptual (donde se describen los pasos que se llevan a cabo en el sistema) y la arquitectura tecnológica como se muestra en la Figura 15 y la Figura 16. La arquitectura conceptual permite a los desarrolladores obtener una visión clara del sistema que se desea construir, mientras que la arquitectura tecnológica muestra los componentes, herramientas y frameworks a utilizar.

Aprovechando el poder de los modelos grandes de lenguaje natural, el objetivo del primer entregable es clasificar la entrada de texto enviada por el usuario durante el proceso de registro. El resultado de esta clasificación fue definido como "prepType" indicado en el paso 1 de la Figura 15 (Obtener prepTypes). Después de obtener un prepType, extraeremos el precio de cada prepType accediendo a una base de datos que se utilizará en el último paso. El segundo paso (2. Recuperar el precio del vehículo) de la solución implicó proporcionar el precio de los artículos. Para este paso, nos basamos en los datos proporcionados por el equipo de la empresa, así como en los datos de la búsqueda de Google. En el paso final, combinamos el resultado del paso 1 y el paso 2 para devolver el precio final de los artículos.

Figura 15

Arquitectura conceptual de la solución

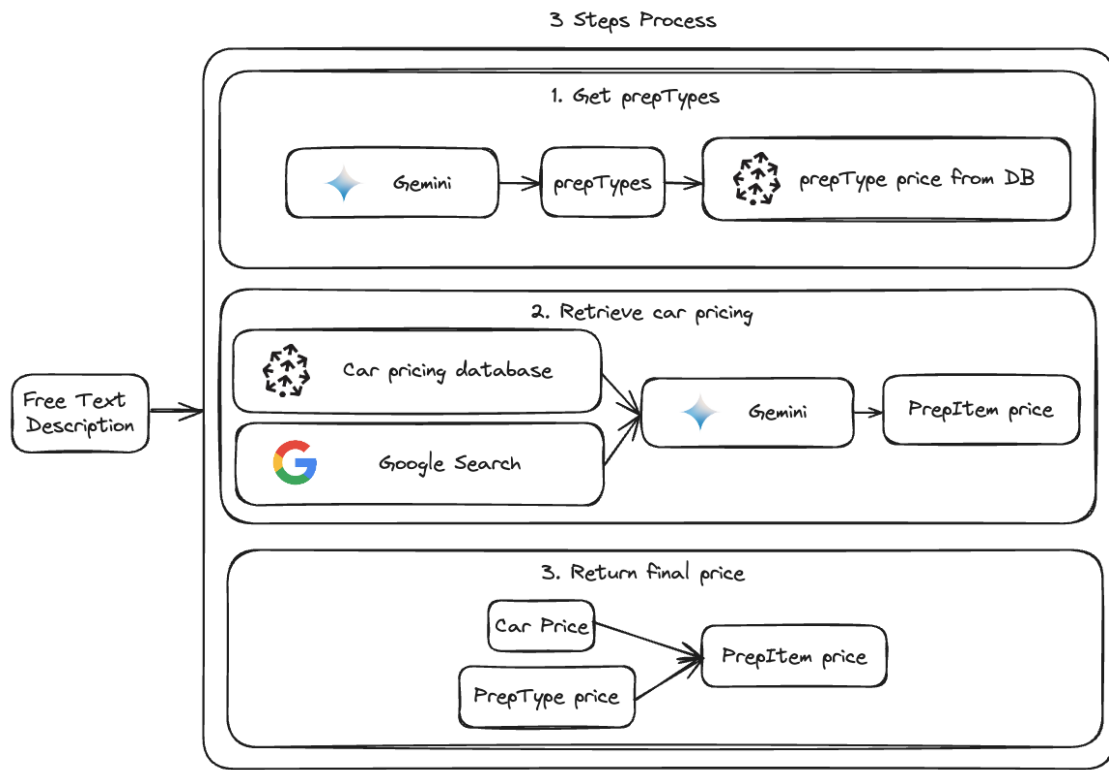
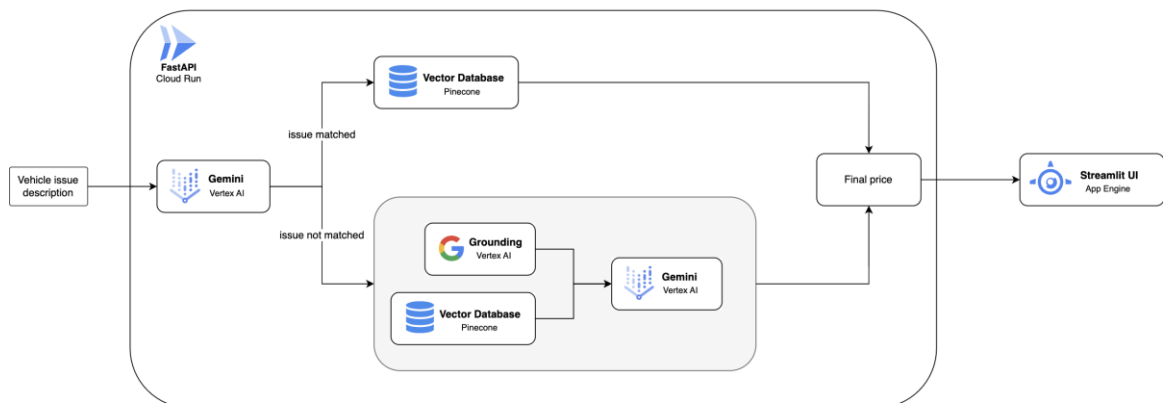


Figura 16

Arquitectura tecnológica de la solución



Con las arquitecturas conceptual y tecnológica diseñadas se procedió al desarrollo de los entregables. El tiempo total de desarrollo del proyecto es de 2 meses.

3.2.5.3.1. ENTREGABLE 1

El entregable de la etapa 1 implica:

- Preparar los datos. Corregir el campo de texto libre en busca de errores tipográficos, errores, etc.
- Separar cada uno de los problemas mencionados en un formato estructurado.
- Intentar asignar cada elemento a nuestra taxonomía predefinida.
- Desambiguar cuando sea necesario (cuando haya múltiples coincidencias creíbles)
- Identificar si no existe una buena coincidencia: asegurarse de registrar esto en algún lugar.

Para hacer más claro el entregable 1, se muestra un esquema en la Figura 17 que explica el proceso esperado.

Figura 17

Pasos seguidos para completar Entregable 1

1. El usuario proporciona una descripción del estado y los problemas del vehículo. Esto puede presentarse en tres formatos: datos estructurados, datos no estructurados (que se pueden asignar a la taxonomía de la empresa) y datos no estructurados sin asignación

Windscreen ☐ Yes ☐ No

Smoking in vehicle ☐ Yes ☐ No

Mechanical and electrical issues ☐ Yes ☐ No

Had the lights changed and now when the full beams are on message comes up

Additional information ☐ Yes ☐ No

Item(s) matched

Item(s) not matched

2. Cuando es posible, hacer coincidir los elementos del paso 1 con la taxonomía, devolvemos un precio. Cuando no es viable, estimar el precio con el modelo de lenguaje.

Item	Condition	Prep item frequency	Auto / Manual	Cost (<£10K)	Cost (>=£10K)
InteriorExteriorPlasticsScuffed	Interior / exterior plastics scuffed	2,100	Manual	£80	£120
InteriorFabricPanelRepair	Interior fabric panel repair	1,551	Manual	£100	£200
InteriorSwitchButtons	Interior switch / buttons	209	Manual	£38	£75
KeyBattery	Key battery	44	Manual	£8	£15
KeyRemoteNotWorking	Key remote not working	30	Manual	£50	£100



Datos

El primer paso fue obtener el texto en lenguaje natural ingresado por el usuario que desea vender su vehículo a través de la plataforma de la empresa. Los datos se encontraban almacenados en una tabla en BigQuery. BigQuery es un servicio de almacenamiento y análisis de datos en la nube desarrollado por Google, que forma parte de su plataforma de Google Cloud. Es un almacén de

datos altamente escalable y rápido que permite a las organizaciones realizar análisis de grandes cantidades de información en tiempo real mediante consultas SQL.

En la Figura 18, se muestra la estructura del proyecto creado en BigQuery para el desarrollo de la solución. En BigQuery se definen nombre del proyecto (motorway-condition-centre), nombre de dataset (google_prep_fsa_v2) y nombre de tabla (dim_enquiry, prep_item_dictionary, dim_enquiry_vehicle_details, dim_vehicle y free_text_prep_item_match). Para el Entregable 1 se utilizaron las tablas “free_text_prep_item_match” y “prep_item_dictionary”. La Tabla 9 describe brevemente el contenido de las tablas utilizadas, así como información importante al momento de extraer información de estas.

Figura 18

Estructura del proyecto en BigQuery

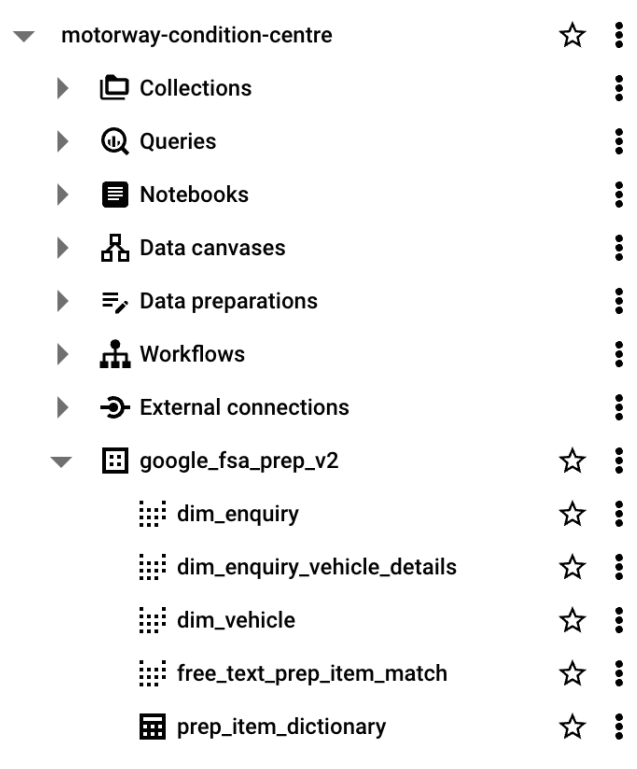


Tabla 9*Descripción de las tablas utilizadas en el Entregable 1*

Nombre de la Tabla	Descripción
	Tabla de descripciones de daños vehiculares que llegan a través de la plataforma digital.
free_text_prep_item_match	– `vehicle_details_id`, `enquiry_id`, `vehicle_key` deben combinarse para recuperar todas las entradas de un vehículo específico – `vehicle_details_id` permitirá la recuperación de la última información proporcionada por el usuario sobre el vehículo, como el kilometraje (sin valores históricos) – `vehicle_key` permitirá la recuperación de detalles del vehículo que son estáticos, como la marca/modelo – Varias filas por vehículo se deben a múltiples entradas `prep_type` coincidentes que coinciden con el campo `Item` de la tabla `prep_item_dictionary`
prep_item_dictionary	Tabla de tipos de daños vehiculares y/o reparaciones válidas en el sistema de la empresa.

En la Tabla 10 se describen los campos relevantes de cada tabla.

Tabla 10*Descripción de campos relevantes de las tablas utilizados en el Entregable 1*

Nombre de la Tabla	Campos relevantes
free_text_prep_item_match	- “cost” es el costo asignado en el sistema de la empresa y puede no necesariamente coincidir con el costo

	asignado para ese artículo en la tabla “prep_item_dictionary”.
	- “prep_origin” contiene 2 valores: manual y automated.
	- “electrical_problems_description” es el texto ingresado por el usuario.
	- “prep_type” es la categoría asignada al texto ingresado por el usuario.
	- “prep_description” es la descripción de la categoría asignada.
	- “Item” es la categoría del daño vehicular.
	- “Condition” es la descripción de la categoría del daño vehicular
	- “Costo” contiene el precio en GBP de la categoría. Se consideró este valor como costo base, pero no exacto, por ejemplo, podría estar bien si está dentro de 2x o 3x este costo.
prep_item_dictionary	

Se analizó el texto en lenguaje natural de la tabla.

“free_text_prep_item_match”, y se calcularon que en promedio el usuario envía 20 palabras, lo cual se traduce como 30 tokens en total. Se encontró además que el texto puede contener una o más categorías de daño vehicular.

La combinación de todas estas distintas fuentes de información permitió crear una única fuente de información fiable en la que basar nuestro análisis y probar el rendimiento de Gemini. Los resultados intermedios y los resultados de las pruebas se guardaron en el servicio “Cloud Storage” de GCP (Google Cloud Platform).

Configuración del ambiente de preparación de datos

La exploración de datos para este proyecto se llevó a cabo en Colab Enterprise. Antes de comenzar a desarrollar, es importante asegurarse de que el notebook esté autorizado para ejecutarse con el IAM del usuario actual mediante el siguiente comando:

```
!gcloud auth application-default login
```

Una vez iniciada la sesión, debemos asegurarnos de que la facturación de la ejecución de los diferentes servicios que se usen esté asociada al proyecto correcto. Para ello es necesario utilizar el siguiente comando:

```
!gcloud auth application-default set-quota-project my-quota-project
```

Por último, para poder utilizar algunos servicios en GCP por primera vez, algunas veces es necesario activar primero la API. Esto se puede hacer con el siguiente comando:

```
!gcloud services enable YOUR_SERVICE_NAME
```

Para abordar el primer paso según la arquitectura conceptual del proyecto, se utilizaron las tablas `prep_item_dictionary` y `free_text_prep_item_match` (Paso 1: Obtener `prepTypes`). Para la fase de precios de `prepltems` se combinaron las diferentes tablas en una sola vista utilizando la siguiente consulta SQL (Paso 2: Recuperar precios de automóviles):

```
SELECT * FROM `motorway-condition-  
centre.google_fsa_prep_v2.free_text_prep_item_match` AS free_text  
INNER JOIN  
  `motorway-condition-centre.google_fsa_prep_v2.dim_enquiry` AS  
dim_enquiry ON free_text.vehicle_details_id =  
dim_enquiry.vehicle_details_id AND free_text.enquiry_id =  
dim_enquiry.enquiry_id  
LEFT JOIN `motorway-condition-centre.google_fsa_prep_v2.dim_vehicle`  
AS dim_vehicle ON dim_vehicle.vehicle_key = dim_enquiry.vehicle_key \  
LEFT JOIN `  
motorway-condition-  
centre.google_fsa_prep_v2.dim_enquiry_vehicle_details` AS  
dim_enquiry_vehicle_details ON dim_enquiry.vehicle_details_id =  
dim_enquiry_vehicle_details.vehicle_details_id
```

Como se detallará en las siguientes secciones, se utilizó Pinecone para almacenar los precios de “`prep_item_dictionary`” y los embeddings de texto para la “descripción” de cada elemento. El código para crear los embeddings se muestra en la Figura 19.

Figura 19

Código para crear embeddings usando VertexAI

```
create_embeddings.py

import vertexai
from vertexai.language_models import TextEmbeddingModel

vertexai.init(project=PROJECT_ID, location=LOCATION)
MODEL_ID = "text-embedding-004"
model = TextEmbeddingModel.from_pretrained(MODEL_ID)

df_dictionary = pd.read_csv("prep_item_dictionary.csv")
df_dictionary = df_dictionary[['Item', 'Condition', 'Auto___Manual', 'Cost']]

def get_embedding(row):
    time.sleep(2) # Keep this if it's necessary for rate limiting
    embedding = model.get_embeddings([row['Condition']])
    for e in embedding:
        vector = e.values
    return vector

df_dictionary['embedding'] = df_dictionary.apply(get_embedding, axis=1)
```

Para cargar los embeddings en el índice de Pinecone usamos el siguiente script mostrado en la Figura 20.

Figura 20

Código para cargar los embeddings a Pinecone

```
load_embeddings.py

from pinecone import Pinecone
import uuid

index_name = "prep-items"
pc = Pinecone(api_key=PINECONE_API_KEY)
index = pc.Index(index_name)
df_dictionary["Cost"] = df_dictionary["Cost"].fillna(0)
for i, row in df_dictionary.iterrows():
    index.upsert(
        vectors=[
            {
                "id": str(uuid.uuid4()),
                "values": row["embedding"],
                "metadata": {"item": row["Item"], "condition": row["Condition"],
                            "cost": row["Cost"]}
            }
        ],
        namespace="prompts"
    )
```

Modelo de Lenguaje Natural

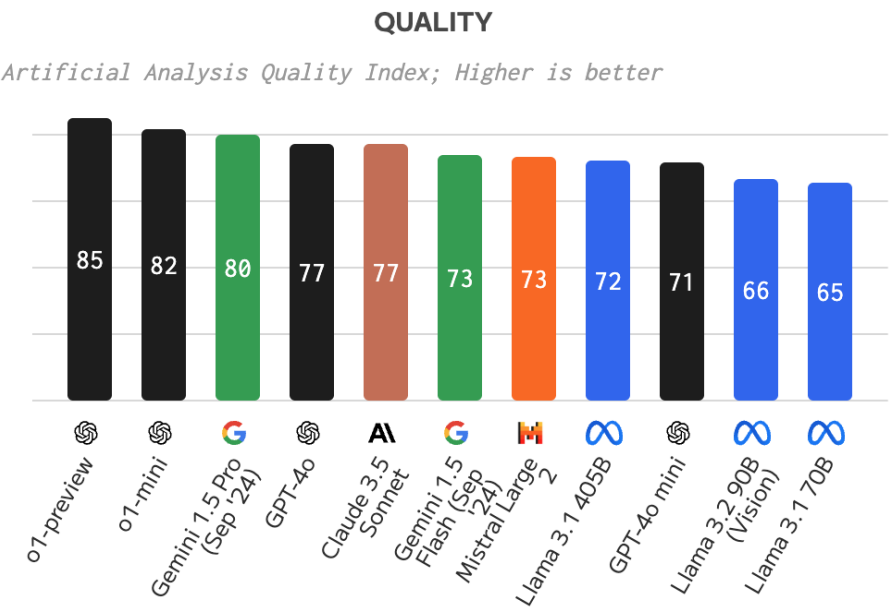
Se utilizó un LLM para encontrar las categorías correspondientes al texto. La elección del LLM estuvo basada en dos criterios:

1. El precio por token de entrada y salida.
2. El rendimiento del modelo para tareas de clasificación.
3. Opción de utilizar “caching” en el modelo para contextos largos.

Actualmente existe una gran variedad de LLMs y el número crece de manera diaria. Sin embargo, podemos mencionar que los proveedores de modelos más conocidos y con más uso son: Open AI, Google, Meta y Anthropic. En base a la rúbrica descrita anteriormente y después de analizar los modelos de los diferentes proveedores (ver Figuras 21, 22 y 23) se decidió utilizar Gemini 1.5 Flash (versión 002) de Google.

Figura 21

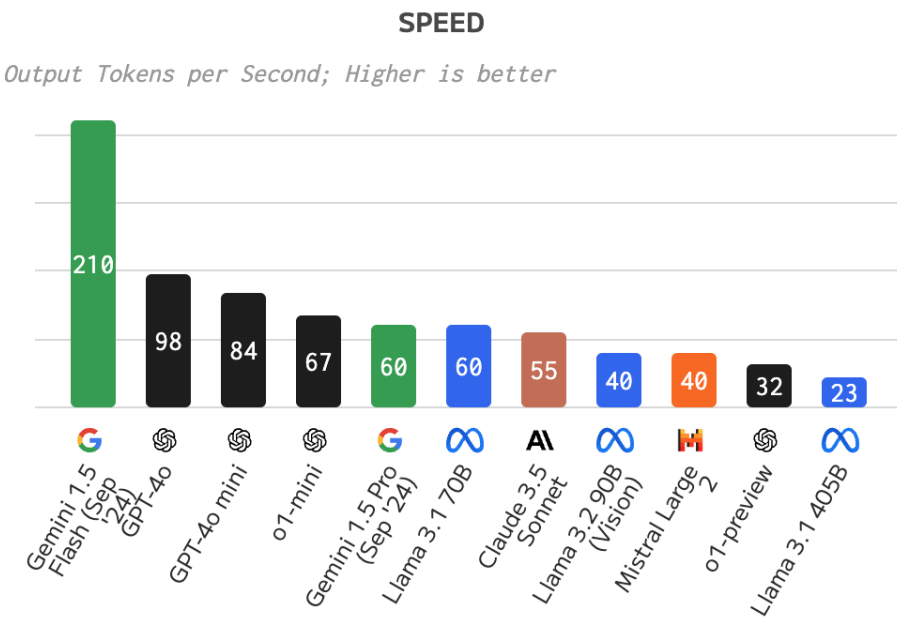
Análisis del índice calidad de modelos



Nota. Tomado de artificialanalysis.ai

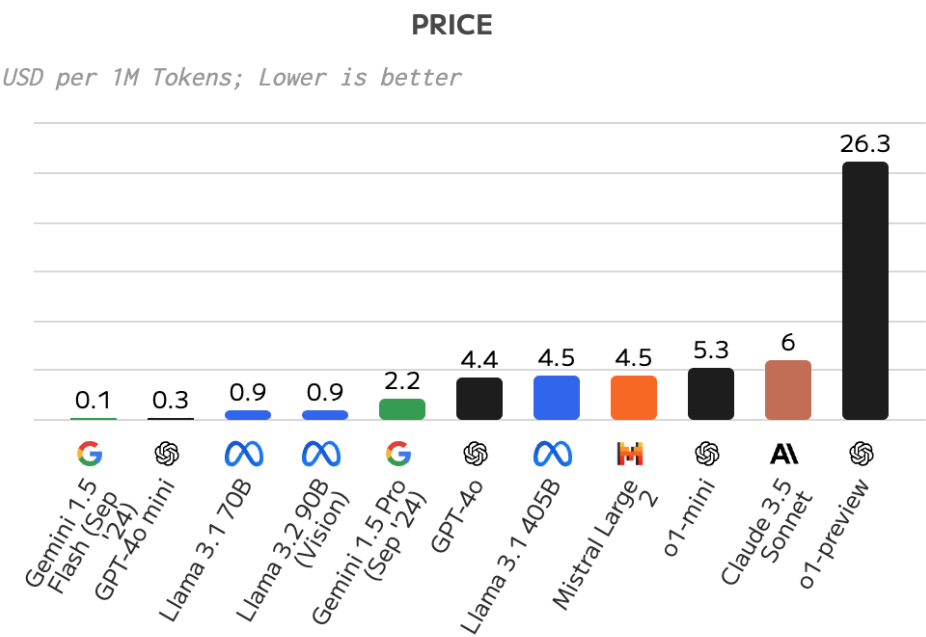
Figura 22

Análisis de tokens de salida por segundo



Nota. Tomado de artificialanalysis.ai. En esta gráfica, un valor mayor es mejor.

Figura 23
Análisis de precios



Nota. Tomado de artificialanalysis.ai. En esta gráfica, un valor menor es mejor.

La familia de modelos de Gemini incluye modelos que funcionan con entradas de texto y multimodal (es decir el modelo es capaz de entender audio, voz y video). En la Tabla 11 se listan los modelos de la familia de Gemini.

Tabla 11

Listado de modelos de Gemini y características principales

Modelo	Versión	Características principales del modelo
Gemini 1.5 Flash	gemini-1.5-flash-001	Versión inicial de Gemini 1.5 Flash.
	gemini-1.5-flash-002	Mejora de la calidad general del modelo con importantes mejoras en las siguientes categorías: - Facticidad y reducción de alucinaciones en el modelo. - Preguntas y respuestas de libro abierto para casos de uso de RAG. - Seguimiento de instrucciones.

		- Comprensión multilingüe en 102 idiomas, especialmente en coreano, francés, alemán, español, japonés, ruso y chino. - Generación de SQL. - Comprensión de audio. - Comprensión de documentos. - Contexto extenso. - Matemáticas y razonamiento.
	gemini-1.5-pro-001	Versión inicial de Gemini 1.5 Flash. Mejora de la calidad general del modelo con importantes mejoras en las siguientes categorías: - Facticidad y reducción de alucinaciones en el modelo. - Preguntas y respuestas de libro abierto para casos de uso de RAG. - Seguimiento de instrucciones. - Comprensión multilingüe en 102 idiomas, especialmente en coreano, francés, alemán, español, japonés, ruso y chino. - Generación de SQL. - Comprensión de audio. - Comprensión de documentos. - Contexto extenso. - Matemáticas y razonamiento.
Gemini 1.5 Pro	gemini-1.5-pro-002	Ejecuta inferencias sobre imágenes y vídeos. Cuando se le pasa una imagen, una serie de imágenes o un vídeo, Gemini puede: - Describir o responder preguntas sobre el contenido.
Gemini 1.0 Pro Vision	gemini-1.0-pro-001 gemini-1.0-pro-vision-001	

- Resumir el contenido.
- Extrapolar a partir del contenido.

Gemini 1.0 Pro	gemini-1.0-pro gemini-1.0-pro-001 gemini-1.0-pro-002	Versión inicial de Gemini.
----------------	--	----------------------------

Para controlar la salida del modelo, se estableció el parámetro de temperatura en cero y el parámetro top-K en uno. Las tareas de clasificación suelen ser determinísticas, por lo que con esta configuración se obtuvieron mejores resultados.

Ingeniería de instrucciones (Prompt Engineering)

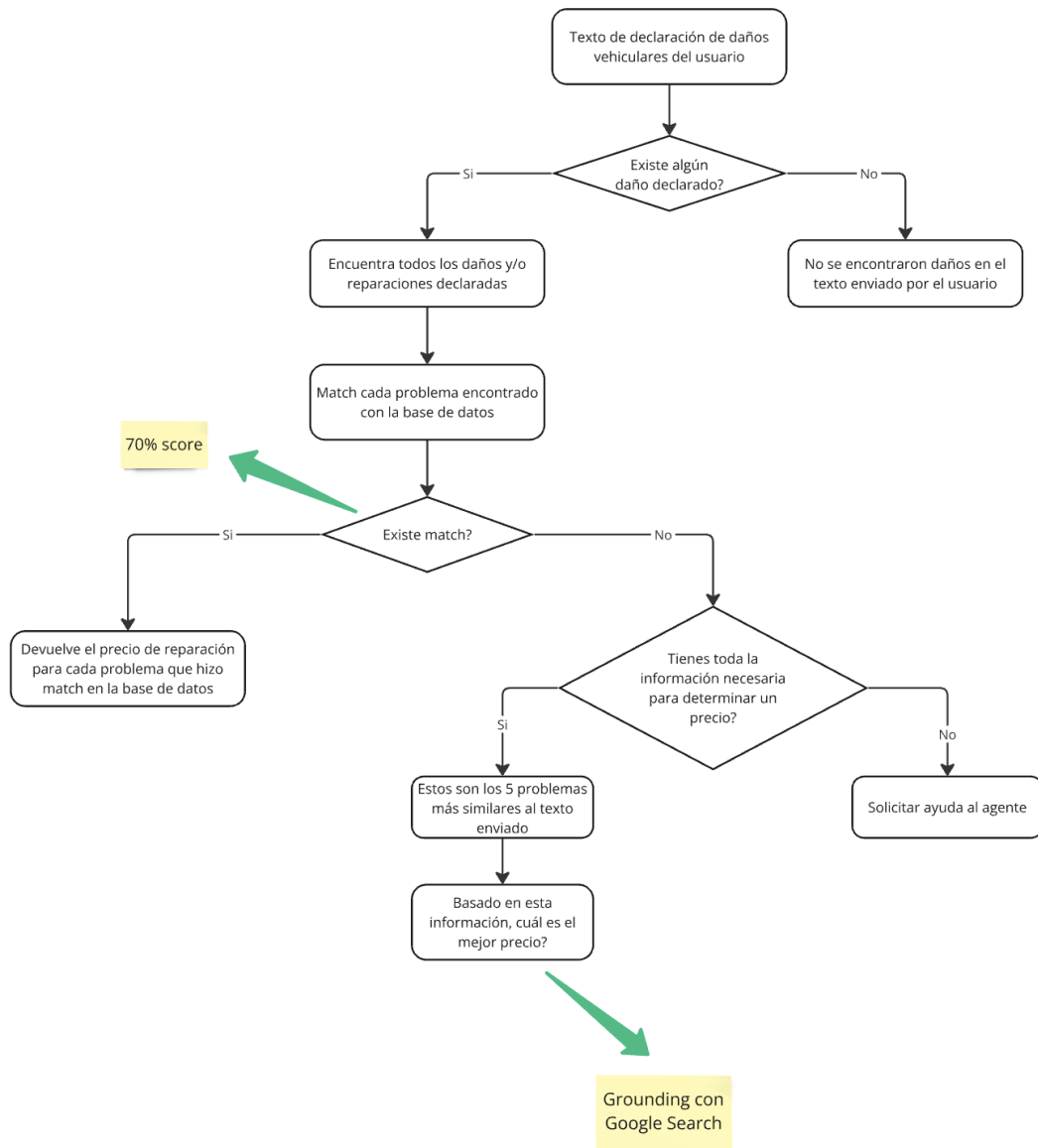
Obtener respuestas de un LLM que se ajusten a lo que necesitemos, considerando diferentes casos de uso, es una tarea de prueba y error. Realizar pruebas y evaluaciones exhaustivas sigue siendo fundamental para optimizar el rendimiento del modelo.

Después de muchas iteraciones se decidió utilizar instrucciones (prompts) pequeños y que ejecuten una única tarea. La clave para implementar prompt engineering con éxito es diseñar un proceso de razonamiento con pasos claros y definidos.

En la Figura 24, se muestra el flujo de razonamiento que el LLM debe seguir. Algunos estados en el diagrama representan un prompt y una llamada al modelo de forma independiente.

Figura 24

Flujo de razonamiento que debe ser seguido por el LLM



Para el primer entregable, se diseñaron los siguientes prompts:

1. “Existe algún daño declarado?”

- Instrucción del sistema: *“You are an expert car mechanic and your job is to identify if a given description by a car owner represents a vehicle issue.”*

2. “Encuentra todos los daños y/o reparaciones necesarias que han sido declaradas”

- Instrucción del sistema: *“You are an expert car mechanic, and your job is to identify issues based on a description given by a car owner.”*

They can either be descriptions of damage, repairs to be carried out, or items needed to replace.”

- Prompt: “Given the following user input: {user_input}, return a list of the issues you have identified.”

Base de datos de vectores y embeddings de texto

Para el paso “Encuentra el match del daño declarado con una categoría de la base de datos” se utilizó una base de datos de vectores sin servidor (serverless) en Pinecone. Pinecone es una base de datos vectorial basada en la nube para aplicaciones de machine learning.

En la tabla 12 se muestran las configuraciones de la base de datos de vectores y la descripción de cada parámetro.

Tabla 12

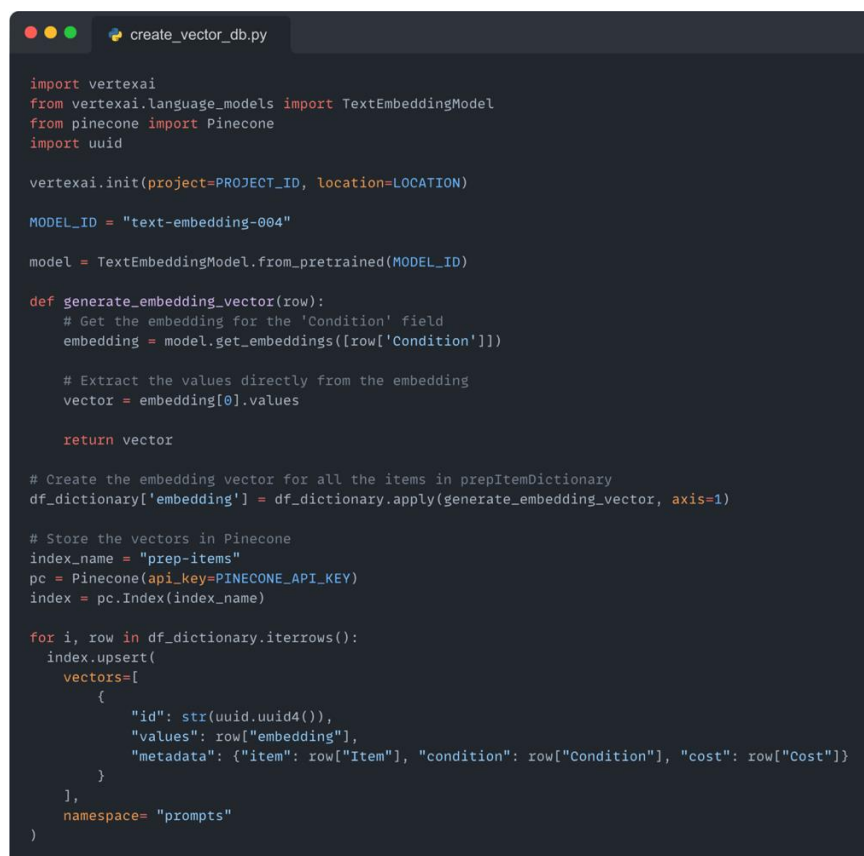
Parámetros de la base de datos de vectores en Pinecone

Parámetro	Descripción	Valor
Nombre de índice	Nombre del índice donde se almacenan los vectores	prep
Dimensión	Describe el tamaño del vector de embeddings. Es decir, la cantidad de elementos de punto flotante que componen un vector determinado.	768
Métrica de similitud	Al crear un índice, se puede elegir entre las siguientes métricas de similitud: euclidean, cosine y dotproduct. Para obtener resultados más precisos, se recomienda elegir la métrica de similitud utilizada para entrenar el modelo de embeddings para los vectores.	coseno
Espacios de nombres	Pinecone le permite dividir los registros de un índice en espacios de nombres. Las consultas y otras operaciones se limitan a un espacio de nombres, por lo que las solicitudes pueden buscar en distintos subconjuntos de un índice.	prompts

Pinecone ofrece un único modelo de embeddings multilingual, sin embargo, para este proyecto se decidió utilizar un modelo propietario de Google. Este modelo llamado “*text-embedding-004*” utiliza representaciones de vectores densos, por ejemplo, utiliza vectores de 768 dimensiones. Los modelos de embeddings de vectores densos utilizan métodos de machine learning similares a los que utilizan los LLMs. A diferencia de los vectores dispersos, que tienden a asignar directamente palabras a números, los vectores densos están diseñados para representar mejor el significado de un fragmento de texto. El beneficio de usar embeddings de vectores densos en la IA generativa es que, en lugar de buscar coincidencias directas de palabras o sintaxis, puede buscar mejores fragmentos que se alineen con el significado de la consulta, incluso si los fragmentos no utilizan el mismo idioma. En la figura 25 se muestra el código utilizado para crear los vectores con el modelo de embeddings de texto de Google y luego almacenarlos en Pinecone.

Figura 25

Código que crea y almacena los vectores en la base de datos de vectores



```
import vertexai
from vertexai.language_models import TextEmbeddingModel
from pinecone import Pinecone
import uuid

vertexai.init(project=PROJECT_ID, location=LOCATION)

MODEL_ID = "text-embedding-004"

model = TextEmbeddingModel.from_pretrained(MODEL_ID)

def generate_embedding_vector(row):
    # Get the embedding for the 'Condition' field
    embedding = model.get_embeddings([row['Condition']])

    # Extract the values directly from the embedding
    vector = embedding[0].values

    return vector

# Create the embedding vector for all the items in prepItemDictionary
df_dictionary['embedding'] = df_dictionary.apply(generate_embedding_vector, axis=1)

# Store the vectors in Pinecone
index_name = "prep-items"
pc = Pinecone(api_key=PINECONE_API_KEY)
index = pc.Index(index_name)

for i, row in df_dictionary.iterrows():
    index.upsert(
        vectors=[
            {
                "id": str(uuid.uuid4()),
                "values": row["embedding"],
                "metadata": {"item": row["Item"], "condition": row["Condition"], "cost": row["Cost"]}
            }
        ],
        namespace="prompts"
    )
```

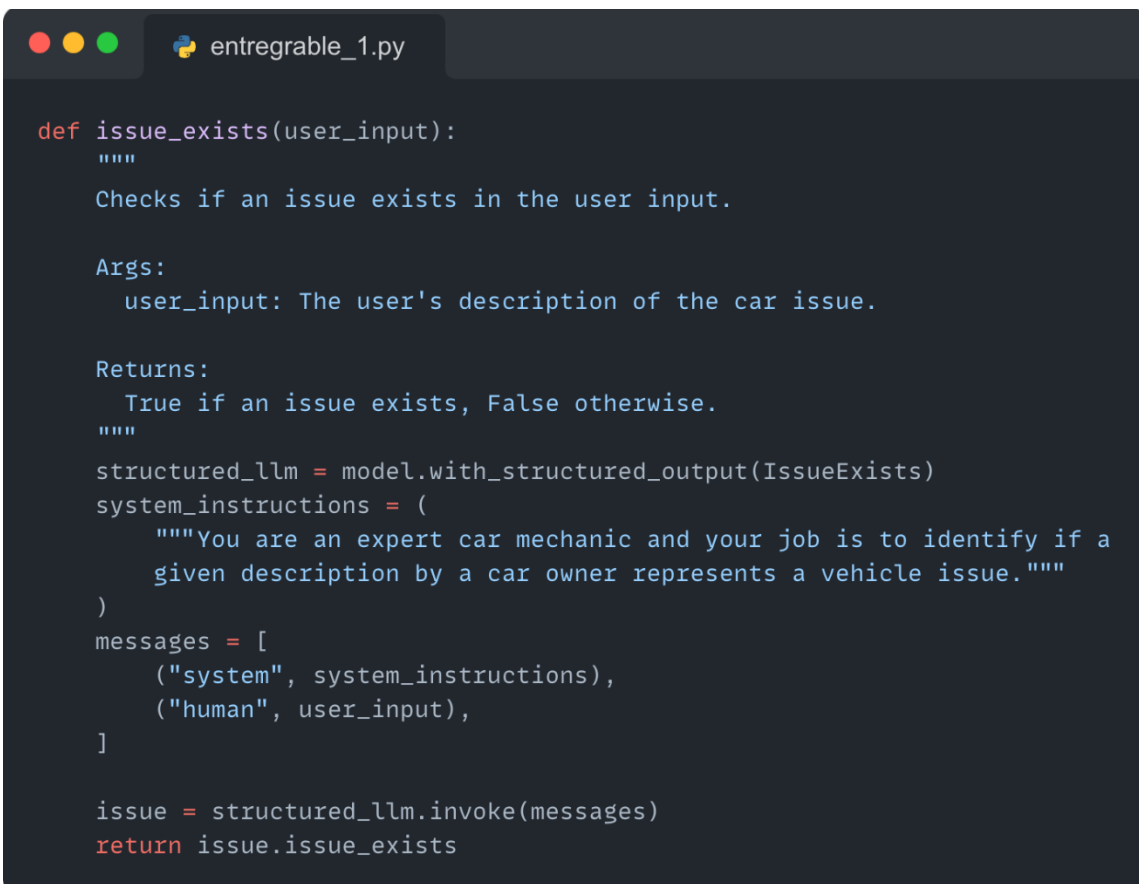
Uso de herramientas y servicios

Para utilizar el LLM se utilizaron herramientas de software libre como LangChain y el SDK de Vertex AI. LangChain permite a los desarrolladores la construcción de soluciones con LLMs conocidas como agentes. Estos agentes son capaces de razonar sobre problemas y dividirlo en tareas pequeñas. Al encadenar comandos complejos puede agregar contexto y memoria al LLM.

Vertex AI es una plataforma de machine learning que permite entrenar e implementar modelos y aplicaciones de IA. Además, permite personalizar modelos grandes de lenguaje (LLM). En este proyecto se utilizó el SDK de Vertex AI para hacer uso del LLM y el modelo de embeddings de texto. Haciendo uso de las herramientas anteriores, las Figuras 26, 27 y 28 representan el código desarrollado para el Entregable 1.

Figura 26

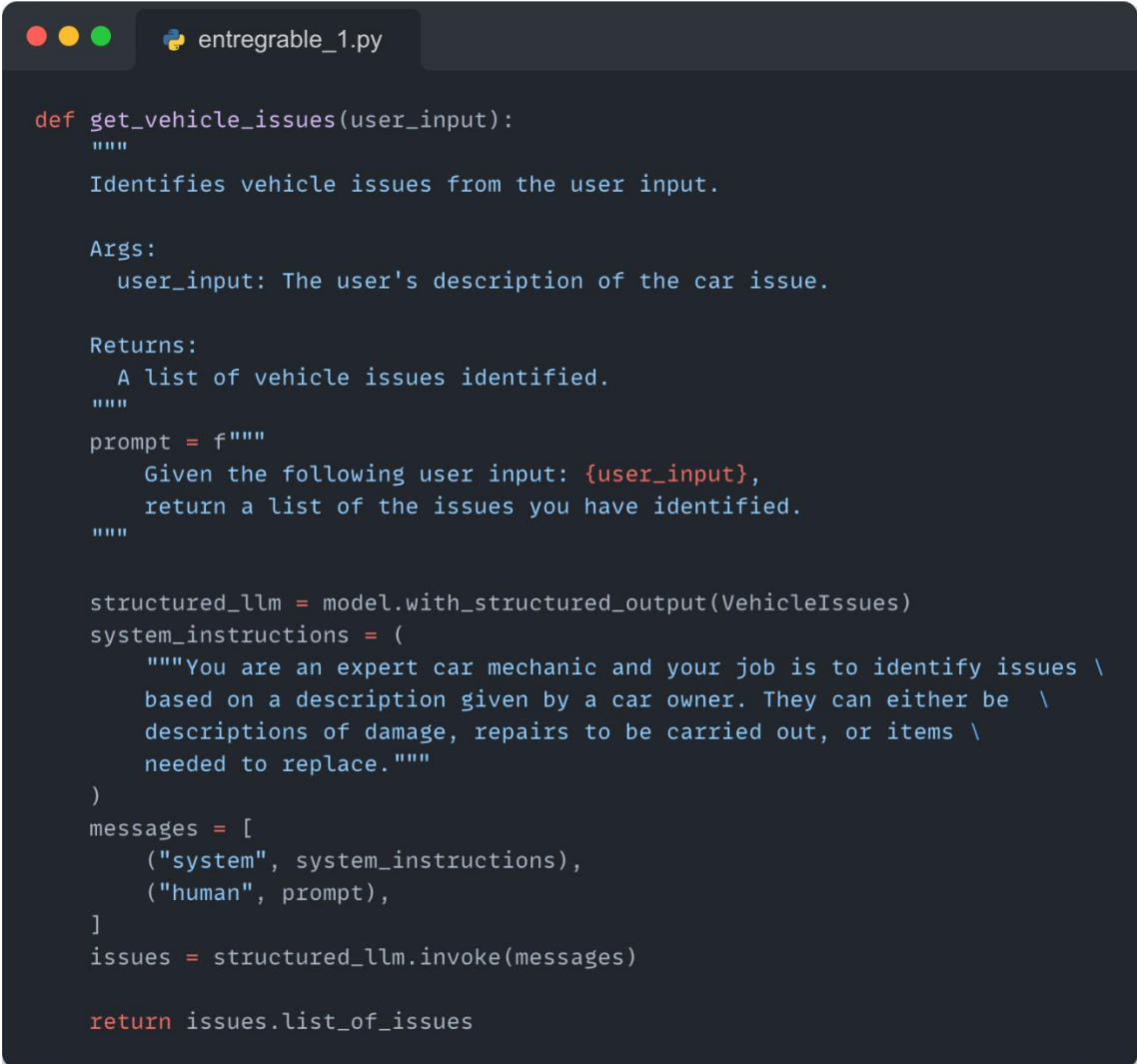
Código que utiliza un LLM para determinar si existe declaración de daño vehicular



```
def issue_exists(user_input):  
    """  
    Checks if an issue exists in the user input.  
  
    Args:  
        user_input: The user's description of the car issue.  
  
    Returns:  
        True if an issue exists, False otherwise.  
    """  
    structured_llm = model.with_structured_output(IssueExists)  
    system_instructions = (  
        """You are an expert car mechanic and your job is to identify if a  
        given description by a car owner represents a vehicle issue."""  
    )  
    messages = [  
        ("system", system_instructions),  
        ("human", user_input),  
    ]  
  
    issue = structured_llm.invoke(messages)  
    return issue.issue_exists
```

Figura 27

Código que utiliza un LLM para determinar encontrar cualquier daño declarado en un texto



```
def get_vehicle_issues(user_input):  
    """  
    Identifies vehicle issues from the user input.  
  
    Args:  
        user_input: The user's description of the car issue.  
  
    Returns:  
        A list of vehicle issues identified.  
    """  
    prompt = f"""  
        Given the following user input: {user_input},  
        return a list of the issues you have identified.  
    """  
  
    structured_llm = model.with_structured_output(VehicleIssues)  
    system_instructions = (  
        """You are an expert car mechanic and your job is to identify issues \\  
        based on a description given by a car owner. They can either be \\  
        descriptions of damage, repairs to be carried out, or items \\  
        needed to replace."""  
    )  
    messages = [  
        ("system", system_instructions),  
        ("human", prompt),  
    ]  
    issues = structured_llm.invoke(messages)  
  
    return issues.list_of_issues
```

Figura 28

Código que utiliza embeddings de texto y una base de datos de vectores para encontrar categorías

```
def prep_type_match(
    vehicle_issue, top_k, index_name=INDEX_NAME, embedding_model=EMBEDDING_MODEL
):
    """
    Matches the vehicle issue to a pre-defined type using Pinecone.

    Args:
        index_name: The name of the Pinecone index.
        embedding_model: The embedding model to use.
        vehicle_issue: The vehicle issue to match.

    Returns:
        A tuple containing the matched pre-defined type and its associated cost,
        or None if no match is found.
    """
    pc = Pinecone(api_key=PINECONE_API_KEY)
    index = pc.Index(index_name)

    query = embedding_model.get_embeddings([vehicle_issue])
    vector = next(iter(query)).values

    result = index.query(
        namespace="prompts",
        vector=list(vector),
        top_k=top_k,
        include_metadata=True,
    )

    prep_type_matches = []
    for match_ in result["matches"]:
        score = match_["score"]
        cost = match_["metadata"]["cost"]
        prep_item = match_["metadata"]["item"]
        prep_type_matches.append(
            {
                "vehicle_issue": vehicle_issue,
                "score": score,
                "cost": extract_numeric_value(cost),
                "prep_item": prep_item,
            }
        )

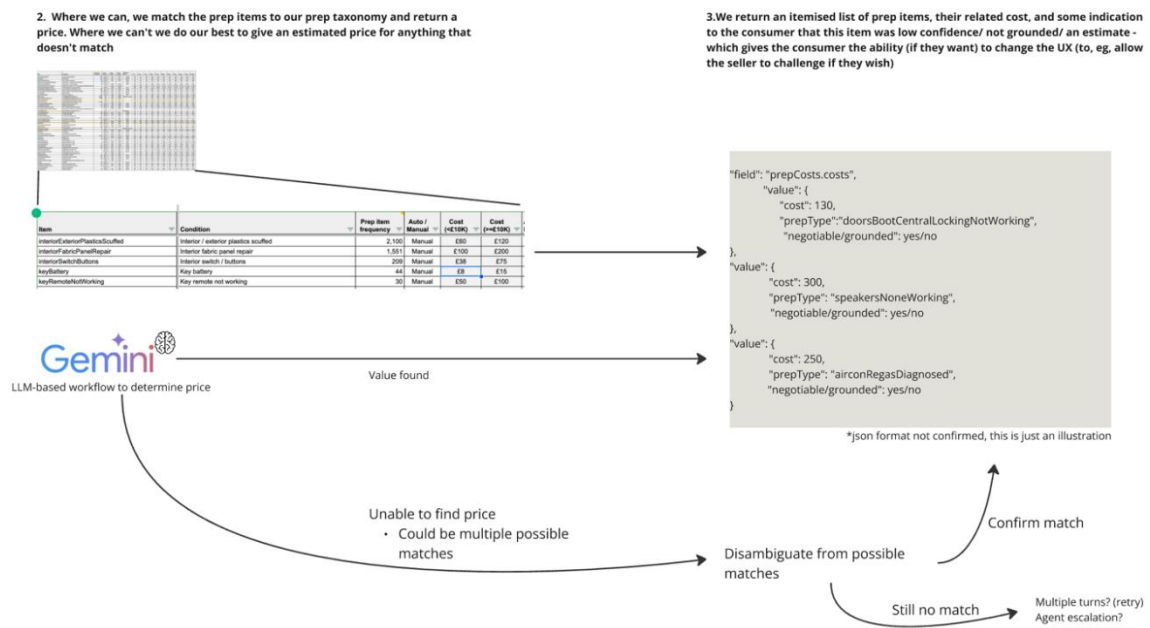
    return prep_type_matches
```

3.2.5.3.2. ENTREGABLE 2

El entregable de la etapa 2 implica encontrar un precio para aquellos artículos dañados sin coincidencia en la base de datos de vectores. La Figura 29 muestra los pasos a seguir para completar el Entregable 2.

Figura 29

Pasos seguidos para completar Entregable 2



Considerando el proceso de razonamiento diseñado en la Figura 24 se diseñaron los siguientes prompts:

1. “Tienes toda la información necesaria para encontrar un precio para este daño vehicular?”
 - Instrucción del sistema: “*You are an expert car mechanic and your job is to determine if you have all the information required to find a price for a car issue*”
 - Prompt: “*Given the following user input: {user_input} and vehicle details characteristics {vehicle_details}, is it possible to find a price in order to fix the car issues?*”
2. “Estos son los 5 elementos más parecidos. Consideras que es suficiente información para encontrar un precio?”
 - Instrucción del sistema: “*You are an expert car mechanic and your job*

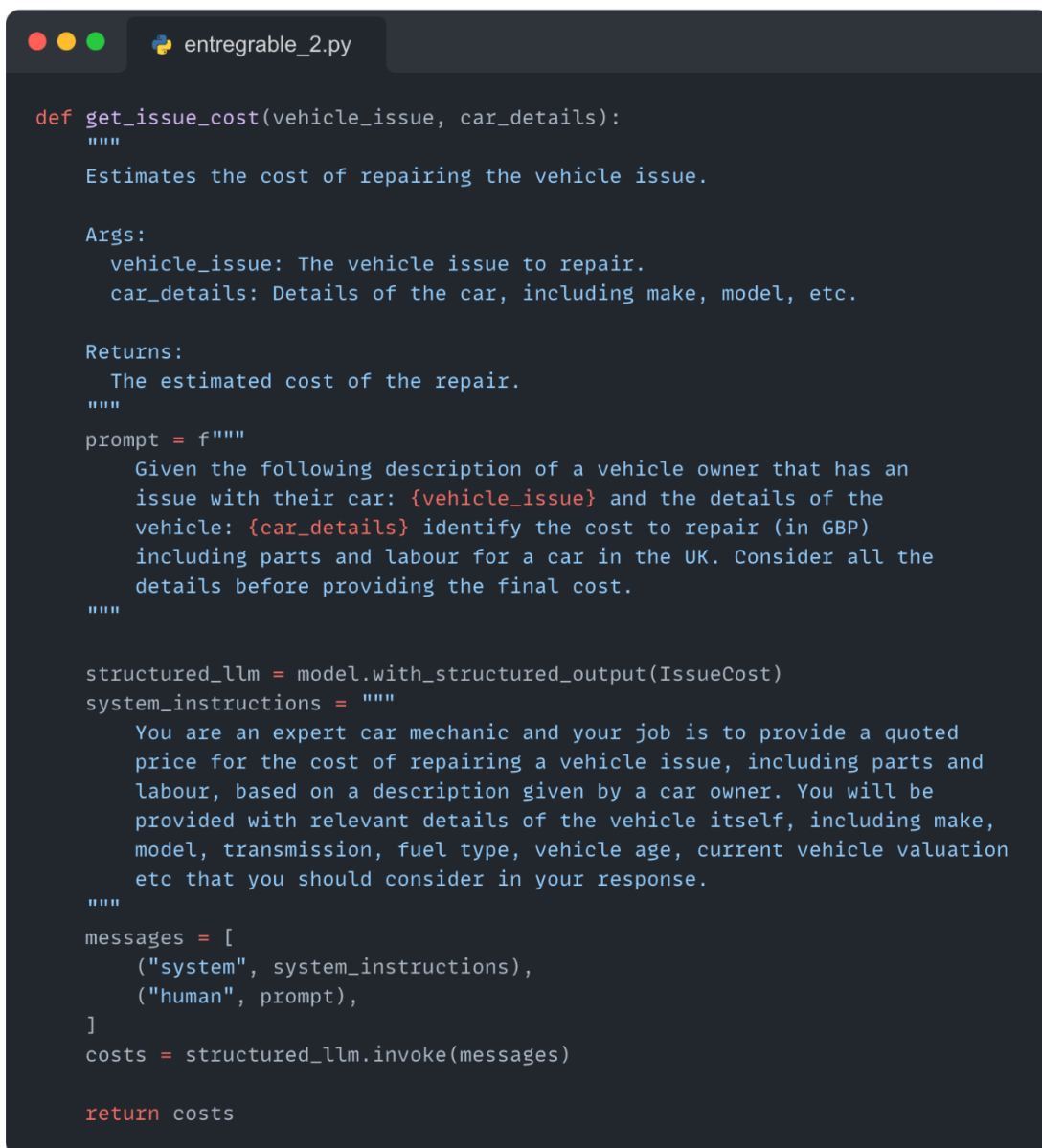
is to determine the price of fixing car issues given the information of a user”

- Prompt: *“Based on the following: {similar_items}, what do you think is a good price for this item?”*

La Figura 30 representa el código desarrollado para el Entregable 2.

Figura 30

Código que estima los costos de reparación de un vehículo



```
def get_issue_cost(vehicle_issue, car_details):  
    """  
    Estimates the cost of repairing the vehicle issue.  
  
    Args:  
        vehicle_issue: The vehicle issue to repair.  
        car_details: Details of the car, including make, model, etc.  
  
    Returns:  
        The estimated cost of the repair.  
    """  
    prompt = f"""  
    Given the following description of a vehicle owner that has an  
    issue with their car: {vehicle_issue} and the details of the  
    vehicle: {car_details} identify the cost to repair (in GBP)  
    including parts and labour for a car in the UK. Consider all the  
    details before providing the final cost.  
    """  
  
    structured_llm = model.with_structured_output(IssueCost)  
    system_instructions = """  
    You are an expert car mechanic and your job is to provide a quoted  
    price for the cost of repairing a vehicle issue, including parts and  
    labour, based on a description given by a car owner. You will be  
    provided with relevant details of the vehicle itself, including make,  
    model, transmission, fuel type, vehicle age, current vehicle valuation  
    etc that you should consider in your response.  
    """  
    messages = [  
        ("system", system_instructions),  
        ("human", prompt),  
    ]  
    costs = structured_llm.invoke(messages)  
  
    return costs
```

3.2.5.3.3. ENTREGABLE 3

En la etapa final del proyecto, se requirió buscar métodos que permitieran asegurar que los valores generados por el modelo de lenguaje sean valores que correspondan a la realidad. Es decir, que los precios generados por el LLM estén de acuerdo con los precios del mercado actual.

Para conseguir esto se utilizó una técnica conocida como fundamentación (grounding) con Google Search. Se puede observar el código en la Figura 31.

Figura 31

Código que permite fundamentar la respuesta de un LLM

```
entregable_3.py

def print_grounding_response(response):
    """Prints Gemini response with grounding citations."""

    # Access grounding_metadata from response_metadata dictionary
    grounding_metadata = response.response_metadata["grounding_metadata"]

    # Access grounding_supports from grounding_metadata dictionary
    grounding_supports = grounding_metadata["grounding_supports"]

    # Citation indices are in byte units
    ENCODING = "utf-8"
    text_bytes = response.content.encode(ENCODING)

    prev_index = 0
    markdown_text = ""

    for grounding_support in grounding_supports:
        # Access end_index from grounding_support dictionary
        end_index = grounding_support["segment"]["end_index"]

        text_segment = text_bytes[prev_index:end_index].decode(ENCODING)

        footnotes_text = ""
        # Access grounding_chunk_indices from grounding_support dictionary
        for grounding_chunk_index in grounding_support["grounding_chunk_indices"]:
            footnotes_text += f"[{grounding_chunk_index + 1}]"

        markdown_text += f"{text_segment} {footnotes_text}\n"
        prev_index = end_index

    if prev_index < len(text_bytes):
        markdown_text += str(text_bytes[prev_index:], encoding=ENCODING)

    markdown_text += "\n---\n## Grounding Sources\n"

    if grounding_metadata["web_search_queries"]:
        markdown_text += (
            f"\n**Web Search Queries:** {grounding_metadata['web_search_queries']}\n"
        )
    if grounding_metadata["search_entry_point"]:
        markdown_text += f"\n**Search Entry Point:**\n {grounding_metadata['search_entry_point']['rendered_content']}\n"
    elif grounding_metadata["retrieval_queries"]:
        markdown_text += (
            f"\n**Retrieval Queries:** {grounding_metadata['retrieval_queries']}\n"
        )

    markdown_text += "### Grounding Chunks\n"

    for index, grounding_chunk in enumerate(
        grounding_metadata["grounding_chunks"], start=1
    ):
        if grounding_chunk["web"]:
            context = grounding_chunk["web"]
        elif grounding_chunk["retrieved_context"]:
            context = grounding_chunk["retrieved_context"]
        else:
            # print(f"Skipping Grounding Chunk {grounding_chunk}")
            continue

        # Access title and uri from context dictionary
        markdown_text += f"{index}. [{context['title']}]({context['uri']})\n"

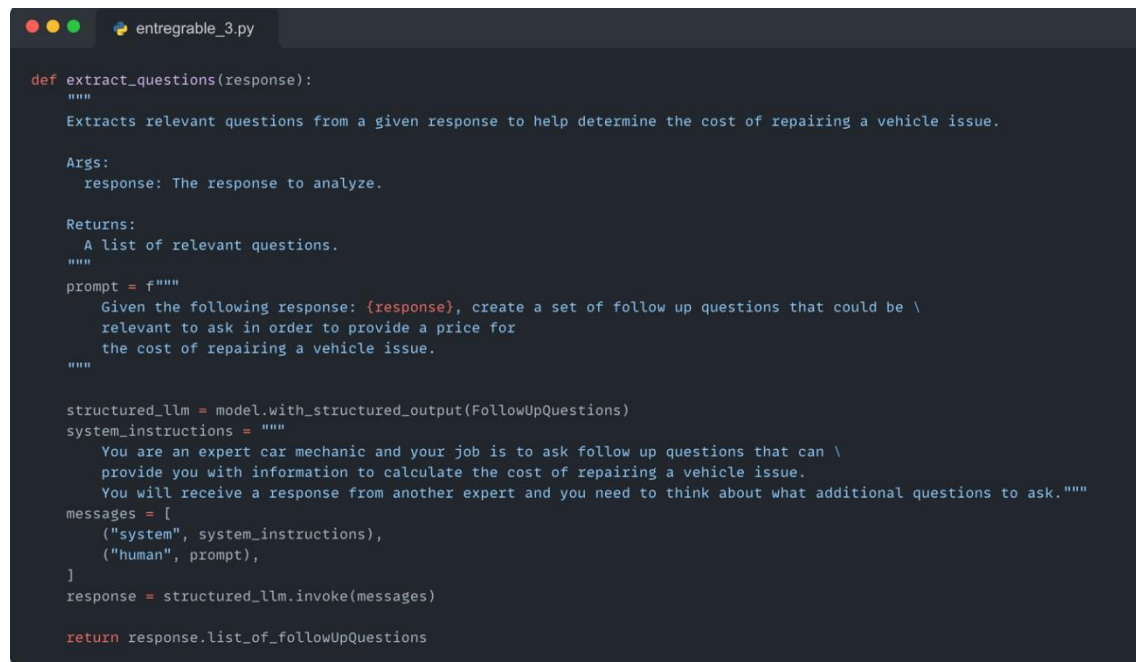
    return markdown_text
```

Una vez obtenido los resultados fundamentados con datos públicos de Internet, se decidió generar un cuestionario de 3 preguntas relevantes como se puede ver en la Figura 32. Esto le permite a la empresa retornar esas preguntas durante el proceso de declaración de daño vehicular, haciendo que el usuario

pueda responder dichas preguntas y el proceso de valoración termine 100% online sin intervención de un agente humano.

Figura 32

Código que permite extraer preguntas



```
def extract_questions(response):
    """
    Extracts relevant questions from a given response to help determine the cost of repairing a vehicle issue.

    Args:
        response: The response to analyze.

    Returns:
        A list of relevant questions.
    """
    prompt = f"""
    Given the following response: {response}, create a set of follow up questions that could be \
    relevant to ask in order to provide a price for \
    the cost of repairing a vehicle issue.
    """

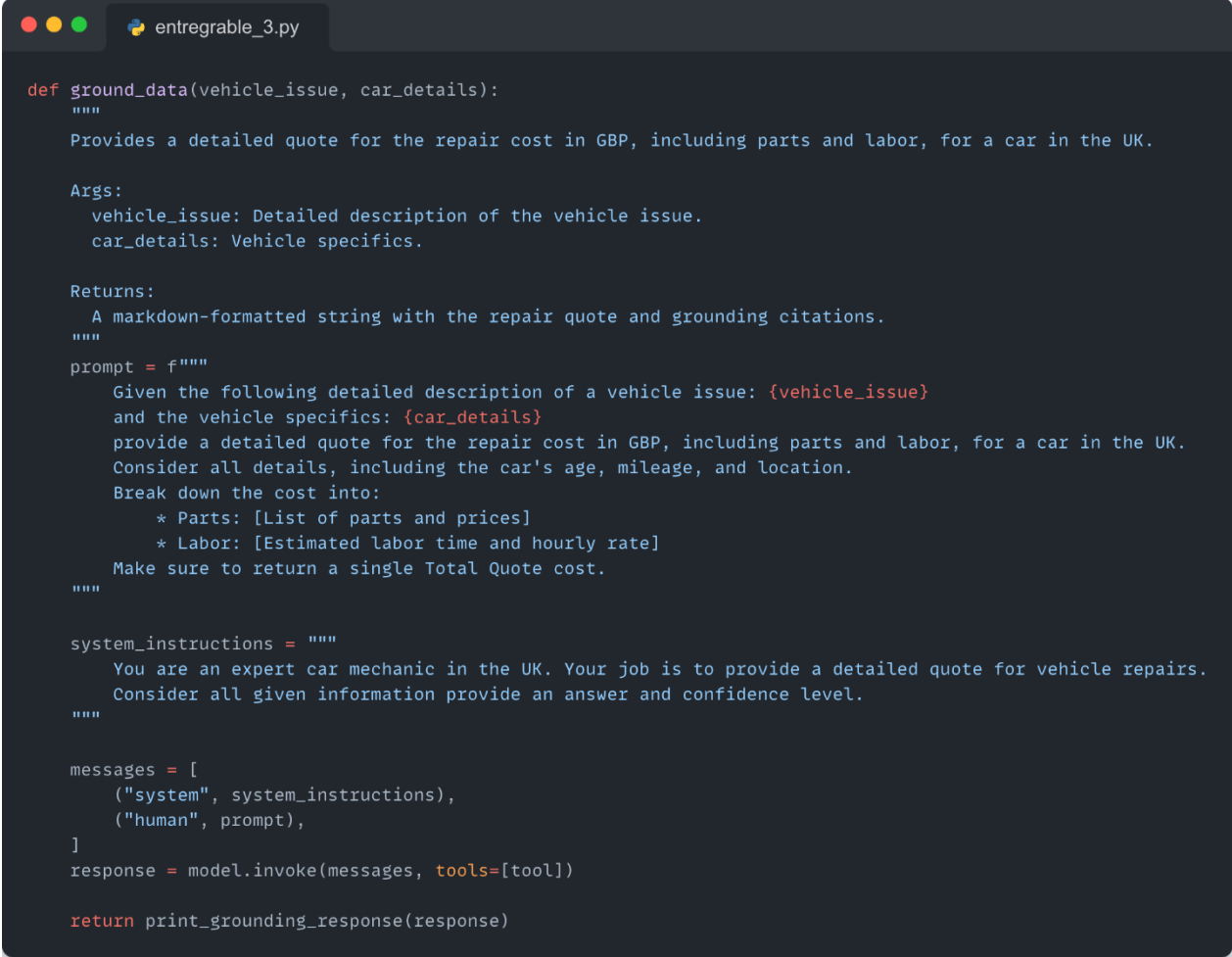
    structured_llm = model.with_structured_output(FollowUpQuestions)
    system_instructions = """
    You are an expert car mechanic and your job is to ask follow up questions that can \
    provide you with information to calculate the cost of repairing a vehicle issue.
    You will receive a response from another expert and you need to think about what additional questions to ask."""
    messages = [
        ("system", system_instructions),
        ("human", prompt),
    ]
    response = structured_llm.invoke(messages)

    return response.list_of_followUpQuestions
```

Con la información fundamentada y validada, se creó una función auxiliar que permite obtener una cotización que considera la mano de obra y el costo de las piezas a reparar. El código implementado para esta tarea se puede encontrar en la Figura 33.

Figura 33

Código que genera un presupuesto total de reparación de daños vehiculares declarados

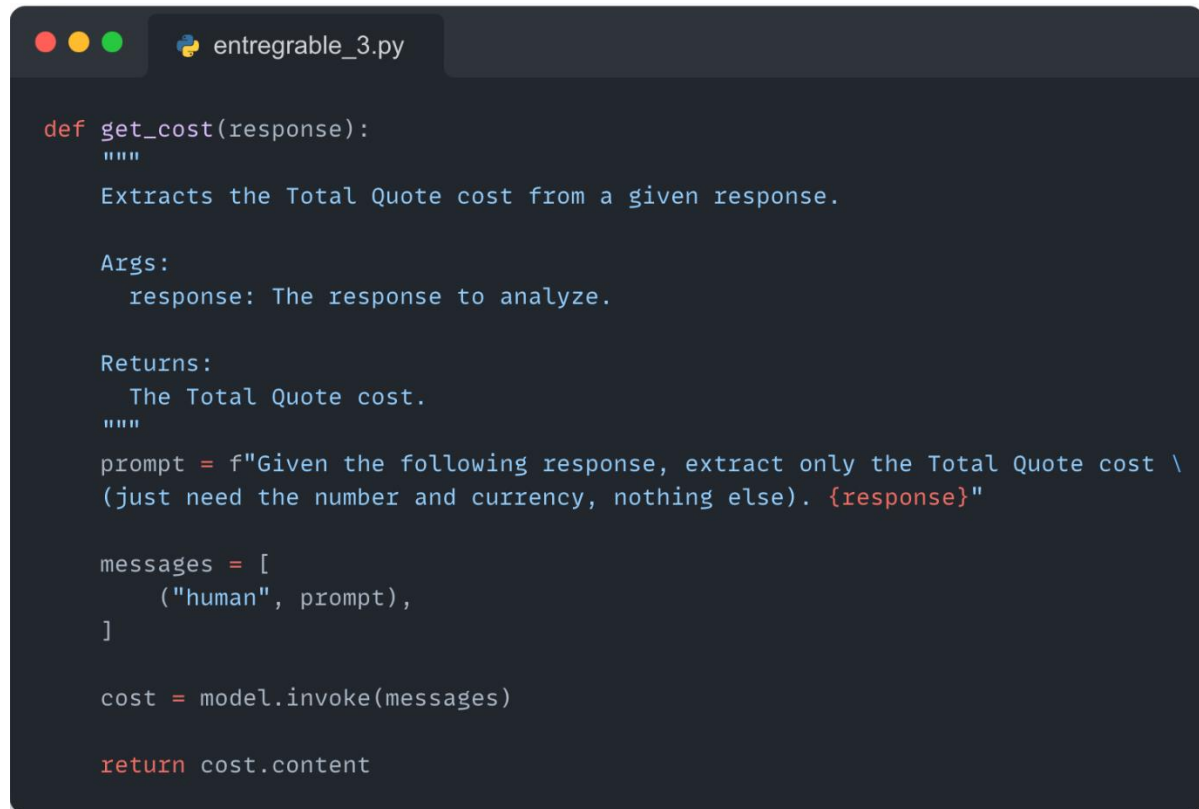


```
def ground_data(vehicle_issue, car_details):  
    """  
    Provides a detailed quote for the repair cost in GBP, including parts and labor, for a car in the UK.  
  
    Args:  
        vehicle_issue: Detailed description of the vehicle issue.  
        car_details: Vehicle specifics.  
  
    Returns:  
        A markdown-formatted string with the repair quote and grounding citations.  
    """  
    prompt = f"""  
    Given the following detailed description of a vehicle issue: {vehicle_issue}  
    and the vehicle specifics: {car_details}  
    provide a detailed quote for the repair cost in GBP, including parts and labor, for a car in the UK.  
    Consider all details, including the car's age, mileage, and location.  
    Break down the cost into:  
        * Parts: [List of parts and prices]  
        * Labor: [Estimated labor time and hourly rate]  
    Make sure to return a single Total Quote cost.  
    """  
  
    system_instructions = """  
    You are an expert car mechanic in the UK. Your job is to provide a detailed quote for vehicle repairs.  
    Consider all given information provide an answer and confidence level.  
    """  
  
    messages = [  
        ("system", system_instructions),  
        ("human", prompt),  
    ]  
    response = model.invoke(messages, tools=[tool])  
  
    return print_grounding_response(response)
```

Finalmente, utilizando la cotización generada se procedió a crear una función que le pide al modelo extraer únicamente el precio final como se ilustra en la Figura 34.

Figura 34

Código que permite extraer el precio final de reparación de los daños vehiculares declarados



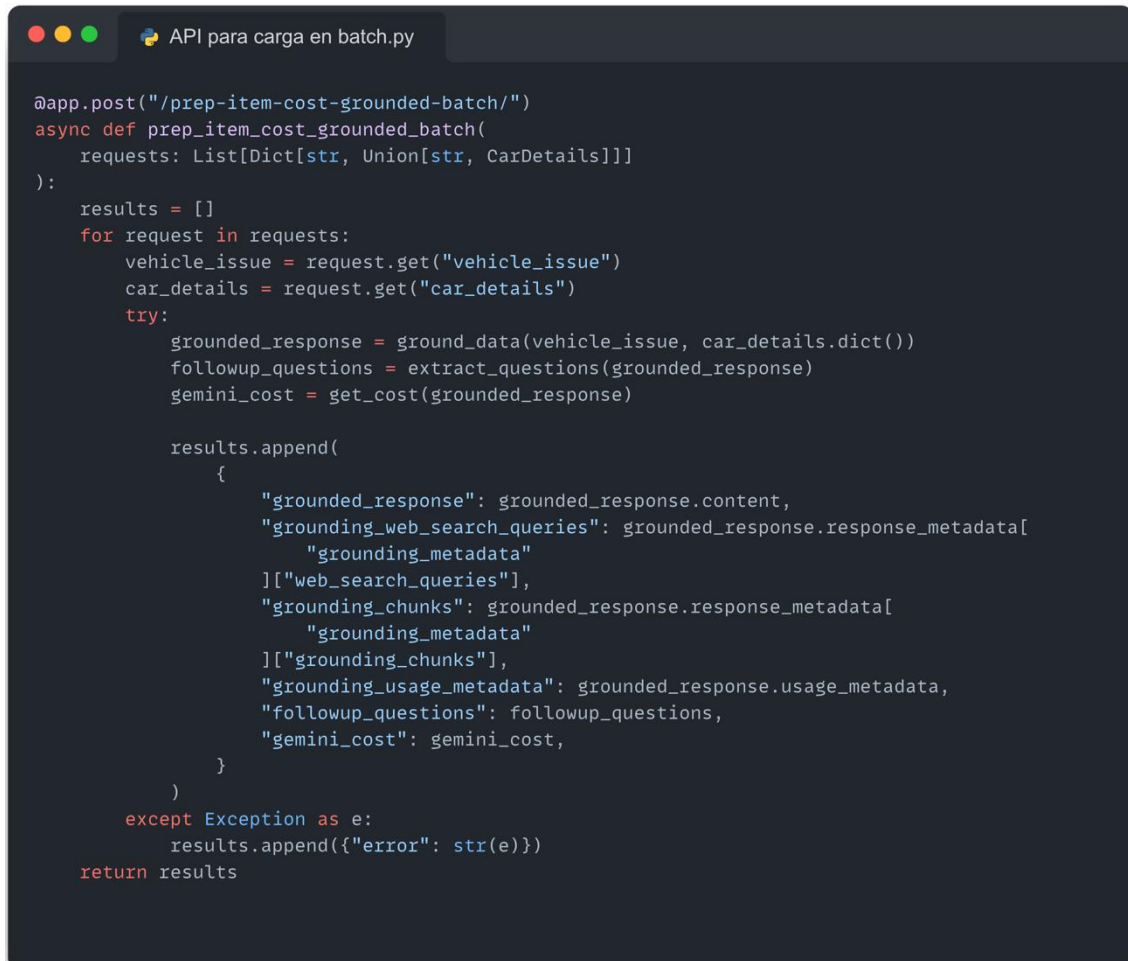
```
def get_cost(response):  
    """  
    Extracts the Total Quote cost from a given response.  
  
    Args:  
        response: The response to analyze.  
  
    Returns:  
        The Total Quote cost.  
    """  
    prompt = f"Given the following response, extract only the Total Quote cost \\  
(just need the number and currency, nothing else). {response}"  
  
    messages = [  
        ("human", prompt),  
    ]  
  
    cost = model.invoke(messages)  
  
    return cost.content
```

El último paso es la puesta en producción de la solución desarrollada. Una vez finalizado el proceso diseñado utilizando funciones en Python que utilizan Gemini, se expusieron dichas funciones a través de interfaces API. En este caso, se eligió FastAPI como el framework para el desarrollo de APIs.

FastAPI es framework web en Python diseñado para crear APIs de alto rendimiento de forma rápida y sencilla. Algunas de sus características principales son las sugerencias de tipos de datos para una mayor velocidad, documentación automática y una depuración más sencilla, lo que lo convierte en una excelente opción para proyectos en los que la velocidad de desarrollo y el rendimiento de las APIs son cruciales. A continuación, se muestra en la figura 35 un ejemplo de cómo se creó la API para la carga en “batch”:

Figura 35

API para carga en batch



```
@app.post("/prep-item-cost-grounded-batch/")
async def prep_item_cost_ground_batch(
    requests: List[Dict[str, Union[str, CarDetails]]]
):
    results = []
    for request in requests:
        vehicle_issue = request.get("vehicle_issue")
        car_details = request.get("car_details")
        try:
            grounded_response = ground_data(vehicle_issue, car_details.dict())
            followup_questions = extract_questions(grounded_response)
            gemini_cost = get_cost(grounded_response)

            results.append(
                {
                    "grounded_response": grounded_response.content,
                    "grounding_web_search_queries": grounded_response.response_metadata[
                        "grounding_metadata"
                    ][
                        "web_search_queries"
                    ],
                    "grounding_chunks": grounded_response.response_metadata[
                        "grounding_metadata"
                    ][
                        "grounding_chunks"
                    ],
                    "grounding_usage_metadata": grounded_response.usage_metadata,
                    "followup_questions": followup_questions,
                    "gemini_cost": gemini_cost,
                }
            )
        except Exception as e:
            results.append({"error": str(e)})
    return results
```

En resumen, se desarrollaron las siguientes APIs:

1. **/find-issues/**: búsqueda de problemas de vehículos según la entrada de texto del usuario.
2. **/prep-item-match/**: encuentra las coincidencias de artículos según los problemas de vehículos y obtiene los costos de los artículos según los “k” elementos similares más importantes.
3. **/prep-item-cost-grounded/**: obtención de los costos de los artículos según los detalles del vehículo que Gemini ha fundamentado (grounding).
4. **/prep-item-cost-grounded-batch/**: ejecución de la función de “grounding” y el cálculo de costos en modo batch.
5. **/prep-item-match-cost/**: encadena las diferentes funciones en un solo flujo.

Para este proyecto, se implementó un servidor en Cloud Run utilizando la siguiente plantilla de Acciones de Github.

Figura 36

YAML utilizado por Github Actions

```
name: Deploy to Cloud Run
on:
  push:
    branches:
      - main
jobs:
  build:
    name: Deploy to Cloud Run
    runs-on: ubuntu-latest
    environment: production
    permissions:
      contents: 'read'
      id-token: 'write'
    steps:
      - name: Checkout code
        uses: actions/checkout@v4

      - name: Authenticate to Google Cloud
        uses: google-github-actions/auth@v2
        with:
          workload_identity_provider: '....'
          service_account: 'github-cloudrun-deploy@abcd.iam.gserviceaccount.com'

      - name: Set up Cloud SDK
        uses: google-github-actions/setup-gcloud@v2

      - name: Configure Docker authentication
        run: |
          gcloud auth configure-docker eu.gcr.io

      - name: Build and push Docker image
        uses: docker/build-push-action@v5
        with:
          context: company/app
          push: true
          tags: eu.gcr.io/company/image

      - name: Deploy to Cloud Run
        id: deploy
        uses: google-github-actions/deploy-cloudrun@v2
        with:
          service: my-fastapi-app
          image: eu.gcr.io/company/image
          region: europe-central2
          env_vars: |
            PROJECT_ID=${{ vars.PROJECT_ID }}
            LOCATION=${{ vars.LOCATION }}
            EMBEDDING_MODEL_ID=${{ vars.EMBEDDING_MODEL_ID }}
            MODEL=${{ vars.MODEL }}
            INDEX_NAME=${{ vars.INDEX_NAME }}
            PROJECT_NUMBER=${{ vars.PROJECT_NUMBER }}

      - name: Use output
        run: 'curl "${{ steps.deploy.outputs.url }}"'
```

Para utilizar esta plantilla, es necesario guardar todas las variables de entorno en un archivo llamado “producción” y se debe otorgar los permisos necesarios a la cuenta de servicio. Esto también se puede replicar localmente en una terminal GCP siguiendo estos pasos desde el directorio de la carpeta de la aplicación:

Figura 37

Configuración para el uso de Cloud Run

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. It contains a series of commands for building and deploying a Docker image to Google Cloud Run.

```
docker build -t gcr.io/company/image .  
  
docker push gcr.io/company/image  
  
gcloud run deploy my-fastapi-app \  
--image gcr.io/company/image \  
--platform managed \  
--region europe-central2 \  
--service-account github-cloudrun-deploy@abc.iam.gserviceaccount.com
```

Finalmente, para poner a prueba la solución y compartir de forma interactiva el resultado del proyecto, se diseñó e implementó una interfaz Streamlit en App Engine. Para implementar la aplicación en App Engine, fue necesario ir con la terminal a la ubicación donde se almacenan los archivos del proyecto (la carpeta de la aplicación), autenticarse y configurar el proyecto.

A partir de las APIs creadas con FastAPI, fue posible utilizar el frontend de Streamlit. La aplicación cuenta con 3 vistas principales:

1. Depuración (Debug): esta vista muestra el desglose paso a paso de las diferentes interfaces de las APIs y las indicaciones que inicializan el proceso (incluidos ejemplos de entrada). El objetivo de esta interfaz es permitir al administrador de la solución, experimentar con los prompts utilizados, buscar oportunidades de mejora, entender los resultados de los diferentes textos de entrada. Esta vista no es accesible para los agentes.

Figura 38

Vista “Debug”

PrepType Match and PrepCost

Backend running on <http://localhost:8000>

Describe the car issue:

Broken Window

See prompt

Find Issues

Issues found: ["Broken Window"]

Describe the car issue to find the top 5 similar items:

Find PrepTypes

Describe the car issue (Grounding):

Vehicle Details

See prompt

See example

Find Price with Grounding

2. Revisión (Review): Herramienta de evaluación de cotizaciones para reparaciones de automóviles. Esta vista incluye una comparación de prueba a ciegas (blind test) entre diferentes cotizaciones y una pestaña de revisión para evaluar las cotizaciones individualmente. El objetivo de esta vista es proporcionar al agente dos cotizaciones (una generada por la solución desarrollada y otra creada previamente por otro agente humano), de las cuales tendrá que elegir la que le parezca más conveniente. El agente deberá elegir -2 si prefiere la cotización creada por un agente, 0 si prefiere cualquiera de las dos y 2 si prefiere la cotización generada por la solución propuesta. De 280 cotizaciones, se obtuvieron 122 valores “0”, 90 valores “-2” y 68 valores “2”. Con esto

se demuestra que las cotizaciones generadas por la solución planteada son similares y/o preferidas por agentes humanos en un 67.8%. Sin embargo, para llegar a resultados concluyentes, es necesario realizar más pruebas con un número mayor de agentes y considerar que cada agente necesita elegir una opción para cada cotización. Al momento de capturar los resultados de esta prueba, cada cotización fue elegida (o valorada) por un solo agente, por lo que no se garantiza que haya un consenso entre todos los agentes para dichas cotizaciones.

Figura 39

Vista “Review”

Blind Test

Review

Blind Test for Car Repair Quotes

Pick the best Car Repair Quote between the 2 provided (Left Option Best = -2, Right Option Best = 2)

Download Dummy CSV

Upload your CSV file

Drag and drop file here

Limit 200MB per file • CSV

Browse files

Issue description: *intermittent near side front door central door lock fault. Exhaust rattles slightly at idle*

See car details

See preitems

Option 1

\$200

Option 2

\$100

Preference

-2

0

2

Submit Preference

Show Scores

	vehicle_details_id	enquiry_id	vehicle_key	electrical_problems_descrig
0	12,328,096	12,331,815	c3fdca8d3983c23b4c536cd62ec9d328	intermittent near side front
1	12,328,096	12,331,815	c3fdca8d3983c23b4c536cd62ec9d328	intermittent near side front
2	12,521,149	12,524,868	5f9d58c4843eaab8ff5627bda8e9b0be	Exhaust rattles slightly at idl
3	10,265,488	10,269,207	f727d96ad2889dc58845ebb12aaa7390	The air recirculation motor i

Download Scores as CSV

3. Vista de Agente (Agent view): Vista principal donde los agentes pueden recorrer las diferentes solicitudes provenientes del sistema, iniciar el flujo LLM y evaluar la propuesta. El objetivo de esta vista es permitir a los agentes obtener la cotización generada por el sistema propuesto ingresando el texto que recibirían desde los mismos usuarios que desean vender sus vehículos. Además, esta vista captura la retroalimentación (feedback), de los agentes con respecto a la cotización. Este feedback es importante para mejorar los prompts a futuro, entender las respuestas y mejorar el flujo.

Figura 40
Vista “Agent”

The screenshot shows a web application titled "PrepType Match and PrepCost" with a car icon. Below the title, it states "Backend running on <http://localhost:8000>". The main heading is "Car Repair Quotes", followed by the instruction "Provide a Car Repair Quote based on the car details and issue provided".

The "Issue description" is: *intermittent near side front door central door lock fault. Exhaust rattles slightly at idle*.

There is a dropdown menu labeled "See car details". Below it is a red-outlined button labeled "Get Issues".

The "Issue" is: *intermittent near side front door central door lock fault*
The "Issue" is: *Exhaust rattles slightly at idle*

Below the issues are four buttons: "Get Prep Items", "Get Similar Prep Items", "Get Quote", and "Submit response".

There is a feedback section: "Was the quote accurate? (Left Option 'No, it wasn't even close' = -2, Right Option 'Yes, it was very accurate' = 2)". It features a horizontal slider with a red line and a red dot at 0. Below the slider is a text input field labeled "What is the quote you could have given for each issue?".

At the bottom, there is a "Show Scores" dropdown menu.

Como parte del proceso de control de calidad, se creó un conjunto de pruebas unitarias utilizando PyTest con datos simulados que cubren funciones adicionales (utils) como la verificación de: si se detectan problemas, la coincidencia de tipos de preparación, la función “grounding” y el cálculo de costos. Las pruebas se han configurado para ejecutarse automáticamente a través de acciones de Github cada vez que se crea o fusiona un nuevo “pull request”.

Para evaluar la calidad de los resultados del sistema, se compararon clasificaciones realizadas manualmente por los agentes a textos ingresados por los usuarios en la plataforma. Se creó una función que permite calcular el porcentaje total de coincidencias perfectas entre los valores reales y la clasificación generada por el sistema. El resultado de esta evaluación es de

87.46%, lo que demuestra que utilizando un LLM es posible clasificar texto en lenguaje natural con diferentes clases (multiclase) con un alto porcentaje de precisión.

3.3 EVALUACIÓN ECONÓMICA

La implementación de la solución propuesta detallada en este trabajo implica un ingreso económico proyectado a fin del primer año luego de la puesta en producción del proyecto en la plataforma digital de la compañía automotriz.

3.3.1 ANÁLISIS DE COSTOS

La implementación del proyecto se desarrolló en un plazo de 2 meses, culminando en el tercer trimestre del 2024. El costo del proyecto considera el pago al personal del equipo interno (ingenieros de la empresa del presente estudio), el personal externo (arquitectos de Google) y el total proyectado de registro de usuarios que quieren vender sus vehículos.

El importe de costo total que se necesita para realizar la implementación del proyecto de automatización de valoración de vehículos fue de £118,000.00, el cual se detalla en la Tabla 13.

Tabla 13

Costo total de implementación del proyecto

Personal	Porcentaje de participación	Sueldo mensual	Sueldo total por 2 meses
Líder del equipo de Machine Learning	100%	£9,000.00	£18,000.00
Ingeniero de datos	100%	£6,000.00	£12,000.00
Ingeniero DevOps	50%	£14,000.00	£14,000.00
Arquitecto senior de soluciones	100%	£25.000,00	£50,000.00

Arquitecto de soluciones – Backup 1	50%	£12,000.00	£12,000.00
Arquitecto de soluciones – Backup 2	50%	£12,000.00	£12,000.00
Costo total			£118,000.00

3.3.2 ANÁLISIS DE BENEFICIOS

La compañía calculó que el ingreso promedio por unidad fue de aproximadamente £310 en enero y que como se muestra en la Figura 41, en promedio ha habido aproximadamente 2,991 ventas mensuales en el último año. Los beneficios generados con la implantación de este proyecto se obtienen de la diferencia de la estimación del margen promedio anual que generan las ventas de autos y el costo de mantener la solución en Google Cloud. Este cálculo se muestra en siguiente ecuación.

Beneficio total por año = [Total de usuarios (autos) que llegan a la etapa de “QA Ready” * Tasa de conversión] - [Costo de Implementación del proyecto + Costo Google Cloud]

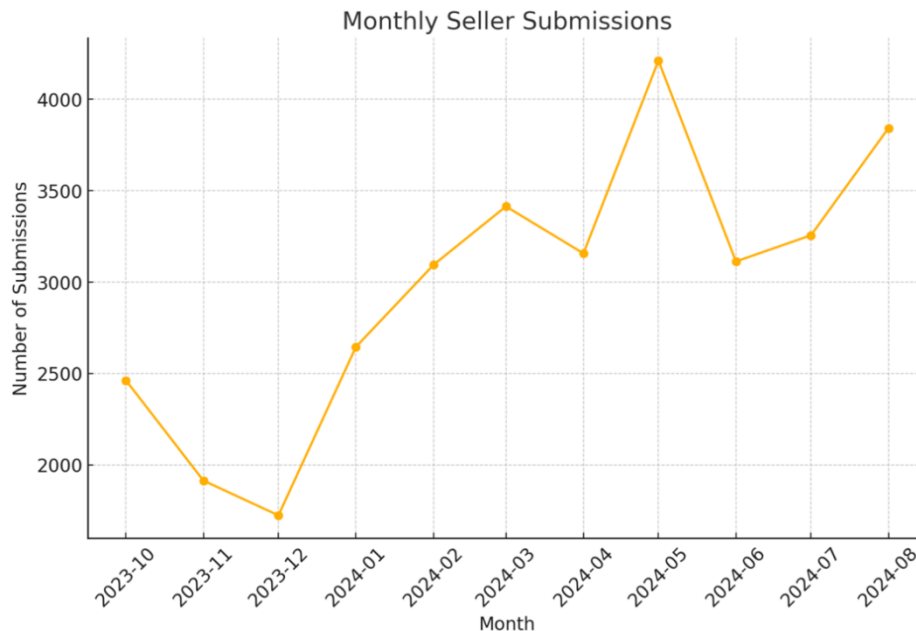
Considerando una tasa de conversión del 70% de autos vendidos y una tasa de cancelación del 20%, se obtienen 1,674 autos. Con ello la empresa estimó generar £51,894.00 adicionales en ingresos mensuales, culminando en un ingreso anual de £622,728.00.

Sin embargo, el tráfico generado durante los primeros meses después de la puesta en producción del proyecto fue mucho menor al esperado. En promedio se lograron vender 128 autos generando un ingreso mensual promedio de £39,680.00 y una proyección anual de £476,160.00.

El costo de mantener la solución en Google Cloud es de £8,589.00 mensualmente. Considerando el costo total (£118,000.00) de implementación del proyecto de la Tabla 9 y el costo en Google Cloud con una proyección a 12 meses (£103,068.00) se obtiene un beneficio anual de £255,092.00.

Figura 41

Ventas mensuales en el último año



Nota. Tomado de la documentación interna del proyecto.

3.3.3 EVALUACIÓN DEL RESULTADO

La relación costo-beneficio de la implementación del proyecto es de aproximadamente 2.15. Esto significa que, por cada libra invertida en el proyecto, la empresa genera un retorno de £2.15, lo cual refleja un nivel de rentabilidad positivo.

La estimación anterior muestra que, a pesar de los costos de infraestructura y desarrollo, la implantación del proyecto generará un retorno económico significativo para la empresa, incrementando sus ingresos anuales de manera notable y fortaleciendo su posición financiera a largo plazo.

3.3.4 INTERPRETACIÓN DEL VAN Y DEL TIR

Del análisis costo-beneficio realizado, se proyecta que la empresa generará un beneficio anual de £255,092.00. El cálculo se basó en el tráfico real después de poner en producción el proyecto desarrollado y considerando un costo anual de mantenimiento en Google Cloud de £103,068.00.

A partir de estos valores, se llevó a cabo el cálculo del VAN (Valor Actual

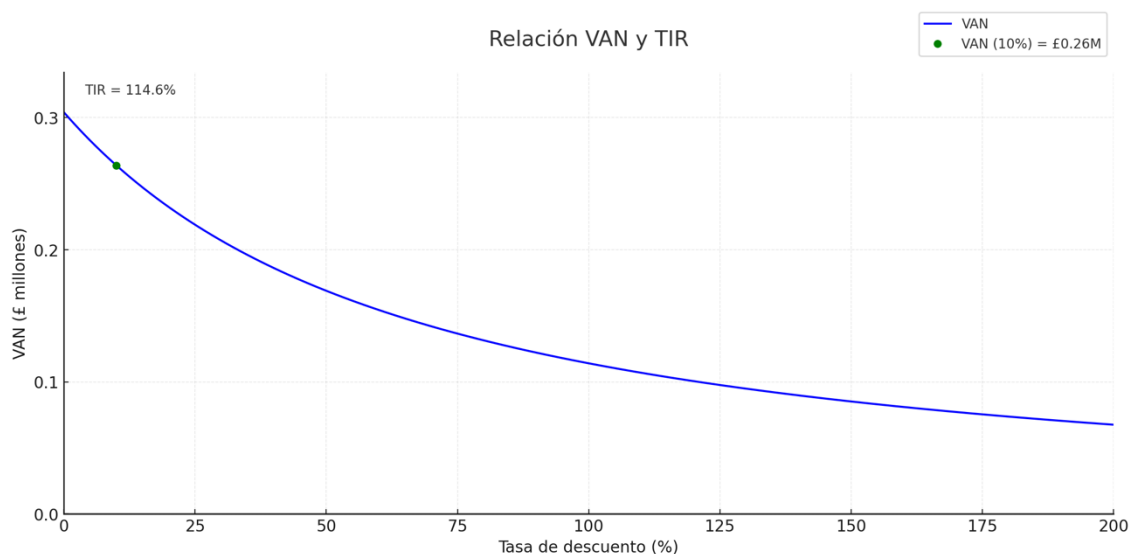
Neto) y la TIR (Tasa Interna de Retorno) tomando en cuenta una tasa de descuento del 10% anual para este análisis y un horizonte de 2 años. Al realizar el cálculo del VAN, se obtuvo un valor de £145,842.00. Esto implica que el proyecto no solo será rentable, sino que generará un valor adicional significativo desde los primeros meses, superando los costos de implementación y los gastos operativos en la nube.

Por otro lado, el cálculo del TIR arrojó un valor de 114.6%, lo que indica que la tasa interna de retorno es superior a la tasa de descuento utilizada en el análisis. Esto sugiere que el proyecto tiene una rentabilidad por encima de lo esperado, mostrando un rendimiento mayor que la inversión inicial y una productividad desde el primer año. En la Figura 42, se puede observar cómo el VAN y la TIR se comportan a lo largo del tiempo, confirmando la viabilidad y el éxito proyectado del plan.

La inversión se recuperaría en aproximadamente 9 meses y 9 días (0.31 meses * 30 días $\approx$ 9 días). Este rápido período de recuperación es consistente con el valor obtenido para la TIR (114.6%) y el VAN positivo (£145,842.00) que calculamos anteriormente, lo que confirma que el proyecto es financieramente atractivo.

Figura 42

Relación VAN y TIR



CAPÍTULO IV - REFLEXIÓN CRÍTICA DE LA EXPERIENCIA

El presente trabajo de Suficiencia Profesional desarrollado por la autora, donde estuvo a cargo de la dirección del proyecto en las etapas de análisis, diseño de arquitectura, desarrollo, pruebas e implementación para el ambiente productivo de la plataforma digital de la empresa automotriz, se convirtió en una oportunidad para afianzar las habilidades de liderazgo y obtener un entendimiento más profundo de las aplicaciones de las últimas tecnologías de inteligencia artificial en un sector poco explorado por la autora. De esta experiencia se destacan los siguientes logros:

- El proyecto de automatización de valoración de vehículos fue el primero en el cual la autora tuvo el liderazgo luego de su proceso de incorporación a la empresa.
- El uso de la metodología Kanban fue muy importante ya que permitió la administración las tareas del proyecto de forma fluida y comunicar objetivamente los avances obtenidos a través de la herramienta MIRO.
- Las reuniones semanales para presentar el progreso obtenido fueron clave para obtener retroalimentación, modificar los prompts y código del proyecto, además de recibir ideas sobre el flujo de negocio y comportamiento de los usuarios.
- El conocimiento obtenido sobre una industria que ha explotado poco las tecnologías más recientes de la inteligencia artificial en el campo de procesamiento de lenguaje.
- Se desarrolló el proyecto con altos estándares de calidad tanto en programación como en comunicación de resultados de forma semanal. Se obtuvo una retroalimentación positiva de la empresa y el compromiso de afianzar la asociación con Google desarrollando otros proyectos.
- Una lección aprendida para el equipo de desarrollo fue la importancia de la comunicación de resultados a los interesados de proyecto que no estuvieron involucrados directamente desde el principio (por ejemplo, el VP de Ingeniería y COO) pero que son sponsors. Comunicarles los

resultados y avances a través de un correo hubiera acelerado la puesta en producción después de la entrega del proyecto.

- Otra lección aprendida fue la revisión de la calidad de datos antes de iniciar el proyecto. Al iniciar la primera etapa, el equipo de desarrollo pudo notar que los datos en las tablas necesarias se encontraban duplicados, así como las categorías de daño vehicular. Esto hizo que se retrasara un poco el desarrollo del código la fase 1, pero a la vez agilizó las siguientes etapas.
- Utilizar instrucciones (prompts) claros y precisos mejora la calidad de respuesta y en general la efectividad del modelo para ejecutar la tarea encomendada. Dividir el proyecto en etapas y en cada etapa crear prompts específicos para cada paso, permitió al modelo enfocarse en una única tarea y cumplirla de forma correcta, reduciendo el número de alucinaciones.

CAPÍTULO V – CONCLUSIONES Y RECOMENDACIONES

5.1 CONCLUSIONES

- Se automatizó y redujo el tiempo de valoración de vehículos, de 24 horas en promedio a minutos, utilizando inteligencia artificial generativa y técnicas avanzadas como RAG.
- La implementación de la metodología Kanban permitió una gestión ágil y eficiente del proyecto, facilitando la coordinación con la empresa y asegurando la entrega continua de valor. El uso de esta metodología ágil demostró ser efectiva para mantener la alineación entre los objetivos del proyecto y las expectativas de los stakeholders.
- La arquitectura diseñada, que integra componentes de IA con sistemas existentes de la empresa, ha demostrado ser robusta y escalable. La selección de tecnologías y herramientas se realizó considerando tanto las necesidades actuales como el potencial de crecimiento futuro del sistema.
- La implementación de técnicas de IA Generativa para el análisis de texto ha permitido automatizar significativamente el proceso de interpretación de las descripciones de vehículos, reduciendo el tiempo de procesamiento y mejorando la consistencia en las evaluaciones. La combinación de técnicas tradicionales de procesamiento de lenguaje natural como los embeddings de texto y LLMs nos permite aprovechar el poder de la inteligencia artificial para diversos casos de uso.
- El marco de evaluación desarrollado ha permitido validar la precisión del sistema, demostrando una mejora significativa en los tiempos de valoración sin comprometer la exactitud de las valoraciones. Los resultados muestran una reducción promedio del tiempo de valoración del 200%, manteniendo un nivel de precisión del 87% en comparación con las valoraciones realizadas por agentes. Estos valores se obtuvieron después de comparar el texto en lenguaje natural clasificado manualmente por los agentes previamente y la clasificación que retorna la solución desarrollada en este trabajo.

5.2 RECOMENDACIONES

- Implementar una estrategia para capturar los resultados de la ingeniería de instrucciones.
- Implementar procesos de limpieza de datos regularmente aprovechando modelos de embeddings de texto y LLMs para completar datos donde sea necesario y/o corregir las categorías. Las tablas almacenadas en BigQuery permiten integración nativa con modelos de lenguaje, lo que haría fácil la adopción de estos para procesamiento de datos.
- Desarrollar un marco de evaluación para modelos grandes de lenguaje que consideren las reglas y flujos del negocio. Evaluar los resultados generados por el modelo es de suma importancia para escalar rápidamente este tipo de soluciones en otras áreas de negocio.
- Trasladar la solución técnica y las buenas prácticas aprendidas y realizadas en el proyecto al resto de proyectos que utilicen inteligencia artificial generativa.
- Para reducir la latencia se recomienda utilizar la característica de caching de los modelos grandes de lenguaje. Almacenar de forma temporal todo el contexto (en este caso todas las categorías de daño vehicular) podría aportar beneficios en el tiempo de respuesta, especialmente si se decide llevar este proyecto a millones de usuarios.

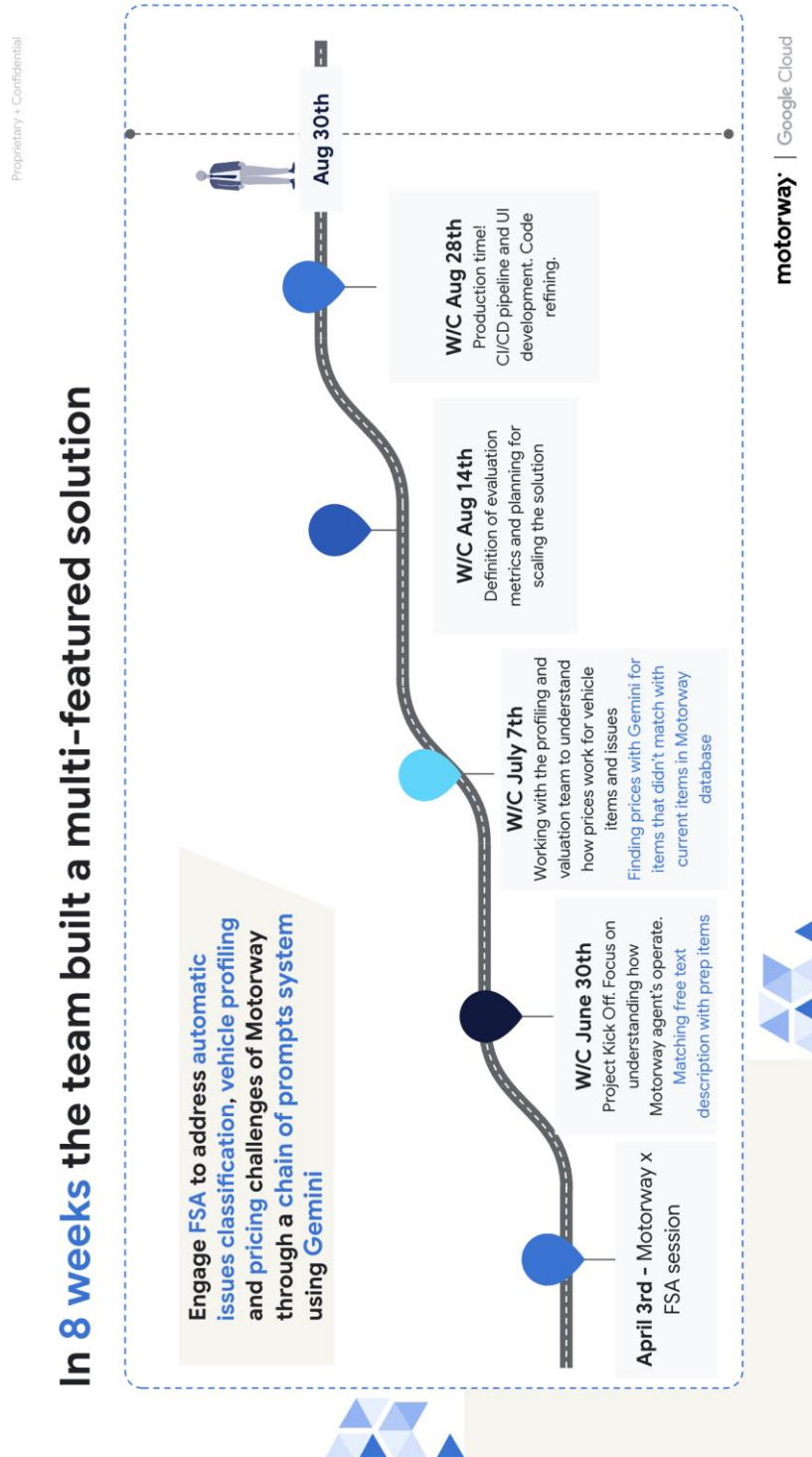
5.3 FUENTES DE INFORMACIÓN

- BigQuery overview. (s.f.). Google Cloud. Recuperado de <https://cloud.google.com/bigquery/docs/introduction>
- Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., ... Amodei, D. (2019). Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33. https://proceedings.neurips.cc/paper_files/paper/2019/file/dc6a7e655d7e5840e66733e9ee67cc69-Paper.pdf
- Chang, Y., Wang, X., Wang, J., Wu, Y., Zhu, K., Chen, H., Yang, L., Yi, X., Wang, C., Wang, Y., Ye, W., Zhang, Y., Chang, Y., Yu, P. S., Yang, Q., y Xie, X. (2023). A survey on evaluation of large language models. *arXiv*. <https://arxiv.org/pdf/2307.03109>
- Chen, H., Jiao, F., Li, X., Qin, C., Ravaut, M., Zhao, R., Xiong, C., y Joty, S. (2023). ChatGPT's one-year anniversary: Are open-source large language models catching up?. *arXiv*. <https://arxiv.org/pdf/2311.16989>
- Embeddings in the Gemini API. (s.f.). Google Cloud. Recuperado de <https://ai.google.dev/gemini-api/docs/embeddings>
- Ge, T., Hu, J., Wang, L., Wang, X., Chen, S., y Wei, F. (2023). In-context autoencoder for context compression in a large language model. *arXiv*. <https://arxiv.org/pdf/2307.06945>
- Han, Y., Liu, C., y Wang, P. (2023). A comprehensive survey on vector database: Storage and retrieval technique, challenge. *arXiv*. <https://arxiv.org/pdf/2310.11703>
- Introduction to Large Language Models. (s.f.). Google Cloud. Recuperado de <https://developers.google.com/machine-learning/resources/intro-llms>
- Introduction to Vertex AI. (s.f.). Google Cloud. Recuperado de <https://cloud.google.com/vertex-ai/docs/start/introduction-unified-platform>
- Kazemnejad, A., Padhi, I., Natesan Ramamurthy, K., Das, P., & Reddy, S. (2023). The impact of positional encoding on length generalization in transformers. *arXiv*. <https://doi.org/10.48550/arXiv.2305.19466>
- Langrené, N., Zhu, S., Chen, B., y Zhang, Z. (2023). Unleashing the potential of prompt engineering in large language models: A comprehensive review. *arXiv*. <https://arxiv.org/pdf/2310.14735>
- Laoyan, S. (2024, 2 de febrero). What is Agile methodology? (A beginner's guide). Asana. Recuperado de <https://asana.com/resources/agile-methodology>

- Naveed, H., Khan, A. U., Qiu, S., Saqib, M., Anwar, S., Usman, M., Akhtar, N., Barnes, N., & Mian, A. (2023). A comprehensive overview of large language models. arXiv. <https://doi.org/10.48550/arXiv.2307.06435>
- Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., y Sutskever, I. (2019). Language models are unsupervised multitask learners. OpenAI. https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf
- Schwaber-Cohen, R. (s.f.). What is a Vector Database & How Does it Work? Use Cases + Examples. Pinecone. <https://www.pinecone.io/learn/vector-database/>
- Shi, L., Zhang, H., Yao, Y., Li, Z., & Zhao, H. (2024). Keep the cost down: A review on methods to optimize LLM's KV-cache consumption. arXiv. <https://arxiv.org/pdf/2407.18003>
- Törnberg, P. (2023, July 24). How to use LLMs for Text Analysis. arXiv.org. <https://arxiv.org/abs/2307.13106>
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., y Polosukhin, I. (2017). Attention is all you need. Advances in Neural Information Processing Systems, 30. https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf
- Vilar, R., Gomez-Rodriguez, M., y Gomez, V. (2023). ChatGPT re-writes history: Hallucinated political events in open-domain dialogue models. arXiv. <https://arxiv.org/pdf/2307.06435>
- Wang, L., Yang, N., Huang, X., Jiao, B., Yang, L., Jiang, D., Majumder, R., y Wei, F. (2022). Text embeddings by weakly supervised contrastive pre-training. arXiv. <https://arxiv.org/pdf/2212.03533>
- Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., Chi, E. H., Hashimoto, T., Vinyals, O., Liang, P., Dean, J., y Fedus, W. (2022). Emergent abilities of large language models. arXiv. <https://arxiv.org/pdf/2206.07682>
- What is Cloud Run. (s.f.). Google Cloud. Recuperado de <https://cloud.google.com/run/docs/overview/what-is-cloud-run>

ANEXOS

Anexo 1: Cronograma a alto nivel



Anexo 2: Requerimiento creado en Salesforce para iniciar el proyecto

Q Search...

Q Search...

Sales

Home

Cloud Connect

Accounts

Opportunities

Expert Requests

CloudHub

Quotes

Leads

Tasks

Contacts

Campaigns

Dashboards

Reports

Cases

More

Star

Plus

Help

Refresh

More

Expert Request

ER-290525

Follow

Transfer/Edit Request

Share Request

Assign To Me

Edit

New Note

Stalled/Paused

Rejected

Completed

Mark Status as Complete

Details

Tasks & Docs

Expert Request Details

Account

MOTORWAY ONLINE LTD

Opportunity

Motorway - Profiling Journey - Seller Experience \$1.05m #GenAI (FSA)

Expert Team

EMEA AI FSA

Priority

P3

Engagement Type

GenAI POC/MVP Customer Co-Build

Solution / Products

AI/ML

Additional Details

https://docs.google.com/document/d/1hRQx03MZzCwp3OV-JJ7D702LXbSx7zTgxcv3FY/edit?resourcekey=0-lwLE6E_LBCXFXbJ2kjzf_w&tab=L0

Workflow

Owner

Jenny Vega

Request Start Date

7/31/2024

Request End Date

Status

Completed

Priority

P3

Rejection Reason

Rejection Reason Additional Details

Shared With

Activity

Weekly Status

New Task

Create a task...

Add

Filters: All time • All activities • All types

Refresh

Expand All

View All

Upcoming & Overdue

No activities to show.

Get started by sending an email, scheduling a task, and more.

No past activity. Past meetings and tasks marked as done show up here.

Chatter

Post

Question

Poll

Notes

History

Create Cloud Helpdesk Case

Helpdesk Chat

To Do List