**Título**  
**DISTRIBUTED SYSTEM FOR AUTOMATED VEHICLE IDENTIFICATION USING GENERATIVE ARTIFICIAL INTELLIGENCE IN MOBILE APPLICATIONS**

**Autores**  
Yoshua Calahorrano – aycalahorrano@uce.edu.ec  
John Guerra – jaguerrac@uce.edu.ec  
Universidad Central del Ecuador – Carrera de ingeniería en informática.

**Resumen**   
Este artículo presenta el desarrollo de una solución tecnológica para reconocer vehículos de forma automática a partir de imágenes tomadas con dispositivos móviles. La propuesta aprovecha la inteligencia artificial generativa a través de la API Gemini 2.0 Flash de Google, capaz de analizar imágenes y extraer datos relevantes como marca, modelo, año, precio estimado y una reseña breve. El sistema se construye bajo una arquitectura distribuida basada en microservicios, utilizando tecnologías como FastAPI para el backend, Flutter para la aplicación móvil, NGINX como proxy y una base de datos MySQL para el almacenamiento. Toda la infraestructura se orquesta mediante Docker Compose, garantizando escalabilidad y mantenimiento eficiente. A lo largo del desarrollo se empleó la metodología CRISP-DM para organizar las fases del proyecto. Las pruebas demostraron una alta precisión en el reconocimiento, tiempos de respuesta óptimos y una experiencia fluida para el usuario final. Esta herramienta representa un avance significativo para la gestión vehicular y tiene potencial de uso en contextos como concesionarios y movilidad urbana en el Ecuador.

**Palabras clave:** Inteligencia artificial generativa, Gemini, reconocimiento de vehículos, microservicios, Flutter, minería de datos, CRISP-DM.

**Abstract**  
This article introduces the development of a distributed system designed to automatically identify vehicles from images captured through mobile devices. The system leverages generative artificial intelligence using Google’s Gemini 2.0 Flash API, which analyzes images and extracts key information such as the vehicle’s make, model, year, estimated price, and brief description. The backend is built with FastAPI, while a Flutter-based mobile app allows users to interact with the platform. A MySQL database stores the processed data, and NGINX handles routing and access through microservices architecture orchestrated via Docker Compose. The CRISP-DM methodology was used to structure the project’s data lifecycle. Results show high accuracy in detection, fast response times, and a smooth user experience. This work demonstrates a practical and scalable application of AI in Ecuador’s mobility and automotive sector.

**Keywords:** Generative artificial intelligence, Gemini, vehicle recognition, microservices, Flutter, data mining, CRISP-DM.

1. **Introducción**  
   En la actualidad, el reconocimiento visual automatizado representa una de las aplicaciones más relevantes de la inteligencia artificial en contextos cotidianos. La identificación de vehículos a partir de imágenes no solo tiene valor comercial, sino también logístico, administrativo y urbano lo que este sentido, el presente trabajo propone el desarrollo de un sistema móvil e inteligente capaz de analizar imágenes capturadas por el usuario para extraer información específica de un vehículo: marca, modelo, año, precio estimado y una centrada y breve reseña técnica.

Esta solución se apoya en la API Gemini 2.0 Flash de Google, una de las tecnologías más avanzadas de IA generativa, permitiendo resultados en tiempo real con gran nivel de detalle. El sistema está desarrollado sobre una arquitectura de microservicios, con un backend construido en FastAPI, un frontend móvil desarrollado con Flutter y un entorno de despliegue gestionado con Docker y NGINX. Además, se adoptó la metodología CRISP-DM para estructurar las etapas del proyecto desde una perspectiva de minería de datos.

A través de esta investigación se busca demostrar que es posible implementar una herramienta distribuida, precisa y eficiente para la identificación vehicular, adaptable al mercado ecuatoriano. Esta tecnología tiene potencial aplicación en concesionarios, verificaciones vehiculares, o incluso plataformas de movilidad.

1. **Materiales y métodos**  
   El desarrollo de esta solución tecnológica se estructuró utilizando la metodología CRISP-DM, que permite una organización clara de las fases de un proyecto basado en datos: comprensión del negocio, comprensión de los datos, preparación, modelado, evaluación e implementación.

Herramientas tecnológicas

**IA Generativa**: API Gemini 2.0 Flash (Google) para el análisis de imágenes mediante prompts establecidos en backend.

**Backend**: FastAPI en Python, estructurado bajo microservicios independientes.

**Frontend móvil**: Flutter, facilitando una experiencia fluida y multiplataforma.

**Base de datos**: MySQL 5.7 para almacenar los resultados analizados por Gemini.

**Infraestructura**: Orquestación con Docker Compose y NGINX como proxy reverso.

Metodología de implementación

Se desarrollaron tres microservicios principales: verificación (verify\_phase), análisis (analyze\_phase) y almacenamiento (logger).

El usuario, desde la aplicación, selecciona o captura una imagen.

La imagen es enviada al servidor y procesada mediante Gemini, retornando un JSON con los datos del vehículo.

Esta información es visualizada en tiempo real y almacenada en la base de datos MySQL.

Validación y pruebas

Se realizaron pruebas unitarias, de integración, y de carga.

Se midió la precisión del modelo y el tiempo de respuesta en distintos escenarios.

Todo el sistema fue probado en entornos controlados utilizando imágenes de diferentes tipos de vehículos, condiciones de luz, calidad de imagen y cantidad.

1. **Resultados**  
   A partir de las pruebas realizadas, se obtuvieron resultados consistentes respecto a la precisión del sistema. En promedio, el modelo generativo Gemini logró una precisión superior al 90% en la identificación correcta de vehículos cuando las imágenes cumplían con condiciones óptimas (buena iluminación, enfoque frontal).

**Métricas clave:**

Se observaron mejoras significativas al estructurar prompts precisos en la fase de analyze\_phase, lo cual aumentó la calidad del texto generado por Gemini. Además, la interfaz móvil demostró ser intuitiva y estable, permitiendo al usuario interactuar fácilmente con el sistema.

También se implementó un módulo de historial, donde los usuarios pueden visualizar vehículos analizados anteriormente con formato visual tipo tarjetas.

1. **Discusión**  
   La implementación del sistema propuesto reveló aspectos clave sobre la aplicabilidad de la inteligencia artificial generativa en contextos reales, especialmente en entornos móviles y distribuidos. Una de las principales fortalezas observadas fue la precisión de Gemini 2.0 Flash para identificar vehículos, incluso cuando las imágenes presentaban ligeras variaciones de iluminación o ángulo.

A diferencia de modelos tradicionales como YOLO, que requieren entrenamiento intensivo y preetiquetado de datos, el uso de un modelo LLM preentrenado permitió reducir significativamente los tiempos de desarrollo e incrementar la capacidad de adaptación del sistema. No obstante, se identificaron limitaciones: imágenes con múltiples vehículos o baja calidad afectaron la precisión del análisis.

Otro aspecto destacado fue la arquitectura modular basada en microservicios. Esta permitió escalar componentes por separado, realizar pruebas aisladas y facilitar futuras integraciones, como consultas al historial o servicios externos como validación vehicular. El uso de Docker Compose y NGINX contribuyó a la estabilidad y al balance de cargas, reduciendo la complejidad del despliegue.

Finalmente, la experiencia del usuario en la aplicación móvil fue intuitiva y funcional. El historial tipo tarjetas, el tiempo de respuesta inferior a cinco segundos, y la claridad de los datos generados aportaron valor al diseño final.

1. **Conclusiones**

* El sistema desarrollado demuestra que es viable integrar inteligencia artificial generativa, arquitectura distribuida y aplicaciones móviles para resolver problemas reales de identificación vehicular. A través del uso de Gemini 2.0 Flash, fue posible automatizar la extracción de características clave de vehículos a partir de imágenes, logrando altos niveles de precisión y tiempos de respuesta óptimos.
* La arquitectura modular facilita el mantenimiento, escalado e integración futura del sistema. La metodología CRISP-DM permitió estructurar el proyecto con claridad desde la comprensión del negocio hasta la implementación.
* Este sistema tiene un alto potencial de uso en contextos como concesionarios de autos, verificaciones técnicas, catastros urbanos o plataformas de movilidad. Además, sienta las bases para futuras ampliaciones como la integración de datos legales, predicción de valores de reventa o conexión con sistemas de tránsito.

1. **Referencias bibliográficas**  
   [1] Google AI, “Gemini API Documentation,” 2024. [Online]. Available: <https://ai.google.dev>  
   [2] S. Raschka, Y. Liu, and V. Mirjalili, *Machine Learning with PyTorch and Scikit-Learn*, Packt Publishing, 2022.  
   [3] Docker Inc., “Docker Documentation,” [Online]. Available: https://docs.docker.com  
   [4] FastAPI Project, “FastAPI Documentation,” [Online]. Available: https://fastapi.tiangolo.com  
   [5] CRISP-DM Consortium, “CRISP-DM 1.0: Step-by-step data mining guide,” 2000.  
   [6] J. Guerra and Y. Calahorrano, “Aplicación móvil para detección de vehículos con IA generativa,” Universidad Central del Ecuador, 2024.

|  |  |  |
| --- | --- | --- |
| Microservicio | Función Principal | Tecnología |
| verify-phase-service | Verificar si la imagen contiene un vehículo | Python + FastAPI |
| analyze-phase-service | Extraer información del vehículo (marca, modelo, etc.) usando Gemini | Python + FastAPI + Gemini API |
| logger-service | Almacenar los datos procesados en la base de datos MySQL | Python + FastAPI + MySQL |
| unifier-backend | Orquestar la lógica entre verify, analyze y logger | Python + FastAPI + Requests |
| nginx-proxy | Redirigir las peticiones desde frontend a backend | NGINX |
| flutter-app | Captura de imagen, visualización de resultados e historial | Flutter |

|  |  |
| --- | --- |
| Métrica | Resultado |
| Precisión de identificación | 91.6% |
| Tiempo medio de respuesta | 3.84 segundos |
| Tasa de éxito de análisis | 96.2% |
| Errores de almacenamiento | 0% (sin duplicidad ni pérdidas) |

**Tablas**  
Tabla de función de microservicios  
  
Tabla de resultados

**Figuras**  
figura 1  
Diagrama de flujo del sistema  
Diagrama

El contenido generado por IA puede ser incorrecto.  
  
figura 2  
Captura de aplicación en funcionamiento  
Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

**Ecuaciones**

**Función de inferencia de un modelo generativo autoregresivo**

Donde:

* : representación de la imagen extraída por el encoder visual
* : token generado en el tiempo
* : parámetros del modelo preentrenado

Muestra que Gemini funciona como un LLM multimodal: entrada visual → salida textual.

**Atención cruzada (cross-attention)**

En modelos como Gemini, se usa atención entre la imagen codificada y los tokens generados:

Esta fórmula puede incluirse el modelo emplea mecanismos de atención entre las capas visuales (imagen) y lingüísticas (tokens).

**Métrica de precisión (opcional, si calculaste)**

Si mide el rendimiento con precisión simple: