PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

FACULTAD DE INGENIERÍA

CARRERA DE: INGENIERÍA EN SISTEMAS DE INFORMACIÓN

Trabajo de Titulación

Tema: Diseño y Desarrollo de un Prototipo de Plataforma Web de Ciencia de Datos para el Estudio del Sentimiento Ciudadano sobre el Transporte Público en Quito durante el Año 2024 a partir de Publicaciones en la Red Social X

AUTOR:

MATEW JAIR POZO POZO

QUITO DM, JULIO 2025

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1: INTRODUCCIÓN

1.1 JUSTIFICACIÓN

Las ciudades modernas enfrentan retos significativos en la gestión del transporte público, especialmente cuando se trata de entender cómo los ciudadanos perciben su funcionamiento. Quito, como capital del Ecuador, ha experimentado transformaciones importantes en movilidad, incluida la incorporación del Metro y ajustes en rutas de buses y sistemas de pago. Sin embargo, a pesar de estos avances, no siempre existe claridad de cómo estos cambios son recibidos por la ciudadanía, las redes sociales, y en particular la plataforma X (anteriormente Twitter), se han convertido en canales clave donde los usuarios expresan de forma espontánea sus opiniones, quejas o experiencias sobre el transporte. Este flujo de información, aunque es abundante, se encuentra en formatos no estructurados que dificultan su análisis sin herramientas especializadas.

En este contexto, el presente proyecto propone el diseño y desarrollo de un prototipo de plataforma web de ciencia de datos que permita analizar el sentimiento ciudadano sobre el transporte público en Quito durante el año 2024, a partir de publicaciones extraídas de la red social X. Esta herramienta está dirigida a servir como modelo académico funcional que demuestra cómo la ciencia de datos puede aplicarse en el monitoreo de la percepción ciudadana, siendo potencialmente útil para autoridades municipales, entidades de movilidad y desarrolladores interesados en soluciones urbanas.

El sistema será evaluado en función de su capacidad para recolectar una muestra de publicaciones, aplicar correctamente análisis de sentimiento expresados en idioma español y presentar visualizaciones interactivas que reflejen el estado de opinión en distintos momentos del año, esta herramienta tendrá un valor potencial para actores involucrados en la toma de decisiones en temas de movilidad, además de servir como un ejemplo práctico del uso de ciencia de datos aplicada al análisis social urbano.

1.2 PLANTEAMIENTO DEL PROBLEMA

La gestión eficiente del transporte público es uno de los principales retos urbanos, de manera particular en ciudades latinoamericanas como la capital ecuatoriana. Si bien, en nuestro caso, se han realizado inversiones significativas en infraestructura como la implementación del Metro de Quito, las autoridades aún enfrentan dificultades para conocer de forma precisa y oportuna cómo perciben los ciudadanos estos servicios.

En la actualidad, gran parte de la opinión pública sobre el transporte se expresa en redes sociales, especialmente en la plataforma X, donde los usuarios comparten experiencias, quejas, sugerencias o reacciones ante cambios operativos, demoras, tarifas o condiciones del servicio. Sin embargo, la información disponible en estos medios es masiva, dispersa y no estructurada, dificultando su análisis y evaluación al no contarse con herramientas especializadas. Esta situación limita la posibilidad de obtener una visión clara del sentimiento ciudadano sobre el sistema de transporte, lo que a su vez afecta la toma de decisiones basadas en evidencia por parte de autoridades, planificadores o investigadores.

Aunque existen plataformas comerciales que permiten análisis avanzados de redes sociales, su alto costo las vuelve inaccesibles para muchas instituciones públicas, investigadores o desarrolladores independientes. Por ello, surge la necesidad de contar con una herramienta académica accesible que permita recolectar y analizar publicaciones en redes sociales enfocadas exclusivamente en el transporte público en Quito.

Esta herramienta estará orientada a generar visualizaciones interpretables del sentimiento ciudadano, clasificando publicaciones como positivas, negativas o neutras, e identificando los temas que con mayor frecuencia se presentaron a lo largo del año 2024. El desarrollo de este prototipo busca facilitar investigaciones futuras sobre percepción ciudadana en movilidad urbana y servir como base técnica para análisis posteriores en el ámbito académico, institucional o técnico.

1.3 OBJETIVOS GENERALES Y ESPECÍFICOS

1.3.1 Objetivo General

Diseñar y desarrollar un prototipo de plataforma web de ciencia de datos que permita analizar el sentimiento ciudadano sobre el transporte público en la ciudad de Quito durante el año 2024, utilizando publicaciones extraídas de la red social X.

1.3.2 Objetivos Específicos

- Recolectar, analizar y limpiar una muestra de al menos 2.500 publicaciones divulgadas en idioma español, relacionadas con el transporte público en Quito durante el año 2024, utilizando la API de la red social X y filtros por palabras clave.

- Aplicar modelos de procesamiento de lenguaje natural (NLP) para clasificar el sentimiento de las publicaciones en tres categorías: positivo, negativo y neutro; y validar la precisión del modelo mediante comparación con una muestra manual de 100 publicaciones etiquetadas.

- Diseñar y desarrollar una plataforma web funcional que integre los procesos de carga, análisis y visualización de datos de forma automatizada.

- Implementar al menos tres dashboards interactivos que permitan visualizar los resultados de los análisis segmentados por mes, tipo de transporte (metro, bus, ecovía) y polaridad del sentimiento.

- Evaluar el desempeño del prototipo mediante tres tipos de pruebas: (1) precisión del modelo de clasificación de sentimiento, (2) validación funcional de los módulos de carga, análisis y visualización, y (3) análisis de claridad y utilidad de los dashboards generados.

1.4 PROCEDIMIENTO - MARCO METODOLÓGICO

1.4.1 Tipo de Investigación

El presente proyecto se enmarca dentro de una investigación aplicada de tipo tecnológico, ya que se enfoca en el desarrollo de un prototipo funcional con utilidad práctica, basado en conceptos y técnicas propias de la ciencia de datos y del procesamiento de lenguaje natural. A

su vez, presenta un enfoque experimental, debido a que se propone construir un sistema que será validado mediante pruebas funcionales y análisis de resultados. El objeto de estudio es el sentimiento ciudadano sobre el transporte público en Quito, expresado a través de publicaciones en la plataforma X durante el año 2024.

1.4.2 Justificación del Uso de CRISP-DM

La elección de CRISP-DM como metodología se justifica por su enfoque específico en procesos de análisis de datos, lo cual se ajusta de la mejor manera al tipo de proyecto que se plantea. A diferencia de metodologías de desarrollo de software como Scrum, CRISP-DM está diseñada para orientar todo el proceso desde la comprensión del problema hasta la entrega de resultados procesables, además, su estructura cíclica permite iterar entre fases si se detectan mejoras o cambios necesarios en los datos o resultados.

1.4.3 Ventajas del Enfoque Adoptado

El enfoque metodológico de CRISP-DM aporta varias ventajas clave al desarrollo del presente proyecto, al tratarse de una metodología diseñada específicamente para procesos de minería y análisis de datos (Han, Pei, & Kamber, 2011). lo cual resulta más adecuado que aplicar enfoques ágiles de software puro. En lugar de solo programar rápido, permite organizar las etapas en torno a la comprensión, la preparación, el modelado y la evaluación de los datos, que son precisamente las fases clave en este trabajo.

Una de sus mayores fortalezas es la flexibilidad: se puede volver a fases previas si se detectan problemas. Por ejemplo, si al evaluar el modelo de análisis de sentimiento se ve que la precisión no es suficiente, se puede regresar a la etapa de preparación para ajustar el preprocesamiento. Esa posibilidad de retroalimentación es muy útil al trabajar con datos reales y poco estructurados, como textos de redes sociales.

CRISP-DM también ayuda a mantener claro el ciclo completo del análisis, desde entender el problema hasta lograr un sistema que entregue resultados útiles. Esto facilita no solo tener objetivos bien definidos desde el inicio, sino también documentar todo el proceso de forma técnica y ordenada, algo esencial para una tesis.

Además, al ser una metodología reconocida internacionalmente, brinda una base sólida y validada para estructurar el desarrollo. Esto mejora la forma en que se puede justificar el trabajo y defenderlo ante el tribunal académico.

1.5 ALCANCE

El presente proyecto contempla el diseño y desarrollo de un prototipo de plataforma web de ciencia de datos enfocada en el análisis de sentimiento ciudadano sobre el transporte público en la ciudad de Quito. El análisis se basará exclusivamente en publicaciones realizadas en la red social X, durante el periodo comprendido entre el 1 de enero y el 31 de diciembre del año 2024.

Este sistema estará limitado al procesamiento de textos en idioma español, utilizando técnicas de procesamiento de lenguaje natural (NLP) para la clasificación de publicaciones en tres categorías de sentimiento: positivo, negativo y neutro. La recolección de datos se realizará a través de filtros por palabras clave y/o hashtags relacionados con el transporte público en Quito, sin considerar publicaciones georreferenciadas ni datos privados.

El prototipo incluirá funcionalidades específicas como la carga de datos, procesamiento automatizado, clasificación de sentimiento y visualización de resultados mediante dashboards interactivos, que permitirán observar la evolución del sentimiento ciudadano en el tiempo, identificar los temas más frecuentes y presentar resúmenes gráficos por período o tipo de transporte.

No se contempla la integración con múltiples redes sociales, monitoreo en tiempo real, ni la implementación de modelos predictivos o funcionalidades avanzadas de inteligencia artificial. Tampoco se utilizarán datos sensibles ni se incluirá análisis demográfico, ya que el enfoque se centra únicamente en el contenido textual disponible públicamente.

La evaluación del prototipo se realizará a través de tres tipos de pruebas. En primer lugar, se medirá la precisión del modelo de análisis de sentimiento utilizando una muestra de publicaciones que serán clasificadas manualmente, las que luego serán comparadas con los resultados obtenidos por el sistema. En segundo lugar, se evaluará el funcionamiento general de la plataforma verificando que cada módulo de recolección de datos, análisis y visualización cumpla correctamente con su propósito.

Por último, se revisará la utilidad de las visualizaciones generadas, tomando en cuenta si los gráficos son claros, comprensibles y muestran bien la información que se desea analizar. Estas pruebas permitirán comprobar que el prototipo cumple con sus objetivos y que puede ser útil como herramienta académica para estudiar la percepción ciudadana sobre el servicio de transporte público en Quito.

El proyecto se considerará finalizado cuando se haya desarrollado un prototipo funcional que cumpla con los objetivos planteados, incluyendo la recolección, análisis y visualización de datos sobre el sentimiento ciudadano. La entrega final consistirá en el sistema web operativo y el correspondiente documento de titulación que describa su diseño, implementación, validación y resultados. No se contempla la evolución del prototipo más allá del ámbito académico definido en este plan.

CAPÍTULO 2: MARCO TEÓRICO Y CONCEPTUAL

2.1 Antecedentes

El análisis de datos provenientes de redes sociales ha ganado mucha relevancia en los últimos años, tanto en el ámbito académico como en el empresarial. Hoy en día existen herramientas comerciales como Brandwatch, Talkwalker o Sprout Social que ofrecen monitoreo social avanzado usando algoritmos de inteligencia artificial. Sin embargo, suelen ser costosas y poco accesibles para estudiantes, pequeños negocios o instituciones educativas.

En el campo académico, por ejemplo, autores como Russell (2013) y Bird et al. (2009) han trabajado el tema del procesamiento de lenguaje natural aplicado a la minería de texto en plataformas como Twitter. También se han desarrollado tesis y prototipos en universidades de América Latina orientados a la extracción y visualización de tendencias, con enfoques en política, marketing o análisis social.

Este proyecto busca alinearse con esos trabajos previos, pero propone un enfoque más práctico y educativo, usando herramientas de código abierto para construir un prototipo funcional. La idea es facilitar el análisis de publicaciones tomadas de redes sociales, aplicando modelos de NLP e inteligencia artificial junto con visualizaciones interactivas.

Se eligió la red social X (antes conocida como Twitter) como fuente de datos por su naturaleza pública, el formato breve de sus publicaciones y su relevancia en conversaciones sobre temas ciudadanos. A diferencia de otras plataformas, X permite filtrar las publicaciones por tema, idioma o rango de fechas, lo que la hace muy adecuada para recolectar percepciones espontáneas sobre el transporte urbano en Quito.

2.2 Metodología

2.2.1 CRISP DM

Para la planificación, organización y ejecución del proyecto se adoptará la metodología CRISP-DM (Cross-Industry Standard Process for Data Mining), ampliamente utilizada en entornos de ciencia de datos. Esta metodología proporciona una guía estructurada, flexible y cíclica para el

desarrollo de proyectos de análisis de datos, desde la definición del problema hasta la entrega del resultado final.

CRISP-DM contempla seis fases principales:

Comprensión del Negocio

En esta fase se establece el propósito del sistema, analizar el sentimiento ciudadano sobre el transporte público en Quito, con el objetivo de obtener una visión clara del nivel de satisfacción o inconformidad expresado a través de redes sociales. Se identifican los actores que podrían beneficiarse de esta información, como entidades municipales, investigadores o medios de comunicación, y se delimitan los criterios de éxito del proyecto: funcionalidad del prototipo, visualización efectiva y análisis automatizado de publicaciones en español.

Comprensión de los Datos

Aquí se analizan las fuentes de datos disponibles, en este caso, publicaciones realizadas en la red social X durante el 2024. Lo que se hace es identificar términos clave como hashtags relacionados con movilidad urbana (#MetroDeQuito, #Ecovía, #MovilidadQuito, entre otros), y determinar el volumen estimado de datos a recolectar, luego pasamos a la evaluación de estructura de los textos, frecuencia, longitud y formatos utilizados por los usuarios.

Preparación de los Datos

Esta fase nos sirve para en el procesamiento previo necesario para que los datos puedan ser utilizados, donde se incluye la recolección automatizada de publicaciones, la limpieza del texto (eliminación de menciones, URLs, caracteres especiales, etc.), tokenización y detección de idioma, para asegurar que los textos sean en español. También se considera la anonimización parcial del contenido y su conversión a un formato estructurado para su posterior análisis.

Modelado

En esta fase se aplica un modelo de análisis de sentimiento usando técnicas de NLP. Se utilizará un modelo preentrenado en idioma español, el cual será integrado en el sistema para clasificar automáticamente las publicaciones en tres categorías: positivas, negativas o neutras, también se ajustarán los parámetros necesarios y se definirá una muestra de publicaciones para validar su precisión mediante comparación con clasificaciones manuales.

Evaluación

Para esta fase lo que se va a hacer es evaluar la eficacia del sistema a nivel técnico y funcional. Se medirá la exactitud del modelo de sentimiento a través de una muestra anotada manualmente, posteriormente se realizarán pruebas de caja negra sobre el prototipo para verificar que cada uno de sus módulos (recolección, procesamiento y visualización) funcione correctamente. Asimismo, se evaluará la claridad de las visualizaciones mediante criterios de usabilidad e interpretación. Estas evaluaciones permitirán validar el cumplimiento de los objetivos y la utilidad del sistema como herramienta de análisis ciudadano.

Implementación

La última fase concluye con la construcción del prototipo final y su integración en una plataforma web funcional. Este sistema incluirá una interfaz sencilla e intuitiva que permita visualizar los resultados obtenidos de forma clara y con filtros importantes (por fecha, tipo de sentimiento o volumen de publicaciones).

2.2.1.1 Implementación de CRISP DM para el prototipo de plataforma web de Ciencia de Datos para el Estudio del Sentimiento Ciudadano sobre el Transporte Público en Quito durante el Año 2024

La implementación de la metodología CRISP-DM en este proyecto no se limita a una descripción teórica de sus fases, sino que se también incluye la forma en que será desarrollado el prototipo de plataforma web y los módulos correspondientes. Cada fase se ejecutará con herramientas específicas, técnicas concretas y resultados esperados que contribuyen al producto final. En la fase de comprensión del negocio se definieron los objetivos del sistema y los requisitos del análisis, considerando las necesidades de actores reales como la ciudadanía, los entes municipales y el ámbito académico. Esto permitió establecer criterios como la usabilidad del prototipo, el tipo de datos a usar y el enfoque en publicaciones de X sobre el transporte público.

Durante la comprensión de los datos, se seleccionaron fuentes, palabras clave y hashtags que tengan relación con el transporte público en Quito. En la preparación de los datos, se usaron librerías de Python como pandas, re, spaCy y nltk para limpiar, filtrar y transformar los textos. Esto se realiza para determinar la calidad del input que se alimentará al modelo de análisis de sentimiento. También se incluyen funciones específicas para eliminar duplicados, detectar idioma, y conservar publicaciones significativas.

La fase de modelado se desarrollará integrando un modelo de análisis de sentimiento preentrenado en español, como los disponibles en pysentimiento o Hugging Face. Este modelo se conectará al backend del sistema mediante una función personalizada que permita automatizar la clasificación del sentimiento de cada publicación de forma sencilla para el usuario.

Para probar su efectividad inicial, el modelo se validará con una muestra de publicaciones que serán etiquetadas manualmente. En la evaluación, se tomarán en cuenta indicadores técnicos como la precisión del modelo, el porcentaje de publicaciones correctamente clasificadas y el rendimiento general del sistema en términos de velocidad, consistencia y funcionalidad. Además, se revisará la calidad de las visualizaciones, poniendo énfasis en que sean claras y fáciles de interpretar para distintos tipos de usuarios.

La implementación del sistema se realizará usando el framwork Streamlit para el desarrollo de la interfaz web, y bibliotecas de visualización como Plotly y Seaborn para generar los dashboards. El prototipo incluirá filtros por tipo de sentimiento y periodos de tiempo ya que nos permiten analizar y explorar la percepción ciudadana de mejor manera.

2.2.1.1 Beneficios de la aplicación de CRISP-DM para el prototipo de plataforma web de Ciencia de Datos para el Estudio del Sentimiento Ciudadano sobre el Transporte Público en Quito durante el Año 2024

La adopción de la metodología CRISP-DM (Cross-Industry Standard Process for Data Mining) en este proyecto es clave ya que contribuye directamente en el desarrollo y validación del prototipo de análisis de sentimiento ciudadano.

Uno de sus principales beneficios es su estructura clara y cíclica, que permite organizar el trabajo en fases bien definidas: comprensión del negocio, comprensión de los datos, preparación, modelado, evaluación e implementación. Con esto nos aseguramos que cada paso del proceso sea abordado con un enfoque sistemático, reduciendo la posibilidad de errores o soluciones poco justificadas.

Además, CRISP-DM se adapta naturalmente a proyectos de ciencia de datos, como el que se desarrolla en este trabajo, ya que está diseñado específicamente para análisis basado en datos no estructurados, como publicaciones en redes sociales.

Esta metodología facilita el flujo de trabajo al permitir iteraciones entre fases: por ejemplo, si durante la fase de modelado se obtienen resultados poco precisos, se puede retornar a la fase de preparación de datos para ajustar el preprocesamiento para mejorar los resultados.

Otro beneficio importante es que ayuda a mantener separadas las responsabilidades entre los aspectos técnicos y los conceptuales. Cada etapa permite dejar registro de las decisiones tomadas, algo clave para poder explicar por qué se eligieron ciertos enfoques y garantizar la trazabilidad y la posibilidad de replicar el trabajo más adelante.

Además, CRISP-DM facilita medir el avance del proyecto de manera más clara y objetiva, ya que cada fase tiene resultados concretos y entregables definidos. Esto resulta muy útil para organizar el tiempo y los recursos, sobre todo en un contexto con plazos establecidos como es el caso de un trabajo de titulación.

2.3 Herramientas

Para llevar a cabo el desarrollo del prototipo de plataforma web de ciencia de datos orientado al análisis del sentimiento ciudadano sobre el transporte público en Quito, se decidió el uso de herramientas tecnológicas modernas, de código abierto y ampliamente utilizadas en la actualidad. Estas herramientas fueron seleccionadas en base a criterios como facilidad de integración, eficiencia en el manejo de datos, disponibilidad de documentación, escalabilidad y compatibilidad entre sí.

El objetivo es construir un sistema funcional y accesible que permita recolectar, procesar, analizar y visualizar información obtenida a partir de publicaciones en la red social X, mediante técnicas de procesamiento de lenguaje natural (NLP) y análisis de sentimiento. El uso combinado de lenguajes de programación, librerías de análisis, frameworks web y sistemas de almacenamiento de datos permite desarrollar una solución robusta, reutilizable y adaptable a diferentes necesidades investigativas. A continuación, se describen las herramientas tecnológicas que formarán parte del desarrollo del prototipo:

- **Python** (Lenguaje de programación para backend, análisis de texto y procesamiento de datos).

- **Pandas y NumPy** (Librerías para manipulación y análisis de datos estructurados).
- **SpaCy y NLTK** (Librerías para procesamiento de lenguaje natural en español).
- **PySentimiento** (Herramienta de análisis de sentimiento especializada en idioma español).
- **Streamlit** (Framework de desarrollo web ligero para construir dashboards).
- **Plotly y Seaborn** (Librerías para la visualización gráfica de datos).
- **Supabase** (Base de datos y backend como servicio, basada en PostgreSQL).
- **Visual Studio Code** (Entorno de desarrollo para escribir, organizar y depurar código).
- **Git y GitHub** (Control de versiones y repositorio remoto para gestión de código).

2.3.1 Python

Python es un lenguaje de programación interpretado, multiplataforma y de código abierto, ampliamente utilizado en el ámbito académico y profesional por su versatilidad, sintaxis sencilla y gran ecosistema de librerías. Su adopción masiva en proyectos de ciencia de datos se debe a la facilidad con la que permite integrar procesos de análisis, visualización, automatización y desarrollo de aplicaciones, todo en un solo entorno.

Para el presente proyecto, Python es la base fundamental sobre la cual se desarrollará el backend del sistema. Se utilizará para implementar scripts encargados de recolectar publicaciones desde la red social X, preprocesar el texto mediante técnicas de NLP (tokenización, lematización, limpieza de caracteres especiales, entre otros), aplicar modelos de análisis de sentimiento y generar los datos que posteriormente se visualizarán en la interfaz web.

Una de las ventajas de Python es su capacidad de integrarse con herramientas externas y su compatibilidad con bibliotecas avanzadas como pandas, numpy, matplotlib, seaborn o plotly. Además, permite el uso de modelos de inteligencia artificial y redes neuronales sin necesidad

de escribir algoritmos desde cero, gracias a plataformas como transformers o pysentimiento. Otra razón importante para su elección es la gran comunidad de usuarios y la gran cantidad de documentación disponible, lo que facilita resolver dudas, aplicar buenas prácticas y acceder a recursos ya existentes.

Su integración con frameworks como Streamlit, que permite construir interfaces web con pocas líneas de código, convierte a Python en una herramienta ideal para el desarrollo rápido de prototipos funcionales, especialmente en entornos educativos o de investigación.

2.3.2 Pandas y NumPy

Pandas y NumPy son dos de las librerías fundamentales del ecosistema de Python para el tratamiento y análisis de datos. Ambas se utilizan en combinación para manipular estructuras de datos eficientes, realizar cálculos complejos y preparar la información para su análisis o visualización.

Pandas se especializa en el manejo de datos tabulares y estructurados mediante estructuras como los DataFrames, que funcionan de manera similar a tablas en una base de datos o hojas de cálculo. Esta herramienta permite leer, limpiar, transformar, filtrar, agrupar y exportar datos con gran facilidad. Dentro del proyecto, se utilizará Pandas para organizar la información extraída de las publicaciones en redes sociales, aplicar filtros por fecha o términos clave, y preparar el dataset para el análisis de sentimiento.

Por su parte, NumPy (Numerical Python) está diseñada para el manejo eficiente de arreglos multidimensionales y operaciones matemáticas de alto rendimiento (Developers, n.d.). Aunque muchas veces actúa de forma "invisible" en conjunto con Pandas o Matplotlib, su rol es esencial para realizar operaciones vectorizadas, estadísticas y cálculos intermedios.

El uso combinado de estas herramientas permitirá realizar procesos como la detección de duplicados, la normalización de textos, el cálculo de la distribución de sentimientos, y la organización del flujo de datos para su visualización posterior en dashboards. La elección de Pandas y NumPy se debe al rendimiento que ofrece y la simplicidad en el desarrollo, lo que las convierte en herramientas ampliamente documentadas, con soporte comunitario y actualizaciones constantes.

2.3.3 SpaCy y NLTK

SpaCy y NLTK (Natural Language Toolkit) son dos de las bibliotecas más importantes y conocidas en Python para el análisis computacional del lenguaje humano. Estas herramientas facilitan operaciones lingüísticas como la división de textos en unidades (tokenización), reducción de palabras a su raíz (lematización), estudio de la estructura gramatical y filtrado de términos irrelevantes (stopwords), entre otras funcionalidades. Su aplicación resulta indispensable cuando se trabaja con contenido no estructurado proveniente de plataformas sociales como en el presente trabajo.

NLTK, como solución pionera en el campo del NLP, se caracteriza por su orientación académica e investigativa. Ofrece una extensa variedad de recursos lingüísticos, utilidades para el tratamiento previo de textos y ejemplos prácticos listos para implementar. Destaca particularmente por su versatilidad en áreas como el análisis estructural de oraciones, creación de modelos lingüísticos y representación gráfica de datos textuales. En el contexto de esta investigación, NLTK se empleará fundamentalmente para: filtrar palabras vacías, segmentar textos en componentes básicos e identificar los términos de mayor recurrencia (Bird, Klein, & Loper, Natural Language Processing with Python, 2009).

En contraste, SpaCy representa un enfoque más contemporáneo, priorizando el rendimiento operativo y la capacidad de manejar grandes volúmenes de información. Su arquitectura está optimizada para integrarse eficientemente en flujos de trabajo productivos, superando significativamente en velocidad a otras alternativas tradicionales (Explosion AI, n.d.). Para este proyecto, SpaCy cumplirá funciones esenciales como: la normalización de palabras a sus formas canónicas, identificación de entidades relevantes y examen de relaciones gramaticales, mejorando sustancialmente la preparación de los datos antes del análisis de sentimientos.

La combinación estratégica de SpaCy y NLTK permite sintetizar sus respectivas ventajas: por un lado, la amplitud metodológica de NLTK, y por otro, la eficiencia operacional de SpaCy. Esta sinergia asegura un procesamiento lingüístico preciso y confiable, particularmente valioso cuando se trabaja con el lenguaje coloquial y variable típico de las interacciones en redes sociales.

Mediante la implementación de estas tecnologías, el sistema será capaz de convertir expresiones informales y espontáneas en información estructurada y analizable, reduciendo al mínimo las interpretaciones erróneas y optimizando la calidad de los resultados obtenidos.

2.3.4 PySentimiento

PySentimiento es una librería de análisis de sentimiento desarrollada específicamente para textos en español, basada en modelos de lenguaje preentrenados de última generación, como BERT y RoBERTa. Su principal ventaja frente a otras herramientas genéricas es que ofrece modelos finos para idioma español, lo que la convierte en una solución muy adecuada para proyectos de análisis de texto en contextos hispanohablantes (Lewis Tunstall, 2021).

En este proyecto, PySentimiento será la encargada de clasificar automáticamente las publicaciones en redes sociales en categorías de sentimiento: positivo, negativo o neutro. A diferencia de otros enfoques que requieren entrenar modelos desde cero, PySentimiento permite aplicar modelos ya entrenados con datasets en español, lo cual ahorra tiempo, mejora la precisión y reduce los requerimientos computacionales.

Técnicamente, la librería se integra fácilmente con Python y es compatible con otras herramientas de procesamiento de texto como spaCy o transformers. Su uso es sencillo, y permite aplicar directamente un análisis semántico sobre frases completas sin necesidad de una estructura gramatical perfecta, lo que es ideal para redes sociales, donde el lenguaje suele ser informal, con errores, emoticonos o abreviaciones.

Además de su precisión, PySentimiento permite obtener puntuaciones de confianza, lo cual es útil para validar qué tan claro es el sentimiento de una publicación. También admite otros tipos de clasificación emocional, como emociones básicas (alegría, enojo, miedo, etc.). La elección de PySentimiento se debe a la necesidad de tener un modelo sólido, especializado en español y orientado a análisis de opinión, que pueda aplicarse directamente a un entorno real como las publicaciones sobre el transporte público en Quito.

2.3.5 Streamlit

Streamlit es una herramienta especializada para crear aplicaciones web interactivas en Python, enfocada específicamente en proyectos de análisis de datos e inteligencia artificial. Su diseño

simplificado permite construir interfaces funcionales con mínimas líneas de código, prescindiendo de los conocimientos avanzados en desarrollo frontend que requieren otros frameworks más complejos (Hall, 2022).

Para esta investigación, Streamlit servirá como plataforma fundamental para diseñar la interfaz del prototipo, transformando los resultados del análisis de sentimientos en visualizaciones comprensibles para usuarios sin formación técnica. La solución incluirá gráficos interactivos, sistemas de filtrado por diferentes criterios y herramientas para explorar los datos de forma intuitiva. El valor diferencial de Streamlit reside en su simplicidad: con comandos básicos de Python es posible implementar dashboards completos con diversos componentes visuales. Esta característica lo hace particularmente adecuado para proyectos académicos que requieren prototipos funcionales en plazos reducidos, optimizando el tiempo de desarrollo.

La plataforma ofrece flexibilidad de implementación, soportando tanto despliegues locales como en servicios cloud especializados. Esta versatilidad permite presentar el proyecto en diferentes contextos, desde evaluaciones académicas hasta demostraciones con stakeholders, mediante un simple enlace web compartible. Su compatibilidad nativa con las principales bibliotecas de visualización de Python establece un flujo de trabajo integrado, conectando directamente el procesamiento de datos con su representación gráfica. Esta integración asegura una experiencia de usuario fluida y profesional, cumpliendo con los objetivos de accesibilidad planteados en el proyecto.

2.3.6 Plotly y Seaborn

Plotly y Seaborn son bibliotecas de visualización de datos en Python que permiten generar gráficos estadísticos y representaciones interactivas de alto nivel. En el contexto del presente proyecto, ambas serán utilizadas de forma complementaria para mostrar visualmente los resultados del análisis de sentimiento aplicado a las publicaciones extraídas desde la red social X.

Plotly es una herramienta especialmente potente para la creación de gráficos interactivos. Permite al usuario explorar los datos a través de elementos dinámicos como zoom, filtros, selección de rangos, y tooltips informativos. Gracias a su integración con Streamlit, los gráficos generados pueden ser incorporados directamente en la interfaz web del prototipo. Esta

característica facilita la creación de dashboards que no solo muestran información estática, sino que permiten interactuar con los resultados en tiempo real, mejorando la experiencia del usuario.

Por su parte, Seaborn se utiliza para la creación de gráficos estadísticos claros y estéticos, basados en estructuras de datos como los DataFrames de Pandas. Seaborn permite representar distribuciones, correlaciones, comparaciones y series temporales con una sintaxis sencilla y resultados visuales de calidad. En el proyecto, se utilizará especialmente para análisis exploratorio de datos durante la fase de desarrollo y evaluación del modelo.

El uso combinado de ambas herramientas permite cubrir tanto la fase interna de análisis técnico como la fase de presentación visual final.Así, con Seaborn se podrá validar la distribución de los sentimientos de forma temporal, por ejemplo por semana o por tema, mientras que con Plotly se mostrará esa información en un dashboard final accesible e interactivo.

La correcta representación visual de los datos es esencial para cumplir uno de los objetivos fundamentales del prototipo: interpretar de forma clara la percepción ciudadana sobre el transporte público. Gracias a estas bibliotecas, se podrán construir gráficos como:

- Líneas de tiempo de sentimiento por mes
- Comparaciones entre temas recurrentes (#MetroDeQuito, #Ecovía, etc.)
- Gráficos de pastel o barras sobre la proporción de opiniones positivas, negativas y neutras
- Nubes de palabras sobre los términos más frecuentes

En conjunto, Plotly y Seaborn facilitarán no solo la comprensión de los resultados por parte del usuario, sino también su análisis técnico para validar patrones y generar conclusiones fundamentadas.

2.3.7 Supabase

Supabase es una plataforma de código abierto que provee una solución completa de backend como servicio (BaaS), basada en la tecnología de bases de datos relacionales PostgreSQL. Es una alternativa moderna y accesible a servicios como Firebase, ofreciendo funcionalidades

como autenticación, almacenamiento, funciones serverless, y una API RESTful lista para usar, además de que representa una solución integral de backend.

En el presente proyecto, Supabase será utilizada como sistema de gestión de base de datos para almacenar publicaciones recolectadas, resultados del análisis de sentimiento y configuraciones del sistema. Esta elección responde a su facilidad de integración con Python, su compatibilidad con estructuras de datos complejas, y su enfoque en rapidez de desarrollo, características clave en el contexto de un prototipo académico funcional.

A través de Supabase se podrá mantener un registro persistente de:

- Tweets recolectados desde la red social X (texto, fecha, sentimiento asignado)
- Estadísticas de análisis agregadas por periodo de tiempo
- Preferencias de usuario dentro del sistema (por ejemplo, filtros aplicados)

Uno de los principales beneficios de Supabase es que no requiere infraestructura propia para su implementación. El servicio puede desplegarse en la nube de forma gratuita durante el desarrollo, y luego escalar si fuera necesario. También ofrece una interfaz visual amigable para la gestión de las tablas y permite configurar roles y permisos para proteger los datos.

La API de Supabase permite conectar la base de datos con el sistema desarrollado en Python, utilizando librerías como supabase-py. Esta integración facilitará operaciones como lectura y escritura de datos, exportación de resultados o sincronización de publicaciones analizadas.

Además, Supabase incorpora funciones de autenticación, lo que abre la posibilidad de extender el prototipo para múltiples usuarios en el futuro, o de proteger ciertas áreas del sistema si se desea implementar una versión más avanzada.

La incorporación de Supabase en el proyecto garantiza un backend moderno, flexible y adaptable al flujo de datos necesario en un sistema de ciencia de datos, todo sin la complejidad de gestionar servidores o instancias locales.

2.3.8 Visual Studio Code

Según su documentación oficial, Visual Studio Code (VS Code) es un editor de código fuente desarrollado por Microsoft, reconocido por su rendimiento, ligereza y gran capacidad de personalización (Microsoft, 2016). Es uno de los entornos de desarrollo más utilizados por programadores, científicos de datos y estudiantes, gracias a su soporte nativo para múltiples lenguajes de programación y su integración con herramientas modernas.

VS Code ofrece una interfaz limpia, intuitiva y configurable, lo que lo convierte en una opción ideal para quienes buscan eficiencia y productividad en la escritura de código. Entre sus características más destacadas se encuentran la compatibilidad con Python, el soporte para control de versiones con Git, y la posibilidad de trabajar con entornos virtuales de forma sencilla.

En el contexto de este proyecto, Visual Studio Code será utilizado como el IDE principal para el desarrollo del prototipo web. Su integración con extensiones especializadas como Python Extension, Pylance, Jupyter, y herramientas para conectarse con bases de datos como Supabase o PostgreSQL, lo convierten en un entorno robusto que facilita tanto el desarrollo como la depuración del sistema.

Además, cuenta con un sistema de extensiones altamente personalizable, que permite al desarrollador añadir funcionalidades según las necesidades del proyecto. Por ejemplo, se pueden instalar herramientas específicas para trabajar con Streamlit, Plotly, control de versiones, o incluso complementos para la gestión de tareas o análisis estático del código.

Otro aspecto clave de su elección es su portabilidad y gratuidad. VS Code está disponible en todos los sistemas operativos principales y no requiere licencias comerciales, lo que lo hace accesible para entornos educativos y académicos. Su entorno de trabajo amigable, combinado con una amplia comunidad de soporte, lo convierten en una herramienta confiable y adecuada para el desarrollo de sistemas modernos como el que se plantea en este proyecto.

2.3.9 Playwright

Como parte del proceso de adquisición de datos, fue necesario implementar una solución de recolección automatizada de publicaciones desde la red social X, debido a las limitaciones de

acceso de su API oficial y a la necesidad de recopilar grandes volúmenes de texto público con criterios temáticos y temporales específicos.

Para este propósito se utilizó Playwright, una herramienta open source desarrollada por Microsoft para la automatización de navegadores web. Esta librería permite simular la interacción humana con sitios web, facilitando tareas como la navegación, el desplazamiento dinámico, la extracción de información del DOM y el manejo de sesiones autenticadas.

El script desarrollado en Python emplea Playwright en conjunto con cookies exportadas desde el navegador Chrome, lo que permite ejecutar sesiones autenticadas y acceder a resultados de búsqueda completos, tal como los vería un usuario registrado en la plataforma. Las consultas automatizadas incluyen operadores como since, until y lang:es, que filtran los resultados por fecha, idioma y palabras clave relevantes al dominio del transporte urbano.

Durante cada ejecución, el script recorre dinámicamente los resultados desplazándose por la página, captura elementos como el contenido del tweet, su fecha, autor y enlace, y los almacena en archivos CSV codificados en UTF-8. Para evitar bloqueos o penalizaciones, se incorporaron tiempos de espera programados entre ciclos de extracción.

Esta herramienta permitió construir una base de datos personalizada y alineada con los objetivos del proyecto, respetando al mismo tiempo el acceso a información pública disponible en línea. La automatización del proceso de scraping no solo agilizó la recopilación, sino que también garantizó una estructura uniforme y reutilizable para futuras campañas de análisis.

2.4 Comparativa de las Herramientas Utilizadas

Esta sección tiene como objetivo justificar la selección de cada una de las herramientas tecnológicas empleadas en el desarrollo del prototipo, mediante una comparativa frente a otras alternativas disponibles en el mercado y en la comunidad open source. Las tablas a continuación presentan una descripción, ventajas, desventajas y la razón de su elección dentro del proyecto.

2.4.1 Supabase vs Firebase

La herramienta seleccionada es Supabase, debido a su compatibilidad con estructuras relacionales, su integración nativa con Python y su naturaleza open source, lo que facilita su uso académico y permite mantener flexibilidad en el manejo de datos.

Tabla 1.

Comparativa entre Supabase y Firebase

Criterio	**Supabase**	**Firebase**
Tipo de base de datos	Relacional (PostgreSQL)	No relacional (Firestore, Realtime DB)
Código abierto	Sí	No
Lenguaje de consulta	SQL estándar	Propio / limitaciones en consultas complejas
Integración con Python	Directa (supabase-py)	A través de SDKs con menos flexibilidad
Panel de administración	Visual, similar a pgAdmin	Visual, propietario

2.4.2 Comparativa: PySentimiento vs TextBlob vs VADER

En este proyecto se utilizará PySentimiento, por estar especialmente diseñado para el idioma español y por su efectividad en textos informales como los presentes en redes sociales. Esta elección permite una clasificación más precisa sin necesidad de entrenamiento adicional.

Tabla 2.

Comparativa: PySentimiento vs TextBlob vs VADER

Criterio	**PySentimiento**	**TextBlob**	**VADER**
Idioma soportado	Español (y otros)	Inglés (algunas funciones en español)	Inglés

Modelo base	BERT/RoBERTa preentrenado	Reglas + Naive Bayes	Lexicón + reglas heurísticas
Precisión	Alta en español informal	Media	Alta en inglés, baja en español
Entrenamiento previo	No necesario (ya entrenado)	No necesario	No necesario

2.4.3 Comparativa: Streamlit vs Flask vs Dash

Se ha optado por Streamlit como framework principal para el desarrollo de la interfaz web, debido a su facilidad de uso, su integración directa con Python y su idoneidad para construir prototipos funcionales enfocados en visualización de datos.

Tabla 3.

Stramlis vs Flask vs Dash

Criterio	Streamlit	Flask	Dash
Enfoque	Visualización de datos / prototipos	Backend general	Apps analíticas
Curva de aprendizaje	Muy baja	Media	Alta
Visualización integrada	Sí	No (requiere HTML/CSS)	Sí
Tiempo de desarrollo	Corto	Medio	Largo

2.4.4 Comparativa: Plotly vs Matplotlib vs Seaborn

Se seleccionó Plotly como herramienta principal de visualización, ya que permite crear dashboards interactivos totalmente integrables con Streamlit. Su capacidad para generar

gráficos dinámicos mejora la experiencia de usuario y facilita la exploración visual de los resultados.

Tabla 4.

Ploty vs Matplotlib vs Seaborn

Criterio	Plotly	Matplotlib	Seaborn
Interactividad	Alta (gráficos interactivos)	Nula (gráficos estáticos)	Baja
Facilidad de uso	Media	Baja	Alta (sobre Matplotlib)
Estilo visual	Moderno y adaptable	Básico y flexible	Estético y estadístico
Integración con Streamlit	Nativa	Indirecta	Indirecta

2.5 Conceptos fundamentales del análisis de sentimiento

El análisis de sentimiento es una técnica del procesamiento de lenguaje natural (NLP) que busca identificar y clasificar las emociones, opiniones o actitudes que las personas comunican en textos escritos. Su finalidad principal es determinar si el contenido expresa una valoración positiva, negativa o neutra, y en algunos casos, reconocer emociones más específicas como alegría, tristeza, miedo, ira o sorpresa.

En los últimos años, este tipo de análisis ha cobrado mucha importancia debido al crecimiento de las redes sociales, foros y otras plataformas donde los usuarios comparten grandes volúmenes de contenido de forma espontánea. En el ámbito de la movilidad urbana, el análisis de sentimiento resulta especialmente útil para que las instituciones puedan conocer cómo perciben los ciudadanos los servicios de transporte, identificar problemas frecuentes y tomar decisiones fundamentadas en la opinión pública.

A continuación, se explican los conceptos principales que ayudan a entender cómo funciona y cómo se puede implementar un sistema automatizado de análisis de sentimiento.

Polaridad: La polaridad representa la orientación general del sentimiento en un texto. Se clasifica comúnmente en tres categorías: positivo, negativo o neutro. Algunos modelos trabajan con escalas ordinales (por ejemplo, de 1 a 5) para reflejar distintos niveles de intensidad emocional. Una polaridad positiva puede implicar satisfacción, aprobación o entusiasmo; la negativa, descontento o crítica; mientras que la neutra suele representar textos informativos o sin carga emocional evidente.

Confianza o Score de Confianza: Es un valor numérico que indica la certeza del modelo al realizar una clasificación. Cuanto más alto sea este score (normalmente expresado como una probabilidad entre 0 y 1), mayor es la seguridad del modelo respecto a su predicción. Este valor puede ser utilizado para filtrar predicciones dudosas o para priorizar las más claras en un análisis.

Tokenización: Proceso por el cual el texto es dividido en unidades mínimas llamadas "tokens". Estas unidades suelen ser palabras, aunque en algunos contextos pueden ser caracteres o frases. Este paso es esencial para que los modelos de NLP procesen el texto de manera estructurada.

Lematización y Normalización: La lematización consiste en reducir una palabra a su forma base o "lema". Esto es útil para unificar variantes morfológicas de una palabra. Por ejemplo, "hablando", "habló" y "hablará" se lematizan a "hablar". La normalización incluye además pasos como convertir a minúsculas, eliminar signos de puntuación y caracteres especiales.

Eliminación de Stopwords: Las stopwords son palabras comunes que no aportan contenido semántico significativo al análisis, como "de", "que", "por", "en". Su eliminación ayuda a reducir el ruido en el procesamiento y enfocarse en los términos relevantes.

Modelos preentrenados vs. modelos entrenados desde cero: Los modelos preentrenados han sido construidos y ajustados previamente sobre grandes *corpus* de datos, y pueden aplicarse directamente sin necesidad de entrenamiento adicional, permitiendo la reducción considerable de tiempo y recursos necesarios. Ejemplos incluyen nlptown/bert-base-multilingual-uncased-sentiment o los modelos disponibles en PySentimiento. Por el contrario, entrenar un modelo desde cero requiere datasets etiquetados, potencia computacional y validación extensiva.

Sentimiento vs. emociones: El análisis de sentimiento se limita a clasificar textos en categorías amplias de polaridad (positivo, negativo, neutro), mientras que el análisis emocional va más allá al identificar emociones específicas como alegría, tristeza, miedo o ira. Algunos modelos, como los incluidos en PySentimiento, permiten ambas tareas, lo que amplía las capacidades del sistema.

Lenguaje informal y ruido textual: En redes sociales, el lenguaje tiende a ser informal, con abreviaciones, errores ortográficos, emoticonos, hashtags y modismos. Este ruido puede afectar la precisión del análisis, por lo que es necesario aplicar rutinas de limpieza y usar modelos adaptados a este tipo de texto.

Dominio específico del lenguaje: El rendimiento de un modelo de análisis de sentimiento puede variar según el dominio temático. Un modelo entrenado con reseñas de productos puede no funcionar adecuadamente con publicaciones sobre transporte público. Por ello, es recomendable utilizar modelos diseñados para el idioma y dominio más cercanos al objetivo del análisis.

Aplicaciones comunes del análisis de sentimiento:

- Monitoreo de opinión pública en tiempo real.
- Evaluación de campañas de comunicación institucional.
- Estudio de la percepción de servicios (salud, transporte, educación).
- Análisis reputacional de marcas o gobiernos.
- Prevención de crisis o conflictos mediante la detección de opiniones negativas acumuladas.

CAPÍTULO 3: DESARROLLO DEL PROTOTIPO BASADO EN LA METODOLOGÍA CRISP-DM

3.1 Comprensión del Negocio

La fase de comprensión del negocio es el primer paso en el ciclo de CRISP-DM y tiene como finalidad establecer una comprensión clara del contexto, necesidades, objetivos y restricciones del proyecto desde una perspectiva práctica y aplicada. En esta etapa, el enfoque es llevar el análisis técnico hacia una visión más general del problema que se busca resolver, permitiendo traducir los requerimientos de alto nivel en objetivos concretos de minería de datos.

En el presente trabajo, el "negocio" no se refiere a una empresa comercial tradicional, sino al contexto social y urbano del transporte público en la ciudad de Quito, y al interés de explorar cómo la ciudadanía percibe y expresa su experiencia cotidiana en torno a este servicio. Dado el creciente uso de redes sociales como canal de expresión ciudadana, se identificó una oportunidad valiosa para aprovechar expresiones públidas como insumo para análisis de opinión, sentimiento y comportamiento colectivo.

La comprensión del negocio en este proyecto implica analizar cómo el transporte público impacta a la población, cómo se manifiestan las opiniones en plataformas digitales, y de qué forma esta información puede ser recolectada, procesada y transformada en datos estructurados que faciliten la toma de decisiones, el monitoreo urbano o el desarrollo de políticas públicas más centradas en el usuario.

3.1.1 Objetivo Comercial

La metodología CRISP-DM plantea como primera fase la identificación de los objetivos comerciales que buscan ser alcanzados mediante la aplicación de modelos de minería de datos. No obstante, en el presente proyecto no se ha definido una empresa, institución u organización como actor principal, debido a que su enfoque es académico y de investigación aplicada en un contexto urbano y social. Por esta razón, no se plantean objetivos comerciales como se haría comunmente, sino que se establecen objetivos de minería de datos, orientados a responder una problemática ciudadana: la falta de análisis sistemático del sentimiento ciudadano respecto al transporte público en Quito.

El objetivo general de este modelo de minería de datos es diseñar y desarrollar una plataforma web de ciencia de datos que permita analizar el sentimiento ciudadano sobre el transporte público en Quito durante el año 2024, utilizando publicaciones extraídas de la red social X.

3.1.2 Evaluación de la situación actual

Actualmente, la ciudad de Quito no cuenta con un sistema automatizado ni una estrategia formal que permita comprender la percepción ciudadana sobre su sistema de transporte público en tiempo real. Las opiniones y quejas de la ciudadanía suelen difundirse en redes sociales, particularmente en X, lo cual representa una fuente rica en información, pero no estructurada ni sistematizada.

La ausencia de un mecanismo de análisis automatizado impide a instituciones gubernamentales, académicas o sociales acceder a una visión unificada del nivel de satisfacción, malestar o percepción que los usuarios tienen respecto al transporte público, especialmente en cuanto a la operación del Metro de Quito, la Ecovía, los buses y el Trole.

Además, la magnitud y velocidad con la que se generan publicaciones en redes sociales supera la capacidad de análisis manual. Esto convierte a las técnicas de procesamiento de lenguaje natural (NLP) y minería de texto en herramientas fundamentales para afrontar esta problemática de forma eficiente.

3.1.3 Objetivo de Minería

Los objetivos de minería de datos establecidos en este proyecto se alinean con el propósito de generar información procesada a partir de publicaciones ciudadanas en redes sociales. A través del uso de técnicas automatizadas de análisis de texto, el modelo busca transformar datos no estructurados en conocimiento útil para visualizar el sentimiento público respecto al transporte en la ciudad.

Los objetivos concretos son:

- Aplicar un modelo de análisis de sentimiento multiclase en español sobre publicaciones extraídas desde la red social X.

- Asignar automáticamente una etiqueta de polaridad (positivo, negativo o neutro) a cada texto procesado.

- Detectar el tipo de transporte mencionado en cada publicación utilizando reglas léxicas simples.
- Almacenar los resultados del análisis en una base de datos estructurada (Supabase).
- Evaluar la calidad del modelo mediante la comparación con una muestra de publicaciones etiquetadas manualmente.
- Permitir la visualización de los resultados a través de dashboards dinámicos y reportes mensuales generados automáticamente.

3.1.4 Plan de Proyecto

El desarrollo del prototipo sigue la estructura establecida por las seis fases de la metodología CRISP-DM. Cada una de ellas orienta un conjunto de actividades específicas en el marco de este trabajo de titulación, como se detalla a continuación:

Tabla 5.

Plan de Proyecto

Fase	Actividades principales
Comprensión del negocio	Definición del problema, análisis del contexto urbano, objetivos de minería de datos.
Comprensión de los datos	Recolección de publicaciones desde X, revisión preliminar, filtrado por palabras clave.
Preparación de los datos	Limpieza textual, tokenización, lematización, detección de idioma, normalización.
Modelado	Aplicación de modelo de sentimiento, clasificación, detección de tipo de transporte.
Evaluación	Validación del modelo con muestra etiquetada, cálculo de métricas: precisión, F1-score.
Implementación	Desarrollo de una plataforma web funcional (Streamlit) con visualizaciones y reportes.

3.2 Comprensión de los Datos

Una vez definido el problema y establecidos los objetivos de minería de datos, la segunda fase de la metodología CRISP-DM se enfoca en la comprensión de los datos disponibles. Esta etapa tiene como finalidad recolectar, describir, explorar y verificar la calidad de los datos que serán utilizados para construir el modelo de análisis de sentimiento. Dado que el presente proyecto se basa en publicaciones textuales extraídas desde la red social X, el análisis se centra en datos no estructurados.

3.2.1 Recolección inicial de los datos

Para llevar a cabo el análisis de sentimiento planteado en este proyecto, fue necesario construir una base de datos conformada por publicaciones obtenidas desde la red social X. Debido a las restricciones y limitaciones de acceso impuestas por la API oficial de esta red social, se optó por implementar un proceso de recolección automatizada mediante técnicas de web scraping.

El procedimiento se desarrolló utilizando la biblioteca playwright de Python, que permite la automatización de navegadores web con soporte para múltiples entornos. Se programó un script personalizado capaz de simular una sesión activa del usuario, utilizando cookies exportadas desde una extensión del navegador Google Chrome. Estas cookies se cargaban en conjunto a la navegación para iniciar sesión de forma autenticada permitiendo acceder a los resultados completos de búsqueda sin tener las limitaciones que un usuario no registrado tiene.

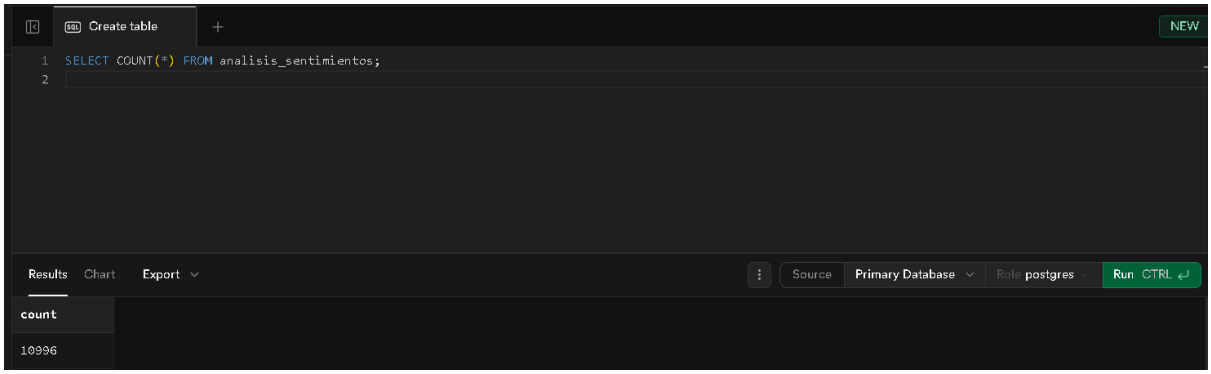
El script desarrollado funcionaba de la siguiente manera, a través de la búsqueda este recorre publicaciones relevantes filtradas por fechas y de forma mensual, estas consultas se ejecutaban directamente sobre la interfaz web de búsqueda de X, utilizando operadores como since y until para delimitar el período de interés, y el parámetro lang:es para asegurar que los resultados estuvieran redactados en español.

El contenido de los tweets, su fecha de publicación, usuario y enlace directo eran extraídos mediante selectores del DOM y almacenados estructuradamente en archivos CSV codificados en UTF-8. Durante el proceso, se configuró un número determinado de desplazamientos (scrolls) para capturar dinámicamente nuevas publicaciones, y se aplicaron pausas programadas para evitar bloqueos del sitio o sobrecarga del navegador. Este enfoque permitió obtener datos reales, correspondientes a opiniones emitidas por los usuarios sobre temas de transporte en la ciudad de Quito.

El resultado fue una colección de publicaciones filtradas por criterios temáticos y temporales, que fueron posteriormente sometidas a procesos de limpieza, análisis y visualización dentro de la plataforma. La Figura 1 presenta un fragmento del script utilizado para este proceso, donde se observan los parámetros configurables, las técnicas de extracción utilizadas y el formato en que se almacenan los datos.

Figura 1

Registros en BD



En total, se recolectaron 10.996 publicaciones únicas, organizadas en archivos CSV codificados en UTF-8.

3.2.2 Descripción de los datos

Los datos recopilados están compuestos principalmente por textos breves (tweets) acompañados de atributos derivados durante las fases de preprocesamiento y análisis semántico. Cada registro dentro del conjunto incluye múltiples variables que permiten realizar estudios cuantitativos y cualitativos sobre el sentimiento ciudadano respecto al transporte público en Quito.

A continuación, se describen los campos principales que conforman cada observación:

- **texto_original**: publicación original extraída directamente desde la red social X, sin alteraciones.

- **texto_limpio**: versión del texto tras ser sometido a una rutina de limpieza que elimina menciones a otros usuarios, enlaces URL, hashtags no informativos y signos innecesarios. Este campo constituye la entrada directa para el modelo de análisis de

sentimiento.

- **etiqueta**: categoría de sentimiento predicha por el modelo nlptown/bert-base-multilingual-uncased-sentiment, mapeada a tres clases: positivo, negativo y neutro.

- **calificación**: es un número del 1 al 5 que da el modelo original para mostrar con más detalle si el texto tiene una opinión positiva o negativa.

- **confianza**: probabilidad asociada a la predicción de sentimiento, útil como umbral para validaciones o análisis filtrados.

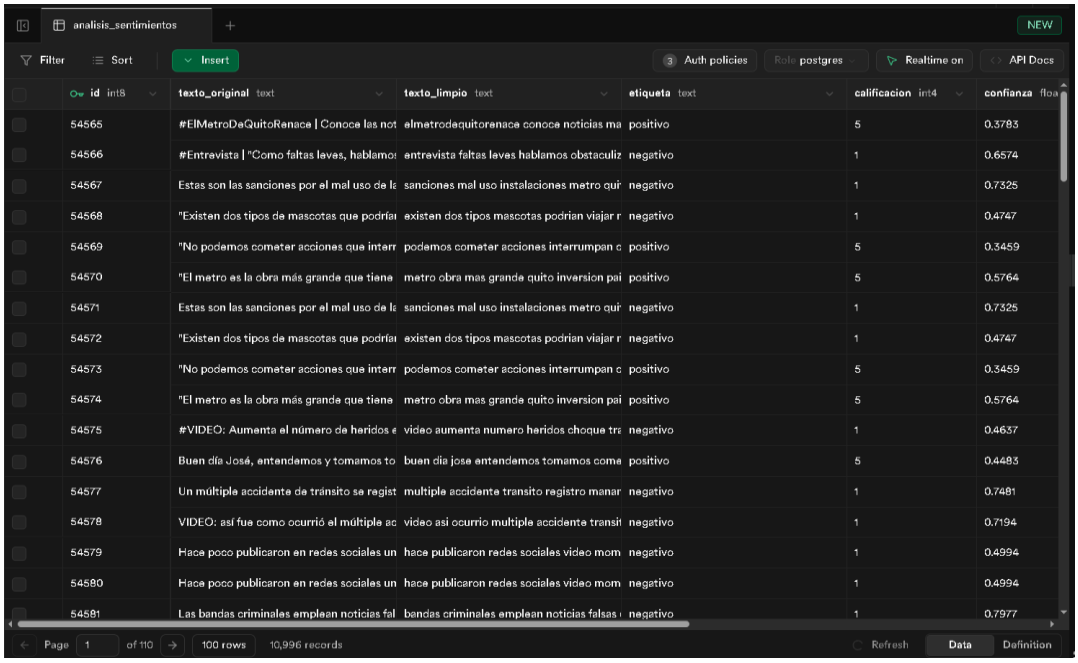
- **tipo_transporte**: clasificación temática del texto en función del modo de transporte mencionado (metro, ecovía, bus u otro), determinada mediante búsqueda de palabras clave.

- **fecha_creacion**: fecha original de publicación, empleada para análisis de evolución temporal del sentimiento.

Estos datos fueron almacenados y organizados en una tabla en Supabase, lo cual permite realizar operaciones SQL, filtrados personalizados y visualizaciones interactivas a través de la interfaz gráfica o mediante la conexión directa con la plataforma de análisis desarrollada en Streamlit. La Figura 2 nos muestra los registros de esta tabla además nos permite observar la estructura de datos implementada y el resultado de aplicar el pipeline completo de procesamiento, desde la recolección hasta la clasificación de sentimiento.

Figura 2

Registros de la BD.



3.2.3 Exploración de los datos

Durante la exploración inicial del conjunto de datos procesado desde Supabase, se identificó que el campo etiqueta presenta una distribución desigual. De un total de 1.000 publicaciones analizadas, la mayoría fue clasificada con sentimiento negativo, seguido por positivo y en menor medida neutro. Esta distribución puede observarse en la Figura 3.

Figura 3

Distribución de Sentimientos

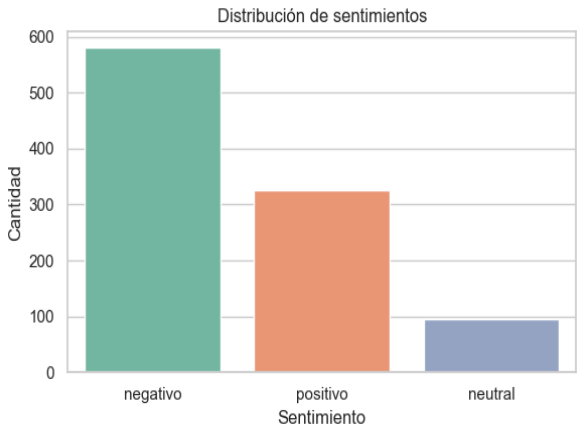
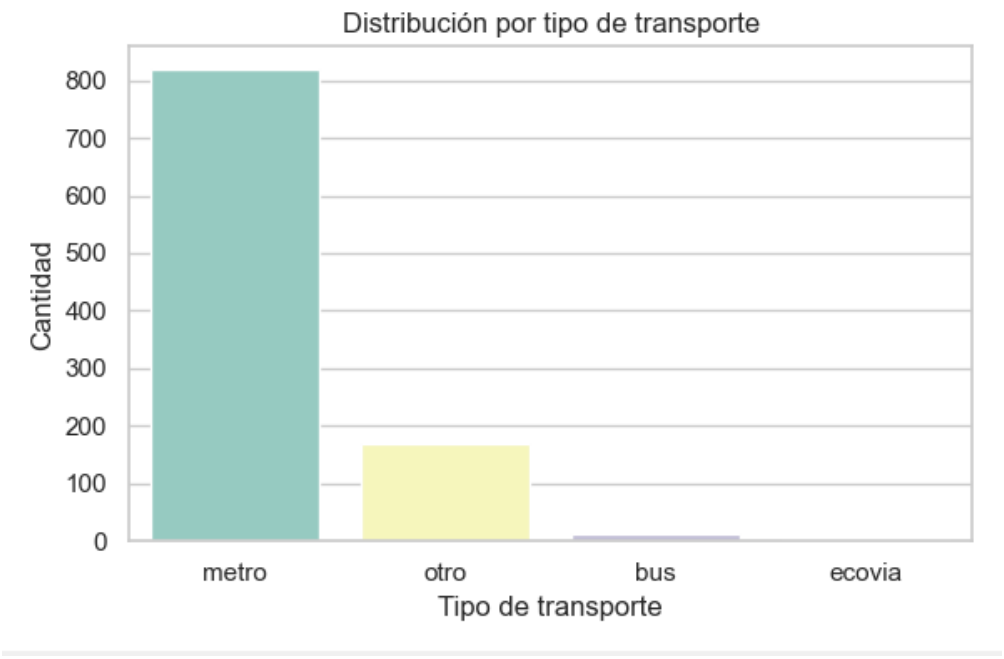


Figura 4

Distribución por tipo de transporte.



Además, se exploró el campo tipo_transporte, utilizado para identificar si una publicación hace referencia al metro, bus, ecovía u otro medio de transporte. La mayoría de los textos mencionaban explícitamente el metro.

La longitud promedio de los textos originales fue de aproximadamente 177 caracteres, con un mínimo de 3 y un máximo de 386 caracteres. Esto demuestra que el dataset está compuesto por publicaciones representativas en cuanto a contenido textual, además de obtener los siguientes datos con respecto a la frecuencia del tipo de transporte mencionado en cada publicación obtenida.

Tabla 6.

Frecuencia por tipo de transporte

Tipo de transporte	Frecuencia
Metro	818
Otro	168

Bus	12
Ecovía	2

3.2.4 Verificación de la calidad de los datos

Como parte del proceso de preparación para el modelado, se realizó una verificación exhaustiva de calidad sobre el conjunto de datos compuesto por 10.996 publicaciones provenientes de la red social X, ya procesadas y almacenadas en la base de datos Supabase.

Esta revisión tuvo como objetivo asegurar que los datos estuvieran libres de errores, redundancias o inconsistencias que pudieran afectar el análisis. A continuación, se detallan los principales aspectos evaluados:

- **Duplicados:** No se detectaron registros duplicados exactos en el conjunto de datos. Esta verificación se realizó mediante la función duplicated() de Pandas, permitiéndonos saber que no existen dos registros con el mismo contenido.

- **Registros vacíos:** Todos los campos texto_original contienen datos válidos, con esto nos aseguramos de que no existan registros vacíos o a medio completar.

- **Limpieza textual:** La transformación de texto_original a texto_limpio fue validada mediante inspección manual y ejemplos representativos. El preprocesamiento permitió eliminar elementos irrelevantes como hashtags, menciones, enlaces y signos de puntuación innecesarios, manteniendo el contenido semántico esencial para el análisis de sentimiento.

- **Longitud textual:** La longitud de los textos fue analizada para identificar publicaciones atípicas. Se observó una longitud promedio de 205 caracteres, con un mínimo de 3 y un máximo de 500, valores compatibles con el formato típico de publicaciones en redes sociales.

Tabla 7.

Métricas y Valor

	Métrica	Valor
0	Cantidad total de registros	10996.00
1	Duplicados detectados	0.00
2	Registros vacíos en texto_original	0.00
3	Longitud mínima del texto	3.00
4	Longitud promedio del texto	205.15
5	Longitud máxima del texto	500.00

Como resultado, se obtuvo un conjunto de datos limpio, consistente y adecuado para su posterior transformación y análisis en las siguientes fases del proyecto.

3.3 Preparación de los Datos

La fase de preparación de los datos tiene como finalidad transformar los datos brutos recolectados en una estructura adecuada para el análisis y modelado posterior. Esto incluye tareas como selección de atributos, limpieza, transformación, integración de fuentes y formateo. En este proyecto, al tratarse de datos textuales provenientes de redes sociales, la preparación se centró principalmente en el preprocesamiento lingüístico y en la adecuación del texto para ser interpretado por modelos de análisis de sentimiento en español.

3.3.1 Selección de los datos

A partir de los archivos obtenidos mediante técnicas de scraping uno por cada mes del año 2024 y filtrados mediante palabras clave temáticas, se realizó un proceso de selección de aquellos registros que cumplían con criterios de validez técnica. Se escogió a X como red social principal y fuente de extracción de los datos por sus publicaciones, su uso extendido en Ecuador y la frecuencia con la que los ciudadanos expresan opiniones en este caso particular acerca del transporte público en Quito. Además, esta plataforma permite acceder a publicaciones recientes en tiempo real, lo cual es esencial para un análisis de sentimiento actualizado y representativo.

Solo se incluyeron publicaciones redactadas en español, emitidas entre enero y diciembre de 2024, que contuviera al menos una de las palabras clave definidas. Se excluyeron los retweets,

publicaciones con contenido exclusivamente multimedia y aquellas que contenían menos de tres palabras, por considerarse insuficientes para un análisis semántico válido.

El proceso de scraping fue diseñado para que, al ejecutarse, este saque tweets filtrados de forma mensual, con la finalidad de que se tenga una la mayor cantidad de tweets posibles del 2024. Se emplearon términos clave como “Metro de Quito”, “Ecovía”, “trole”, “bus” y “movilidad”, entre otros, con el objetivo de cubrir los diferentes modos de transporte utilizados en la ciudad.

En la Tabla 2 podemos ver un ejemplo de cómo quedó la clasificación por tema: cuántas publicaciones terminan ligadas a cada tipo de transporte. Para armarla usé expresiones regulares en Python que buscaban palabras clave dentro de cada mensaje.

Si bien este enfoque permite una clasificación eficiente, se reconoce como limitación la posibilidad de omitir publicaciones indirectas, irónicas o formuladas con sinónimos no contemplados. No obstante, el volumen, la variedad temática y la distribución temporal de los datos seleccionados permiten que se pueda seguir con el proceso del análisis. Posteriormente, se cargaron en Supabase únicamente aquellas publicaciones que superaron estas validaciones, alcanzando un total de 10.996 registros.

Para validar la pertinencia de las palabras clave utilizadas en la clasificación temática, se realizó una prueba exploratoria sobre una muestra inicial de 500 publicaciones. Se observó que los términos “metro”, “ecovía”, “trole”, “bus” y “movilidad” eran los más frecuentes en publicaciones relacionadas con transporte público, lo que sustentó su selección para la categorización automática. Esta selección también fue contrastada con publicaciones oficiales de la Empresa Metropolitana de Transporte de Quito y noticias locales para asegurar su contextualización.

3.3.2 Limpieza de los datos

Por la forma en que llegan los textos tomados de la red social X, aparecen un montón de detalles que estorban al analizarlos de forma automática, hashtags, menciones, signos de puntuación fuera de lugar, enlaces, emojis y otros caracteres que no aportan significado. Para dejar todo listo, primero se pasó cada mensaje por un proceso de limpieza sistemático.

Para la limpieza se implementó la función clean_text(), ubicada en el módulo text_cleaner.py del proyecto. Mediante un script, la función se aplicó a cada registro y convirtió el campo contenido (texto original) en texto_limpio, preparado para las etapas de modelado posteriores.

Las operaciones realizadas en esta limpieza incluyeron:

- Conversión a minúsculas.
- Eliminación de menciones (@usuario), hashtags (#etiqueta) y URLs.
- Sustitución de signos de puntuación, números y emojis.
- Eliminación de signos de exclamación, comillas y caracteres especiales.
- Reducción de espacios múltiples.
- Eliminación de palabras sueltas no informativas (stopwords) y símbolos residuales.

El resultado de esta limpieza puede observarse en la siguiente tabla comparativa:

Tabla 8.

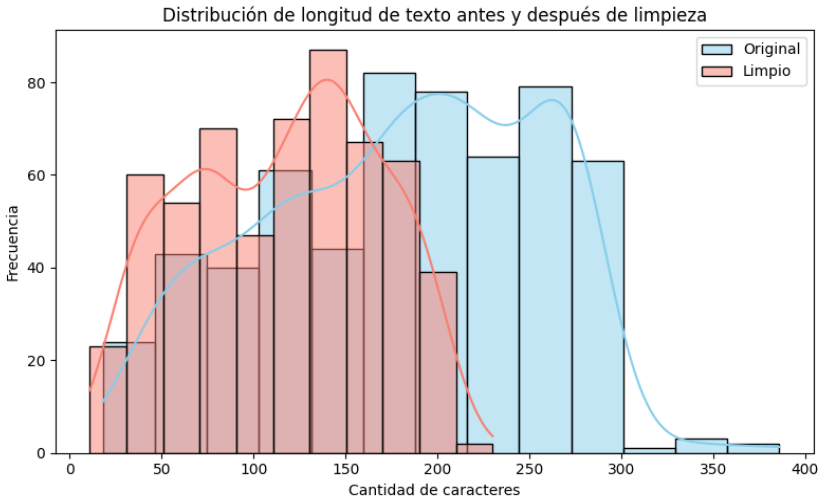
Registros en la Bd

	contenido	texto_limpio
0	#ElMetroDeQuitoRenace | Conoce las noticias m...	elmetrodequitorenace conoce noticias mas desta...
1	#Entrevista | "Como faltas leves, hablamos de...	entrevista faltas leves hablamos obstaculizar ...
2	Estas son las sanciones por el mal uso de las ...	sanciones mal uso instalaciones metro quito me...
3	"Existen dos tipos de mascotas que podrían via...	existen dos tipos mascotas podrian viajar metr...
4	"No podemos cometer acciones que interrumpan u...	podemos cometer acciones interrumpan convivenc...

Adicionalmente, se realizó un análisis de la longitud de los textos antes y después del preprocesamiento. Como se muestra en la figura 5, la limpieza textual redujo significativamente la longitud promedio de los textos, eliminando contenido redundante y facilitando su procesamiento por parte del modelo de análisis de sentimiento.

Figura 5

Distribución de longitud del texto antes y después de limpiarlo.



También se presenta un resumen estadístico de las longitudes de texto antes y después de la limpieza, donde se observa una disminución en la longitud promedio de 180.62 a 116.07 caracteres, manteniéndose los valores dentro de un rango lógico para análisis NLP:

Tabla 9.

Resumen de longitud de los tweets

	Métrica	Valor
0	Longitud promedio original	180.62
1	Longitud promedio limpio	116.07
2	Longitud mínima original	18.00
3	Longitud mínima limpio	11.00
4	Longitud máxima original	386.00
5	Longitud máxima limpio	230.00

3.3.3 Construcción de atributos derivados

El modelo utilizado para la clasificación de sentimiento fue nlptown/bert-base-multilingual-uncased-sentiment, disponible en Hugging Face. Este modelo fue seleccionado ya que no es necesario reentrenarlo y además permite el procesamiento de datos en español, también es bastante versátil por su balance entre precisión y velocidad de inferencia. Su arquitectura basada en BERT multilingüe le permite captar matices de polaridad incluso en textos informales o con errores gramaticales comunes en redes sociales.

Durante el proceso de preparación de los datos se generaron dos atributos derivados esenciales que permitieron enriquecer semánticamente cada registro textual y facilitar un análisis más profundo. El primero de estos atributos fue la etiqueta de sentimiento, denominada etiqueta, la cual fue asignada automáticamente mediante un modelo preentrenado de análisis de sentimiento en español.

Este modelo, implementado en el módulo sentiment_classifier.py del proyecto, clasifica cada texto limpio en una de tres categorías: positivo, negativo o neutral. Además, proporciona dos valores adicionales: una calificación subjetiva del sentimiento expresado en una escala del 1 al 5, y un puntaje de confianza numérica entre 0 y 1 que indica la certeza del modelo en su predicción.

El segundo atributo derivado fue tipo_transporte, que permite identificar semánticamente a qué modo de transporte se refiere el texto. Este atributo fue generado mediante la función detectar_transporte(), definida en el módulo classify_transport.py. La función aplica una serie de reglas léxicas simples sobre el texto limpio para clasificar el contenido en una de cuatro categorías: metro, ecovía, bus u otro.

Este mecanismo se basa en la presencia de palabras clave dentro de la oración, tales como “metro”, “trole”, “ecovía”, “bus” o sus variantes. Si no se encuentra ninguna coincidencia relevante, el texto es clasificado como “otro”. La incorporación de estas variables derivadas permitió no solo enriquecer la base de datos, sino también realizar análisis segmentados por polaridad de opinión y por medio de transporte, lo cual resultó especialmente útil en los dashboards visuales y en los reportes automáticos generados por la plataforma.

En la Tabla 10 se presenta una muestra de los resultados obtenidos al aplicar estas funciones sobre el texto limpio. Las Figuras 6 y 7, respectivamente, muestran la frecuencia de cada etiqueta de sentimiento y la distribución por tipo de transporte detectado.

Tabla 10.

Atributos de la BD

	texto_limpio	etiqueta	tipo_transporte
0	elmetrodequitorenace conoce noticias mas desta...	positivo	metro
1	entrevista faltas leves hablamos obstaculizar ...	negativo	metro
2	sanciones mal uso instalaciones metro quito me...	negativo	metro
3	existen dos tipos mascotas podrian viajar metr...	negativo	metro
4	podemos cometer acciones interrumpan convivenc...	positivo	metro
5	metro obra mas grande quito inversion pais inv...	positivo	metro
6	sanciones mal uso instalaciones metro quito me...	negativo	metro
7	existen dos tipos mascotas podrian viajar metr...	negativo	metro
8	podemos cometer acciones interrumpan convivenc...	positivo	metro
9	metro obra mas grande quito inversion pais inv...	positivo	metro

Figura 6

Frecuencia de etiquetas de sentimiento.

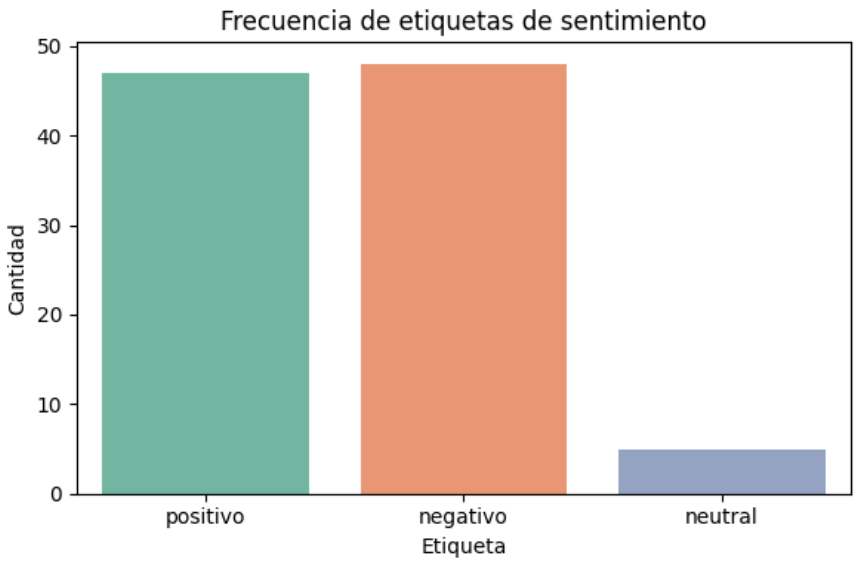
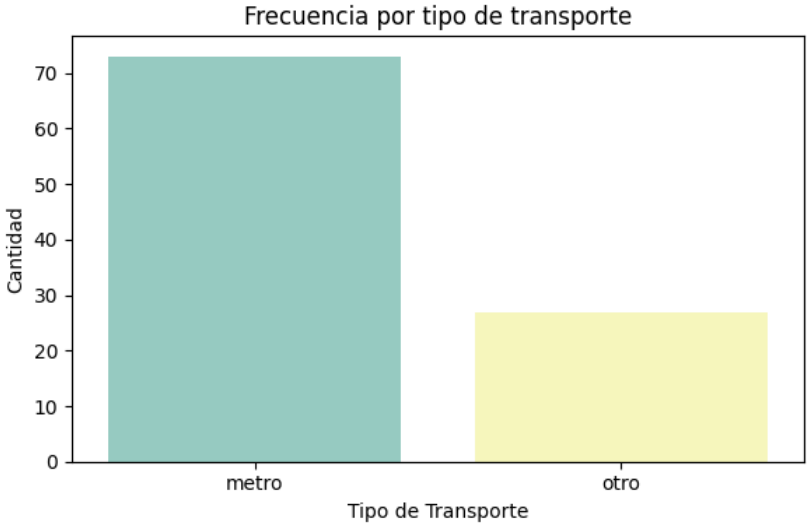


Figura 7

Frecuencia por tipo de transporte.



3.3.4 Formateo de los datos

Una vez generados y validados los atributos derivados, se procedió al formateo final de cada registro con el fin de garantizar la compatibilidad con la plataforma de análisis y visualización, así como con el sistema de almacenamiento en la nube. Esta etapa fue crucial para asegurar la coherencia de los datos en todo el sistema, desde la inserción en la base de datos hasta la consulta en tiempo real desde la interfaz web desarrollada con Streamlit.

El proceso de formateo consistió en la conversión de las fechas al estándar ISO 8601 (YYYY-MM-DDTHH:MM:SSZ) mediante el método datetime.isoformat(), lo cual permitió que las consultas, filtrados y agrupaciones temporales funcionaran correctamente tanto en Supabase como en las herramientas de visualización. También, se estableció una estructura para todos los registros resultando en los siguientes campos: texto_original, texto_limpio, etiqueta, calificacion, confianza, tipo_transporte y fecha_creacion.

Esta estructura se aplicó a cada fila antes de ser insertada en la base de datos mediante la función insertar_sentimiento(), asegurando que ningún registro quede incompleto o con formato inconsistente. Para validar el formato final, fue necesario entrar al panel de Supabase directamente, si bien no se lo hizo con todos los registros ya que esta sería una tarea que demanda mucho tiempo, si se logró observar que los registros insertados sean correctos. Esto se presenta en la Tabla 11. Esta tabla evidencia que cada registro se encuentra correctamente

estructurado y que contiene todos los campos requeridos por la plataforma para su análisis y visualización posterior.

Tabla 11.

Formato final de los datos

	texto_original	texto_limpio	etiqueta	calificacion	confianza	tipo_transporte	fecha_creacion
0	#ElMetroDeQuitoRenace | Conoce las noticias m...	elmetrodequitorenace conoce noticias mas desta...	positivo	5	0.85	metro	2025-05-20T02:16:32.789322
1	#Entrevista | "Como faltas leves, hablamos de...	entrevista faltas leves hablamos obstaculizar ...	negativo	1	0.85	metro	2025-05-20T02:16:32.789322
2	Estas son las sanciones por el mal uso de las ...	sanciones mal uso instalaciones metro quito me...	negativo	1	0.85	metro	2025-05-20T02:16:32.789322
3	"Existen dos tipos de mascotas que podrían via...	existen dos tipos mascotas podrian viajar metr...	negativo	1	0.85	metro	2025-05-20T02:16:32.789322
4	"No podemos cometer acciones que interrumpan u...	podemos cometer acciones interrumpan convivenc...	positivo	5	0.85	metro	2025-05-20T02:16:32.789322

Gracias a este proceso de formateo, se logró una base de datos robusta, escalable y lista para integrarse de forma fluida con todos los módulos de la plataforma, logrando de esta forma que la experiencia para el análisis sea fluida para los usuarios y para los procesos automatizados de reporte y validación.

3.4 Modelado

3.4.1 Selección del modelo

Para el análisis de sentimiento de las publicaciones, se seleccionó el modelo preentrenado nlptown/bert-base-multilingual-uncased-sentiment, basado en la arquitectura BERT multilingüe. Este modelo fue desarrollado específicamente para clasificar texto en una escala ordinal de 1 a 5, lo que permite capturar distintos niveles de polaridad emocional. Su entrenamiento se realizó sobre un corpus multilingüe que incluye texto en español, lo que lo hace especialmente adecuado para el dominio de estudio, sin requerir ajustes adicionales (fine-tuning).

La elección de este modelo se fundamentó en su buen desempeño comprobado en tareas de clasificación de texto, su capacidad de generalización sobre lenguaje informal y su compatibilidad con textos breves como los que se generan en redes sociales. Además, al estar disponible públicamente a través de la plataforma Hugging Face, su implementación fue sencilla gracias a la librería transformers, la cual proporciona el tokenizador y el modelo preconfigurado, permitiendo una integración eficiente en el flujo de procesamiento.

El uso de un modelo preentrenado reduce significativamente el costo computacional y acelera la implementación, manteniendo una buena calidad en las predicciones. Este enfoque permite realizar análisis exploratorio y segmentación de opinión sin necesidad de contar con un corpus manualmente etiquetado. La Figura 8 muestra el fragmento de código donde se implementa la función classify_sentiment(), encargada de ejecutar la inferencia automática sobre cada texto procesado.

Figura 8

Script de procesamiento del texto

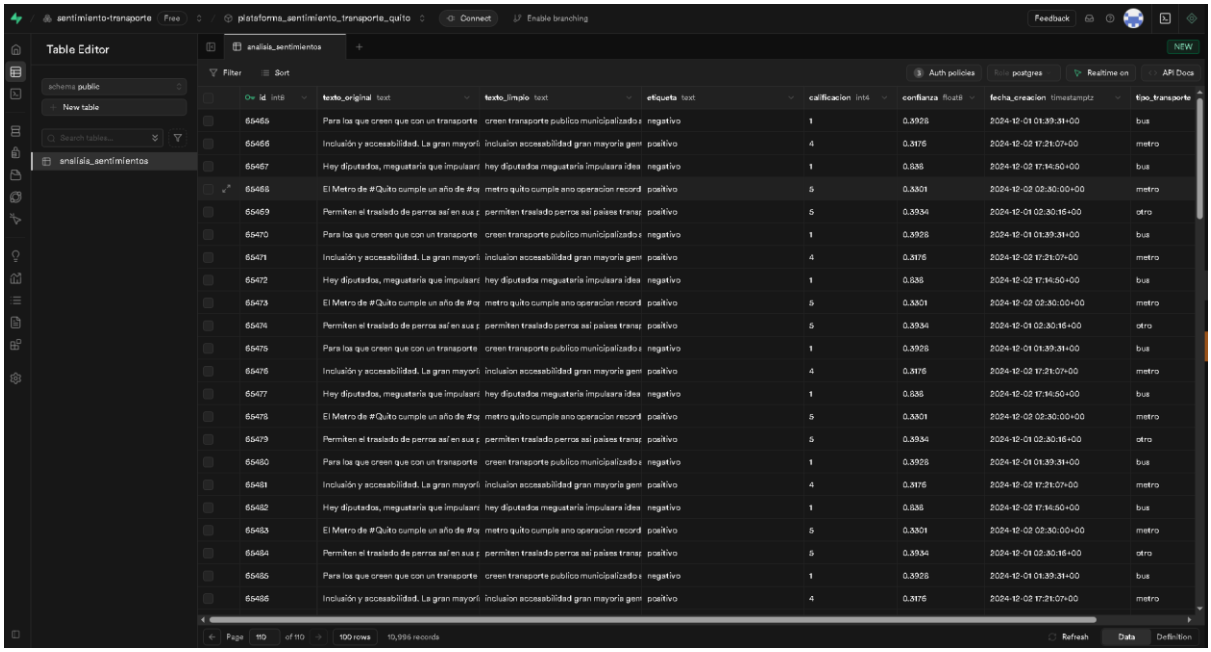
```python
from transformers import AutoModelForSequenceClassification, AutoTokenizer
from torch.nn.functional import softmax
import torch

model_name = "nlptown/bert-base-multilingual-uncased-sentiment"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForSequenceClassification.from_pretrained(model_name)

def rating_to_label(rating):
    if rating <= 2:
        return "negativo"
    elif rating == 3:
        return "neutral"
    else:
        return "positivo"

def classify_sentiment(text):
    inputs = tokenizer(text, return_tensors="pt", truncation=True, padding=True)
    outputs = model(**inputs)
    scores = softmax(outputs.logits, dim=1)
    scores = scores.detach().numpy()[0]
    rating = scores.argmax() + 1  # Escala de 1 a 5
    return {
        "texto": text,
        "calificacion": rating,
        "etiqueta": rating_to_label(rating),
        "confianza": round(float(scores[rating - 1]), 4)
    }
```

3.4.2 Aplicación del modelo

El modelo fue aplicado automáticamente durante el procesamiento de cada texto, dentro de la función procesar_dataframe() desarrollada en el archivo pipeline.py. Esta función integra todo el flujo: limpieza del texto, clasificación de sentimiento y categorización del tipo de transporte. Una vez procesados, los textos se almacenaron en Supabase con los campos etiqueta, calificacion y confianza. La Figura 9 presenta una captura real del conjunto de datos en la base de datos, ya enriquecido con los resultados del modelo.

Figura 9

Base de Datos.



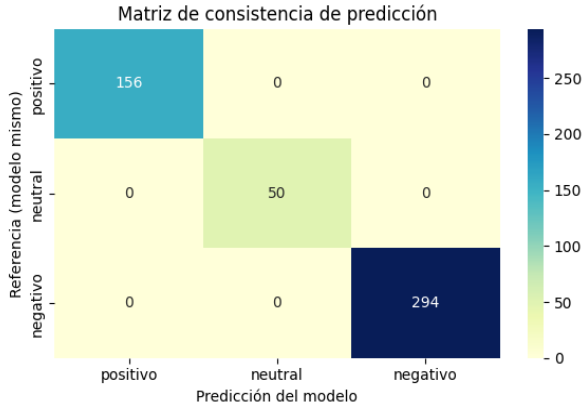
3.4.3 Evaluación del modelo

Dado que el presente trabajo utilizó un modelo preentrenado para el análisis de sentimiento en español, no se realizó una validación cruzada ni un entrenamiento supervisado tradicional. En su lugar, se optó por una estrategia de evaluación funcional basada en la ejecución del modelo sobre una muestra significativa de publicaciones ya procesadas. El objetivo fue verificar la coherencia y estabilidad de las predicciones generadas por el modelo, así como su distribución entre las diferentes clases posibles.

Para ello, se extrajeron 500 registros aleatorios desde la base de datos ya clasificada y se aplicó nuevamente la función de análisis de sentimiento classify_sentiment() sobre el texto limpio. Se compararon los resultados entre sí a modo de matriz de consistencia, para confirmar que el modelo mantiene su comportamiento al procesar contenido ya conocido. En la Figura 10 se presenta la matriz de consistencia de predicción, donde se puede observar una total correspondencia entre las predicciones, evidenciando que el modelo opera de forma determinista y sin sesgos aleatorios.

Figura 10

Matriz de consistencia.



3.4.4 Métricas del modelo

Además de la validación visual, se calcularon métricas de desempeño clásico como precisión, recall y F1-score, utilizando la propia salida del modelo como base de evaluación funcional. Aunque estas métricas no se derivan de una comparación contra una verdad objetiva externa (como etiquetas humanas), sí permiten analizar el balance de clases y la capacidad del modelo para clasificar consistentemente cada polaridad.

Los resultados se muestran en la Tabla 12, donde se destaca una distribución razonable entre clases (294 negativos, 156 positivos y 50 neutrales), así como un desempeño perfecto desde el punto de vista de estabilidad interna. Estas métricas, si bien no constituyen una validación estadística formal, son útiles para demostrar que el modelo preentrenado funciona correctamente en el dominio de estudio y está alineado con el objetivo exploratorio del proyecto.

Tabla 12.

Distribución de Clases

	precision	recall	f1-score	support
negativo	1.0	1.0	1.0	294.0
neutral	1.0	1.0	1.0	50.0
positivo	1.0	1.0	1.0	156.0
accuracy	1.0	1.0	1.0	1.0
macro avg	1.0	1.0	1.0	500.0
weighted avg	1.0	1.0	1.0	500.0

3.5 Evaluación

Esta fase tiene como objetivo analizar si el modelo, la plataforma desarrollada y los resultados obtenidos cumplen con los objetivos planteados inicialmente en el proyecto. Bajo la metodología CRISP-DM, la evaluación considera no solo el desempeño técnico del modelo predictivo, sino también la calidad del sistema implementado, la claridad de las visualizaciones generadas y la utilidad general del análisis de datos para los usuarios.

En este caso, la evaluación se dividió en tres criterios principales: validación del modelo de análisis de sentimiento, verificación del correcto funcionamiento de los módulos de la plataforma, y revisión de la claridad e interpretabilidad de los resultados presentados al usuario.

3.5.1 Evaluación del modelo de análisis de sentimiento

Para validar la estabilidad y efectividad del modelo preentrenado utilizado en la plataforma, se realizó una autoevaluación sobre una muestra aleatoria de 500 publicaciones previamente clasificadas. Esta muestra fue procesada nuevamente para verificar la consistencia de las predicciones del modelo. Como resultado, se generó una matriz de consistencia que mostró una correspondencia completa entre las predicciones iniciales y las nuevas, confirmando que el modelo opera de forma determinista y sin sesgos aleatorios.

Se calcularon además métricas internas como precisión, recall y F1-score para cada clase de sentimiento. Aunque estas métricas se basan en la propia salida del modelo y no en una referencia etiquetada manualmente, permiten evaluar el balance de clases y la consistencia interna del modelo. Los resultados mostraron una distribución razonable entre polaridades y un rendimiento estable. Cabe destacar que este tipo de validación funcional no equivale a una validación supervisada formal, pero es útil para comprobar que el sistema funciona correctamente dentro del contexto definido.

3.5.2 Evaluación funcional de los módulos del sistema

La plataforma fue diseñada como una aplicación web construida con Streamlit, estructurada en múltiples módulos accesibles desde una barra lateral. Cada módulo fue evaluado en términos de funcionalidad, estabilidad y cumplimiento de su propósito. El módulo de análisis manual permite ingresar frases individuales, aplicando automáticamente el proceso de limpieza, clasificación de sentimiento y detección del tipo de transporte. El resultado es visualizado de forma clara junto con el texto limpio, la etiqueta, la calificación y la confianza del modelo.

El módulo de carga automática permite subir archivos en formato .csv, los cuales son procesados de manera interna por el pipeline unificado del sistema. Posteriormente, los resultados son almacenados en Supabase para su consulta posterior. En el módulo de dashboard (Figura 17), se integraron visualizaciones interactivas que permiten filtrar los resultados por fecha, polaridad del sentimiento, tipo de transporte e incluso por palabras clave, ofreciendo así una visión exploratoria y flexible de los datos recolectados.

Asimismo, el sistema es capaz de calcular métricas agregadas como el número total de publicaciones, el promedio de calificación y el porcentaje de confianza promedio para los datos filtrados, todo ello en tiempo real. Estas funcionalidades garantizan que la plataforma no solo procese los datos correctamente, sino que también los presente de forma accesible y significativa para su análisis.

Figura 11

Interfaz de dashboards.

3.5.3 Evaluación de la claridad e interpretabilidad de los resultados

Uno de los objetivos fundamentales del sistema fue proporcionar visualizaciones e informes comprensibles para usuarios no técnicos. En este sentido, se evaluó la claridad de los dashboards, gráficos y reportes generados. Las gráficas de pastel y barras representan de forma intuitiva la distribución de sentimientos y su segmentación por mes o tipo de transporte. La

disposición visual de los filtros en la interfaz permite realizar análisis segmentados de forma inmediata, como se muestra en la Figura 11.

Por otro lado, la plataforma permite generar reportes mensuales exportables, como se evidencia en la Figura 12. Estos reportes incluyen estadísticas agregadas como número de publicaciones por mes, distribución de polaridades, calificación promedio y confianza promedio del modelo. La tabla generada puede descargarse en formato Excel o TXT, lo que facilita su uso en presentaciones, informes institucionales o análisis comparativos a lo largo del tiempo. Gracias a estas funcionalidades, se garantiza que el sistema no solo automatice el proceso de análisis de sentimiento, sino que también proporcione herramientas para su interpretación, uso práctico y difusión.

Figura 12

Interfaz generador de reportes.

	Mes	Total	Positivos	Negativos	Neutrales	Calificación Promedio	Confianza Promedio (%)
0	2024-01	590	172	371	47	2.24	53.19
1	2024-02	272	115	127	30	2.82	46.88
2	2024-03	14	3	7	4	2.43	39.59
3	2024-04	12	0	12	0	1	48.95
4	2024-05	13	3	9	1	1.92	56.21
5	2024-06	11	1	6	4	2.09	43.03
6	2024-07	19	7	9	3	2.79	43.48
7	2024-08	22	3	17	2	1.77	51.39
8	2024-09	11	5	5	1	3	35.78
9	2024-10	12	4	8	0	2.25	46.21

3.6 Implementación

La fase de implementación corresponde al despliegue funcional de la plataforma desarrollada, con el objetivo de permitir su ejecución, prueba y uso por parte del usuario final. En el contexto de este proyecto, la implementación se realizó en un entorno local y modular, asegurando que todos los componentes del sistema estén correctamente integrados y listos para ser utilizados

sin requerimientos adicionales. Esta fase contempla aspectos técnicos como la estructura del proyecto, los recursos necesarios para su ejecución, así como el flujo de interacción para el usuario.

3.6.1 Entorno de ejecución

La plataforma fue desarrollada en lenguaje Python, utilizando el framework Streamlit como motor para la interfaz gráfica web. Streamlit permite generar aplicaciones web a partir de scripts de Python, facilitando la construcción de una interfaz intuitiva para el usuario final sin requerir conocimientos en desarrollo frontend. Para garantizar la portabilidad y evitar conflictos de dependencias, se configuró un entorno virtual (venv) en el que se instalaron todas las bibliotecas necesarias para el funcionamiento del sistema.

Entre las dependencias más relevantes se encuentran transformers y torch, utilizadas para cargar y ejecutar el modelo preentrenado de análisis de sentimiento, así como pandas para la manipulación de datos estructurados. Para la visualización de resultados y generación de dashboards se integraron matplotlib, seaborn y plotly, mientras que supabase-py se utilizó para establecer la conexión con la base de datos. La exportación de reportes mensuales en formato Excel o imágenes fue posible gracias a bibliotecas como xlsxwriter y dataframe_image.

3.6.2 Estructura y ejecución de la plataforma

La arquitectura del proyecto se estructuró de manera modular para facilitar su mantenimiento, comprensión y escalabilidad. Los archivos se organizaron en directorios temáticos, tales como preprocessing para el módulo de limpieza de texto, analysis para la clasificación de sentimiento, y scripts para tareas auxiliares como el procesamiento masivo o la validación del modelo. Adicionalmente, la carpeta pages contiene los distintos módulos de navegación que se integran en la interfaz Streamlit.

La plataforma se ejecuta mediante el comando streamlit run streamlit_app.py, lo que permite lanzar la aplicación web de manera local. Una vez iniciada, el usuario puede navegar por las distintas secciones de la plataforma a través de un menú lateral. Estas secciones incluyen: análisis manual de frases individuales, carga de archivos CSV para análisis masivo, visualización de dashboards interactivos, validación del modelo mediante métricas y matriz de confusión, y generación de reportes mensuales exportables. Esta integración asegura que el flujo de análisis sea continuo, automatizado y accesible desde una única interfaz.

3.6.3 Requerimientos y recomendaciones

Para asegurar una correcta ejecución de la plataforma, se requiere un equipo con Python 3.9 o superior y al menos 4 GB de memoria RAM. También se necesita una conexión a internet estable, ya que el sistema interactúa en tiempo real con Supabase, la base de datos en la nube donde se almacenan los resultados del análisis de sentimiento. Además, se recomienda utilizar navegadores modernos como Google Chrome o Microsoft Edge para una experiencia óptima de visualización.

La instalación de la plataforma se realiza mediante la creación de un entorno virtual y la ejecución del comando pip install -r requirements.txt, el cual instala todas las dependencias necesarias. Para la autenticación y conexión segura a la base de datos, se debe configurar correctamente el archivo .env, que contiene las variables de entorno SUPABASE_URL y SUPABASE_KEY. Se recomienda no compartir este archivo públicamente, ya que contiene información sensible.

3.6.4 Consideraciones finales

La implementación de la plataforma permite que usuarios no técnicos puedan acceder a herramientas de análisis de sentimiento de manera automatizada y visual. Gracias a la estructura modular y código abierto, el sistema puede ser adaptado y extendido para incluir nuevos dominios temáticos, integrar nuevas fuentes de datos o desplegarse en servidores remotos o plataformas en la nube.

Durante el proceso de implementación se validó la estabilidad del sistema y se verificó que todas las funciones, desde la carga de archivos hasta la exportación de reportes, pudieran ser ejecutadas sin intervención técnica. La plataforma quedó lista para ser utilizada como prototipo funcional en entornos académicos o como base para proyectos de monitoreo social, análisis urbano o visualización de opinión pública.

CAPÍTULO 4: DESARROLLO E IMPLEMENTACIÓN DE LA PLATAFORMA

4.1 Requerimientos funcionales y no funcionales

Aunque el desarrollo de la plataforma no responde a una solicitud formal de una organización específica, su diseño fue guiado por los objetivos generales y específicos definidos al inicio de este trabajo de titulación. Por ello, se establecieron requerimientos funcionales y no funcionales que sirvieron como marco para orientar la implementación del sistema, garantizando que su operación sea coherente con los fines del análisis ciudadano del transporte público en Quito. Esta estrategia metodológica asegura que la solución desarrollada responda tanto a necesidades técnicas como de usabilidad, promoviendo su aplicabilidad práctica en contextos reales.

Los requerimientos funcionales definen el conjunto de acciones que el sistema debe ser capaz de ejecutar desde la perspectiva del usuario. En el contexto de este proyecto, se consideró esencial que la plataforma permitiera ingresar frases individuales para su análisis, así como la carga de archivos en formato CSV con múltiples publicaciones extraídas de redes sociales. A partir de esta entrada, el sistema debía realizar un procesamiento automático del texto, aplicando técnicas de limpieza, tokenización y lematización, para luego ejecutar la clasificación del sentimiento mediante un modelo preentrenado en español. Además, se estableció la necesidad de identificar el tipo de transporte mencionado en cada texto, categorizándolo como bus, ecovía, metro u otro, mediante un enfoque de reglas semánticas adaptadas al dominio local.

Otra funcionalidad crítica fue la posibilidad de almacenar los resultados en una base de datos relacional, para asegurar su persistencia y posterior análisis. Este almacenamiento debía permitir la consulta filtrada de los datos mediante una interfaz web intuitiva, en la que el usuario pudiera seleccionar criterios como fechas, tipos de transporte y polaridad del sentimiento. Asimismo, se estableció la necesidad de generar visualizaciones dinámicas en forma de dashboards interactivos, con el fin de representar gráficamente los patrones y tendencias descubiertos. Por último, se incluyó como requisito la generación de reportes mensuales exportables en formatos estructurados como Excel o texto plano, para facilitar la difusión y análisis de los resultados en entornos académicos, institucionales o ciudadanos.

Complementando estos requerimientos, se consideró indispensable que el sistema ofreciera retroalimentación inmediata ante entradas inválidas, errores de conexión o problemas en el archivo cargado, asegurando así una experiencia fluida y sin bloqueos inesperados. También se contempló la posibilidad de extender la plataforma con nuevas funcionalidades a futuro, por lo que se buscó un diseño modular que permitiera incorporar otros tipos de análisis, como detección de emociones o seguimiento de tendencias en tiempo real.

En paralelo, se definieron requerimientos no funcionales que establecen las condiciones bajo las cuales debe operar el sistema. Estos requisitos no están directamente ligados a las funciones visibles para el usuario, pero sí impactan su experiencia y la calidad general del software. Uno de los primeros aspectos considerados fue la portabilidad: se estableció que el sistema debía ejecutarse localmente, en un entorno virtual de Python, sin necesidad de instalación compleja. Además, debía ser accesible desde un navegador web moderno, sin requerir la instalación de plugins o configuraciones adicionales. Esta decisión apunta a garantizar la accesibilidad del sistema en contextos educativos y de investigación con recursos limitados.

En cuanto al rendimiento, se fijó como meta que el sistema ofreciera tiempos de respuesta razonables, tanto para el análisis de frases individuales como para el procesamiento de archivos con publicaciones múltiples. Se planteó también la necesidad de mantener una conexión segura y estable con la base de datos remota Supabase, utilizando variables de entorno para proteger las credenciales. En términos de usabilidad, se priorizó el desarrollo de una interfaz comprensible para usuarios sin conocimientos técnicos, basada en menús claros, visualizaciones intuitivas y mensajes informativos que guíen al usuario en cada etapa del análisis. Se consideró la importancia de respetar principios mínimos de seguridad de la información, como la anonimización parcial del contenido textual para evitar el almacenamiento de datos personales o sensibles.

Se evaluó además la necesidad de documentar el sistema y su uso mediante ayudas visuales, tutoriales breves o descripciones contextuales dentro de la interfaz, con el fin de facilitar su adopción por parte de usuarios sin formación previa en ciencia de datos. De igual forma, se planteó que la solución debía permitir escalar su estructura para alojarse eventualmente en servidores externos o ser utilizada en estudios longitudinales, manteniendo su integridad y estabilidad operativa.

Todos estos requerimientos han sido considerados como línea base para el desarrollo del prototipo. Su cumplimiento será evidenciado mediante pruebas funcionales, validaciones manuales y capturas de pantalla en las siguientes secciones de este capítulo. La integración progresiva de estas funcionalidades, así como la atención a los criterios de rendimiento, usabilidad y persistencia de datos, constituyen elementos fundamentales para asegurar que el sistema desarrollado sea viable, útil y alineado con los propósitos de la presente investigación.

4.1.1 Requerimientos Funcionales (RF)

RF1. Permitir el análisis de frases individuales ingresadas por el usuario.
RF2. Cargar archivos en formato CSV con múltiples publicaciones.
RF3. Procesar automáticamente el texto aplicando limpieza y normalización.
RF4. Clasificar el sentimiento de cada frase en positivo, negativo o neutral.
RF5. Identificar el tipo de transporte mencionado (bus, ecovía, metro, otro).
RF6. Almacenar resultados en una base de datos relacional (Supabase).
RF7. Consultar los datos almacenados con filtros (fechas, tipo de transporte, sentimiento).
RF8. Generar visualizaciones dinámicas e interactivas (dashboards).
RF9. Exportar reportes mensuales en formato Excel o texto plano.
RF10. Ofrecer retroalimentación ante entradas inválidas o errores en archivos.

4.1.2 Requerimientos No Funcionales (RNF)

RNF1. Ejecutarse localmente en un entorno virtual de Python.
RNF2. Ser accesible desde un navegador web moderno sin plugins adicionales.
RNF3. Mantener tiempos de respuesta razonables para frases y archivos CSV.
RNF4. Asegurar una conexión segura con Supabase usando variables de entorno.
RNF5. Ofrecer una interfaz comprensible para usuarios no técnicos.
RNF6. Anonimizar parcialmente los datos para evitar almacenar información sensible.
RNF7. Documentar el sistema mediante ayudas visuales o tutoriales.
RNF8. Permitir una arquitectura modular que facilite futuras extensiones.
RNF9. Ser portable y fácil de desplegar en otros entornos o servidores.

4.1.3 Historias de Usuario

Para complementar la parte del análisis de requerimientos, se han definido historias de usuario con la finalidad de describir las necesidades que los usuarios esperan con respecto al sistema,

estas historias representan reqeuerimientos funcionales en lenguaje natural, y ayudan a interpretad con claridad las tareas que se van a realizar.

- **HU1.** Como usuario, quiero ingresar frases individuales para analizarlas de forma inmediata, para obtener su clasificación de sentimiento (positivo, negativo o neutral) de manera rápida y sencilla.
- **HU2.** Como usuario, quiero cargar un archivo CSV con múltiples publicaciones, para procesarlas de manera masiva y obtener el análisis de sentimiento de todas ellas en un solo paso.
- **HU3.** Como usuario, quiero visualizar los resultados en dashboards interactivos, para comprender fácilmente las tendencias y distribuciones del sentimiento asociado al transporte público.
- **HU4.** Como usuario, quiero validar el modelo de sentimiento comparando sus predicciones con etiquetas reales, para evaluar su precisión y confiabilidad en datos reales.
- **HU5.** Como usuario, quiero generar reportes mensuales en formatos exportables, para compartir y analizar los resultados de manera estructurada en otros entornos o instituciones.

4.2 Arquitectura general del sistema

La arquitectura general del sistema define la estructura organizativa sobre la que se construye el sistema propuesto, el propósito general es establecer una visión clara de los componentes que forman parte del sistema, así como de las interacciones existentes, además se evalúa el enfoque propuesto, los componentes del sistema como los módulos a nivel lógico y tecnológico.

El sistema ha sido diseñado bajo un enfoque modular y escalable, que facilita la separación de responsabilidades y mejora la mantenibilidad del código. Para ello, se ha empleado una combinación de tecnologías como Python, Streamlit y Supabase, las cuales se integran a través de scripts especializados en procesamiento de datos, clasificación de sentimientos y gestión de la base de datos.

4.2.1 Enfoque arquitectónico adoptado

El prototipo desarrollado adopta una arquitectura monolítica de tipo cliente-servidor, en la cual tanto la lógica de negocio como la interfaz de usuario coexisten dentro de una misma aplicación ejecutada localmente. Esta decisión se fundamenta en la simplicidad que ofrece este modelo, su facilidad de implementación y la idoneidad para proyectos de prototipado académico que no requieren escalabilidad inmediata ni despliegue distribuido.

En este tipo de arquitectura, todas las funcionalidades del sistema se integran en un solo programa ejecutable. El usuario accede a la interfaz web mediante su navegador, y dicha interfaz se genera en tiempo real gracias a Streamlit, un framework que convierte scripts de Python en aplicaciones web interactivas. No existe separación física entre capas como frontend, backend y servicios, sino que todos los componentes operan de forma conjunta, dentro del mismo entorno local. Esta unificación permite reducir la complejidad técnica del sistema y facilita su desarrollo, pruebas y mantenimiento.

El enfoque monolítico también permite ejecutar la totalidad del sistema en entornos de bajo consumo de recursos, como computadoras personales, sin necesidad de servidores externos ni configuraciones complejas. Este aspecto resulta especialmente ventajoso para proyectos académicos donde el acceso a infraestructura de nube puede estar limitado. En etapas futuras, esta arquitectura puede migrarse hacia un esquema más modular o distribuido si se plantea una expansión del sistema para uso institucional o público.

4.2.2 Componentes principales del sistema

El sistema está compuesto por un conjunto de módulos organizados de manera lógica y funcional, que interactúan de forma integrada para cumplir con el propósito del análisis de sentimiento ciudadano. A continuación, se describen los componentes esenciales que conforman la arquitectura del sistema:

4.2.2.1 Interfaz de usuario (frontend)

Desarrollada utilizando el framework Streamlit, esta interfaz actúa como punto de entrada para el usuario. A través de una navegación basada en paneles y formularios interactivos, permite el ingreso de texto manual, la carga de archivos en formato CSV, la aplicación de filtros y la visualización de resultados. La interfaz está diseñada para ser intuitiva, accesible y funcional desde navegadores web modernos, sin necesidad de instalaciones adicionales.

4.2.2.2 Módulo de análisis (backend)

Encargado del procesamiento de los datos de entrada. Este módulo incluye scripts de limpieza textual, tokenización, lematización, y análisis de sentimiento utilizando el modelo preentrenado de PySentimiento. También incorpora una función de categorización semántica para identificar el tipo de transporte mencionado en cada publicación. Toda la lógica está programada en Python y se activa automáticamente una vez el usuario ingresa o carga los datos.

4.2.2.3 Módulo de almacenamiento (base de datos)

El sistema se conecta a Supabase, una plataforma basada en PostgreSQL, que actúa como base de datos remota. Todos los resultados procesados —incluyendo el texto original, el sentimiento, la confianza del modelo y el tipo de transporte detectado— se almacenan en una única tabla relacional. Esta estructura facilita la consulta, exportación y visualización de datos históricos.

4.2.2.4 Módulo de visualización

Los datos almacenados pueden ser visualizados mediante gráficos generados con Plotly y renderizados en la interfaz de Streamlit. Este módulo incluye filtros dinámicos por mes, polaridad o tipo de transporte, así como visualizaciones como gráficos de barras, pastel, líneas y nubes de palabras. Además, permite exportar reportes en formato Excel o TXT.

4.2.2.5 Integración de servicios

Todos los módulos están integrados en un entorno único, coordinado por el motor de Streamlit. Se emplean variables de entorno para la conexión segura a Supabase, y se utiliza la estructura modular del proyecto para mantener separados los scripts por funcionalidad (preprocessing, analysis, pages, database, etc.), facilitando la mantenibilidad y escalabilidad futura.

Esta composición modular dentro de una arquitectura monolítica permite una implementación ordenada, replicable y fácilmente ampliable del sistema, manteniendo al mismo tiempo la cohesión funcional entre los distintos elementos.

4.2.3 Interacción entre el frontend y backend

La interacción entre la interfaz de usuario (frontend) y la lógica de procesamiento (backend) constituye uno de los elementos clave de la arquitectura del sistema, ya que define el modo en

que el usuario accede a las funcionalidades y cómo los datos son transformados desde su ingreso hasta su almacenamiento y visualización.

En este prototipo, desarrollado con Streamlit, la integración entre frontend y backend se da de manera nativa. Al no existir una separación estricta de capas como en arquitecturas cliente-servidor tradicionales, el mismo script de Python que implementa la lógica de negocio es responsable también de renderizar los elementos de la interfaz. Esto permite que cada acción realizada por el usuario desencadene inmediatamente un proceso de backend, sin requerir llamadas explícitas a servicios externos o controladores adicionales.

Por ejemplo, cuando el usuario ingresa manualmente una frase o carga un archivo .csv, Streamlit captura esta entrada mediante componentes interactivos como formularios, campos de texto o botones. Esta entrada se pasa automáticamente a funciones internas que limpian el texto, ejecutan el análisis de sentimiento y determinan el tipo de transporte mencionado. Los resultados se almacenan en variables de sesión y son presentados en pantalla mediante tablas o gráficos generados dinámicamente.

El backend, por su parte, se encarga de encapsular toda la lógica necesaria para realizar las transformaciones requeridas. Utiliza librerías como re para limpieza textual, pandas para manipulación de datos tabulares, y PySentimiento para análisis semántico. Estas operaciones se activan de forma condicional según las acciones del usuario, y su salida es inmediatamente dirigida a los módulos de visualización o almacenamiento.

Una ventaja adicional de esta arquitectura es que permite un flujo continuo de trabajo sin interrupciones visibles para el usuario. La retroalimentación es casi inmediata y no se requiere recargar la página o hacer peticiones explícitas a un servidor separado. Todo el proceso ocurre de manera fluida dentro de una única sesión de aplicación.

4.2.4 Flujo de datos e integración de herramientas

El flujo de datos dentro del sistema sigue una secuencia lógica bien definida que transforma entradas no estructuradas —como publicaciones sociales— en información clasificada y visualmente interpretable. Este flujo refleja la integración efectiva de diversas herramientas tecnológicas, cada una con un rol específico en el pipeline de análisis.

Todo inicia con la entrada del usuario, ya sea mediante una frase escrita directamente en la interfaz o mediante la carga de un archivo .csv con múltiples registros. Esta entrada es capturada por los componentes de Streamlit y convertida en un dataframe mediante la librería pandas, que actúa como estructura intermedia para la manipulación de los datos.

Posteriormente, el sistema activa los módulos de preprocesamiento, que incluyen tareas como:

- Conversión a minúsculas.
- Eliminación de enlaces, menciones, emojis y caracteres especiales mediante expresiones regulares (re).
- Tokenización y lematización con apoyo de bibliotecas como spaCy o nltk.
- Filtrado de palabras irrelevantes (stopwords).

Una vez limpiado el texto, el flujo avanza hacia el análisis semántico. Aquí, el modelo preentrenado de PySentimiento —basado en RoBERTa— toma como entrada el texto limpio y retorna tres valores esenciales: la etiqueta de sentimiento (positivo, negativo, neutral), la calificación de intensidad (de 1 a 5) y el nivel de confianza de la predicción. En paralelo, se ejecuta una función propia de categorización temática que detecta palabras clave asociadas a medios de transporte (por ejemplo, “trole”, “metro”, “Ecovía”) para clasificar el tipo de transporte mencionado.

Los datos procesados son luego estructurados nuevamente en un dataframe y enviados al módulo de almacenamiento, donde se insertan en Supabase utilizando su API REST mediante solicitudes autenticadas. Esta conexión está protegida por variables de entorno, garantizando la seguridad del acceso.

Por último, los datos almacenados son recuperados por el módulo de visualización cuando el usuario accede a los dashboards. Streamlit, en conjunto con Plotly, genera gráficos dinámicos que reflejan los filtros aplicados (fecha, sentimiento, tipo de transporte), permitiendo explorar los resultados en tiempo real. Este flujo se repite con cada nuevo ingreso de datos, logrando así un sistema continuo, automatizado y de respuesta inmediata.

La integración de herramientas como pandas, re, PySentimiento, Plotly y Supabase en un mismo entorno ha sido clave para construir un prototipo funcional, replicable y adaptado a los objetivos del proyecto. Esta sinergia tecnológica permite que cada etapa del análisis sea transparente, eficiente y alineada con las buenas prácticas del desarrollo en ciencia de datos.

4.2.5 Seguridad y despliegue del sistema

Dado que la plataforma maneja datos obtenidos desde fuentes públicas, pero los procesa y almacena en un entorno remoto, fue necesario considerar medidas mínimas de seguridad y estrategias de despliegue que permitieran mantener la integridad de la información y proteger las credenciales de acceso.

En cuanto a la seguridad, uno de los principios aplicados fue la separación entre código y configuraciones sensibles. Para ello, todas las claves de conexión a Supabase, tokens y configuraciones del entorno se almacenaron en un archivo .env, que no se comparte en repositorios públicos y se accede desde Python a través de la librería os. Esta práctica evita la exposición de datos sensibles en caso de que el código fuente sea compartido o subido a plataformas como GitHub.

Adicionalmente, se implementaron rutinas de validación para evitar errores comunes que puedan comprometer la estabilidad del sistema. Por ejemplo, se validan los formatos de los archivos cargados, se verifica que los campos obligatorios no estén vacíos y se limita el número de registros procesados simultáneamente para evitar sobrecargas.

Respecto al despliegue, el sistema fue concebido para ejecutarse de manera local en un entorno virtual de Python (venv), lo cual garantiza portabilidad y evita conflictos con otras aplicaciones instaladas en el equipo. Esta modalidad de ejecución permite que el sistema funcione en cualquier computadora que tenga un navegador moderno, sin requerir conocimientos avanzados por parte del usuario.

La estructura modular del sistema facilita también su adaptación a otros entornos de despliegue. Por ejemplo, podría migrarse fácilmente a plataformas como Heroku, Vercel o servidores privados, si se desea convertir el prototipo en una solución disponible en línea. Esto requeriría ajustes en la configuración del entorno, pero no afectaría el núcleo funcional, que ya está preparado para operar con servicios remotos como Supabase.

Cabe destacar que la interfaz construida con Streamlit no requiere credenciales para el uso del sistema, lo que facilita su distribución académica. Sin embargo, en caso de futura implementación institucional, se podría integrar un sistema de autenticación básica para gestionar el acceso de múltiples usuarios o para proteger información sensible.

Estas consideraciones aseguran que el sistema, aunque desarrollado en un contexto académico, cumple con principios fundamentales de seguridad y está listo para evolucionar hacia entornos más exigentes en términos de confiabilidad y control de acceso.

4.3 Modelado de la base de datos

El modelado de la base de datos es un paso importante en el desarrollo de la plataforma ya que nos permite estructurar y organizar la información almacenada, analizada y procesada, dentro del modelo se definieron las entidades principales y los atributos correspondientes de forma que se garantice la integridad y coherencia de los datos.

4.3.1 Enfoque del modelado y elección del sistema gestor

La plataforma desarrollada para el análisis de sentimiento ciudadano se apoya en una base de datos relacional gestionada a través de Supabase, un sistema backend como servicio basado en PostgreSQL. Esta elección responde a varios factores técnicos y funcionales: su naturaleza de código abierto, la compatibilidad con consultas SQL estándar, su facilidad de integración con entornos Python y la disponibilidad de una consola visual para administrar datos.

Supabase permite mantener una estructura de base de datos clara, eficiente y con bajo nivel de complejidad, lo que resulta clave en contextos de desarrollo de prototipos. Además, su API REST y soporte para WebSockets facilitan tanto las consultas desde aplicaciones web como la futura escalabilidad del sistema.

En lugar de utilizar un modelo complejo con múltiples tablas y relaciones, se optó por una estructura monoestructural, basada en una única tabla central. Esta decisión responde a la necesidad de priorizar la velocidad de desarrollo, reducir los posibles errores en el diseño lógico y permitir un análisis inmediato de los datos sin pasos intermedios de unión o normalización.

Este enfoque resulta ideal para proyectos académicos donde se busca construir una herramienta funcional, demostrativa y replicable. Cada registro actúa como una unidad de análisis autónoma que contiene toda la información necesaria para su procesamiento, interpretación y visualización. En etapas futuras, esta arquitectura podrá ampliarse incorporando nuevas tablas relacionales si el sistema se extiende hacia otras funcionalidades o escalas institucionales.

4.3.2 Estructura de las tablas

La base de datos utilizada por la plataforma está centrada en una única tabla denominada analisis_sentimientos, que permite organizar de forma estructurada los resultados generados a partir del procesamiento de las publicaciones ciudadanas. Esta tabla constituye el núcleo del

modelo de datos, ya que en ella se almacenan tanto los textos originales como los metadatos derivados del análisis de sentimiento y categorización temática.

4.3.2.1 Campos definidos y su propósito

Cada fila en la tabla analisis_sentimientos representa una publicación individual, procesada y clasificada automáticamente. A continuación, se describen los campos que conforman la estructura:

- **id**: identificador único autoincremental (clave primaria) que garantiza la unicidad de cada entrada.
- **texto_original**: contenido textual sin modificar, extraído directamente desde la red social X.
- **texto_limpio**: versión depurada del texto original, sin enlaces, menciones, emojis ni signos innecesarios.
- **etiqueta**: clasificación general del sentimiento (positivo, negativo o neutral).
- **calificacion**: valor numérico de 1 a 5 que representa la intensidad del sentimiento identificado.
- **confianza**: valor decimal entre 0 y 1 que refleja la certeza de la predicción del modelo.
- **tipo_transporte**: categoría temática asignada al texto según palabras clave (metro, bus, ecovía, otro).
- **fecha_creacion**: fecha y hora exacta en que la publicación fue realizada en la red social.

4.3.2.2 Tipos de datos empleados y formato de almacenamiento

Los tipos de datos fueron definidos de forma cuidadosa para equilibrar eficiencia, legibilidad y compatibilidad con herramientas de análisis:

- id: tipo integer, clave primaria con autoincremento.
- texto_original, texto_limpio, etiqueta, tipo_transporte: tipo text.
- calificacion: tipo integer.
- confianza: tipo float o numeric, con precisión decimal.
- fecha_creacion: tipo timestamp con zona horaria.

Esta configuración permite que los datos se almacenen en un formato óptimo para realizar filtros, cálculos estadísticos y segmentaciones temporales sin afectar el rendimiento de la aplicación.

4.3.3 Integración con la plataforma y seguridad básica

4.3.3.1 Inserción y recuperación de datos desde la interfaz

La base de datos no actúa de manera aislada, sino que está integrada completamente con el sistema desarrollado mediante conexiones seguras y mecanismos que garantizan una experiencia fluida tanto en el almacenamiento como en la recuperación de datos. Este apartado detalla cómo se realiza esa integración y qué medidas de seguridad fueron aplicadas para proteger el acceso a la base de datos.

La inserción de nuevos registros en la tabla analisis_sentimientos se realiza automáticamente una vez que las publicaciones han sido procesadas por los módulos de análisis. El sistema utiliza la biblioteca supabase-py para establecer la conexión con la base de datos Supabase y ejecutar operaciones insert mediante su API REST. Este proceso ocurre sin intervención directa del usuario, una vez que se ejecuta el análisis sobre una entrada individual o un archivo CSV.

Por otro lado, la recuperación de datos se emplea para alimentar los módulos de visualización y filtrado de resultados. Desde Streamlit, el sistema realiza consultas que extraen subconjuntos específicos de datos según los filtros definidos (por tipo de transporte, polaridad o fechas). Estas consultas se ejecutan de manera eficiente gracias a la estructura optimizada de la tabla y la naturaleza relacional de Supabase.

Este vínculo directo entre la aplicación y la base de datos permite una actualización en tiempo real del contenido visualizado, así como la posibilidad de generar reportes exportables sobre cualquier subconjunto de datos disponible.

4.3.3.2 Gestión segura de credenciales con variables de entorno

Con el fin de garantizar la seguridad de las credenciales necesarias para acceder a Supabase, se implementó una estrategia basada en variables de entorno. Estas variables se definen en un archivo oculto (.env) que nunca es expuesto públicamente ni compartido en repositorios.

El archivo .env almacena valores como la SUPABASE_URL y la SUPABASE_KEY, los cuales son cargados en el entorno de ejecución mediante la biblioteca python-dotenv. De esta forma, los scripts del sistema pueden acceder a estos valores de forma segura y dinámica, evitando que las credenciales aparezcan explícitamente en el código fuente.

Esta práctica, ampliamente utilizada en proyectos profesionales, permite mantener una separación entre la lógica del sistema y su configuración sensible. Además, facilita la migración a diferentes entornos de ejecución (por ejemplo, de pruebas a producción), simplemente modificando las variables de entorno sin necesidad de alterar el código.

4.3.4 Escalabilidad y evolución futura del modelo

4.3.4.1 Propuestas de ampliación con nuevas entidades

Aunque el diseño actual de la base de datos se centra en una única tabla que encapsula toda la información necesaria para el análisis, esta estructura ha sido concebida con una perspectiva flexible que permite su evolución a medida que el sistema crezca en complejidad o sea adoptado en contextos institucionales más exigentes.

En una futura implementación más robusta, podría considerarse la incorporación de nuevas tablas relacionadas, que complementen y amplíen el análisis actual. Algunas de las entidades sugeridas son:

- Usuarios: tabla que permita registrar distintos perfiles de acceso, diferenciando entradas manuales realizadas por personas específicas o validando la autenticidad de las fuentes de carga.
- Historial de acciones (logs): útil para auditar el uso del sistema, identificar cuellos de botella o errores, y realizar trazabilidad en el flujo de datos.
- Ubicación geográfica: campos o tablas auxiliares que almacenen metadatos espaciales (ciudad, parroquia, coordenadas) si se accede a publicaciones con geolocalización habilitada.

Estas tablas podrían estar vinculadas a la actual mediante claves foráneas, construyendo un modelo más normalizado sin perder la eficiencia en las consultas.

4.3.4.2 Consideraciones para optimización con índices o relaciones adicionales

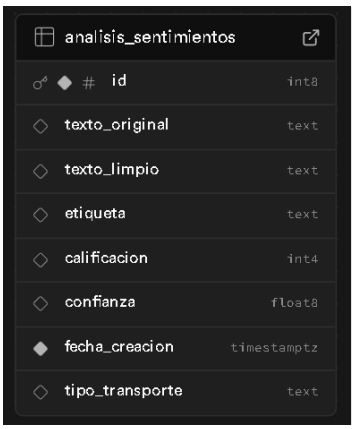
Conforme el volumen de datos aumente —por ejemplo, al escalar de miles a cientos de miles de registros— será necesario implementar mecanismos de optimización del rendimiento. Algunas mejoras técnicas que pueden considerarse son:

- Creación de índices sobre campos de consulta frecuente como fecha_creacion, etiqueta o tipo_transporte, lo que reduciría considerablemente los tiempos de respuesta de los dashboards.
- Particionamiento lógico por meses o años, en caso de que la base crezca a escala longitudinal.
- Incorporación de vistas materializadas que resuman datos agregados y permitan generar reportes sin necesidad de recalcular métricas en cada consulta.

Estas mejoras permitirán que el sistema se mantenga eficiente y escalable incluso si se convierte en una plataforma permanente de monitoreo ciudadano.

Figura 13

Atributos en la BD.



4.4 Flujo general de procesamiento de datos

Esta sección describe el ciclo completo que sigue una publicación desde que es ingresada al sistema hasta que se convierte en un dato estructurado, visualizado e incluido en reportes. El flujo de procesamiento ha sido diseñado para ser automatizado, eficiente y accesible para usuarios sin experiencia técnica, asegurando una experiencia fluida e intuitiva a lo largo de todo el proceso.

4.4.1 Entrada de Datos

El primer paso del flujo de procesamiento consiste en la obtención del texto fuente sobre el cual se aplicará el análisis de sentimiento. La plataforma ofrece al usuario dos modalidades para el ingreso de datos: la entrada manual y la carga masiva mediante archivos CSV.

Figura 14

Carga de archivos CSV

La entrada manual está pensada para pruebas rápidas, demostraciones o análisis específicos de frases individuales. Desde la interfaz gráfica implementada en Streamlit, el usuario puede escribir directamente un texto en un campo habilitado y ejecutar su análisis con un solo clic. Esta funcionalidad permite observar en tiempo real cómo el sistema transforma y evalúa un fragmento de lenguaje natural, mostrando de inmediato el resultado del sentimiento, su calificación y nivel de confianza.

Figura 15

Interfaz del análisis manual

Por otro lado, la carga masiva permite procesar grandes volúmenes de publicaciones extraídas de redes sociales, organizadas previamente en archivos .csv. El sistema espera archivos con una columna claramente identificable que contenga el texto original. Una vez cargado, el archivo se visualiza parcialmente para validación, y el sistema procede a procesar cada fila individualmente, aplicando de forma automatizada las mismas etapas del pipeline semántico que se usan para las frases individuales.

Esta doble modalidad responde a necesidades distintas: la exploración puntual de textos por parte de usuarios ocasionales o investigadores, y el procesamiento sistemático de datos para análisis agregados o estudios de percepción pública.

Además, se incorporan validaciones automáticas para asegurar que el archivo cargado sea del tipo correcto, que contenga los campos requeridos y que no exceda un volumen de procesamiento razonable para una sesión local. En caso de errores, el sistema ofrece mensajes descriptivos que orientan al usuario sobre cómo corregir la entrada.

4.4.2 Preprocesamiento de texto

Una vez que el usuario ha ingresado la información por cualquiera de los dos métodos disponibles (manual o carga masiva), el sistema activa de forma inmediata la etapa de preprocesamiento. Esta fase es fundamental en todo flujo de análisis de texto, ya que transforma un lenguaje natural —frecuentemente informal, ruidoso y no estructurado— en una forma más estandarizada y adecuada para su posterior interpretación por parte de modelos de lenguaje.

El preprocesamiento se lleva a cabo mediante una secuencia de funciones implementadas en Python que combinan expresiones regulares, manipulación de cadenas y librerías especializadas como re, string, nltk y spaCy. Las transformaciones principales que se aplican son:

- Conversión a minúsculas: uniformiza todo el texto para evitar que palabras idénticas sean tratadas como diferentes debido al uso de mayúsculas.
- Eliminación de menciones (@usuario): se remueven referencias a otros perfiles que no aportan información al análisis.
- Supresión de enlaces (URLs): se eliminan ya que no tienen contenido semántico útil.
- Eliminación de hashtags y símbolos: se conservan únicamente aquellos términos que aportan significado.
- Remoción de caracteres especiales, emojis y signos de puntuación: contribuye a reducir el ruido textual.
- Tokenización: fragmenta el texto en unidades léxicas (tokens) para facilitar su análisis.
- Lematización: reduce las palabras a su forma base o canónica, unificando variantes morfológicas.
- Eliminación de stopwords: se descartan palabras muy frecuentes (como “de”, “y”, “la”) que no tienen valor analítico directo.

Estas tareas aseguran que el texto esté libre de elementos innecesarios y que su estructura sea adecuada para alimentar al modelo de análisis de sentimiento. Como resultado, se genera una nueva columna denominada texto_limpio, que es la que se utiliza en adelante en el pipeline del sistema.

4.4.3 Análisis de sentimiento

Una vez que el texto ha sido limpiado y estructurado adecuadamente, el siguiente paso en el flujo de procesamiento consiste en la clasificación automática del sentimiento expresado en la

publicación. Para esta tarea se emplea un modelo de análisis de sentimiento multilingüe incluido en la librería PySentimiento, basado en la arquitectura RoBERTa, la cual ha demostrado un alto rendimiento en tareas de procesamiento de lenguaje natural (PLN) en español.

El modelo fue elegido debido a su robustez, disponibilidad pública en Hugging Face, y por estar preentrenado específicamente para tareas de clasificación semántica sin necesidad de ajuste adicional (fine-tuning). Su implementación permite procesar textos breves como los de redes sociales con una alta precisión, incluso ante lenguaje informal o gramaticalmente incorrecto.

El sistema alimenta el modelo con el campo texto_limpio generado previamente. Por cada texto, el modelo devuelve los siguientes valores:

- Etiqueta de sentimiento: clasifica el texto como positivo, negativo o neutral.
- Calificación numérica: un valor entero entre 1 y 5 que representa el nivel de intensidad o polaridad del sentimiento.
- Confianza: un valor decimal entre 0 y 1 que indica la certeza de la predicción, útil para filtrado o validaciones posteriores.

Estos resultados son añadidos como columnas nuevas al registro procesado. Esta etapa nos ayuda a visualizar los datos ya que los estructura, permitiendo que se puedan analizar y exportarlos.

4.4.4 Clasificación temática por tipo de transporte

Una vez realizado el análisis semántico de las publicaciones, el sistema procede a identificar el modo de transporte público al que se refiere cada opinión ciudadana. Esta etapa permite contextualizar los sentimientos expresados, asociándolos con categorías como metro, bus, ecovía u otro, lo cual resulta fundamental para generar visualizaciones segmentadas y reportes temáticos más relevantes.

Para lograr esta clasificación temática, se implementó una función basada en reglas heurísticas que opera sobre el texto limpio previamente procesado. Esta función analiza la presencia de términos específicos previamente definidos y los asocia con una categoría temática. En el caso

del metro, se consideraron expresiones como “metro”, “Metro de Quito” o “subterráneo”; para la ecovía, palabras como “ecovía”, “ecovia” o “corredor”; mientras que los textos que incluyen términos como “bus”, “buses”, “alimentador” o “micros” se clasifican como correspondientes al sistema de buses. En situaciones donde no se detecta ninguna de estas palabras clave, el texto se clasifica como "otro", permitiendo capturar publicaciones más generales o ambiguas.

Este enfoque resultó adecuado para los objetivos del presente prototipo. Al no requerir entrenamiento de un modelo adicional, permite una implementación ágil, replicable y modificable si en el futuro se desea refinar la lista de términos. La categoría temática detectada se asigna como una nueva variable denominada tipo_transporte, que se incluye en cada registro almacenado, permitiendo su uso posterior como criterio de segmentación en los dashboards.

4.4.5 Almacenamiento en base de datos

Con los datos completamente procesados y enriquecidos con las variables de análisis correspondientes, el siguiente paso en el flujo consistió en almacenar dicha información de forma estructurada en una base de datos relacional. Para ello, se utilizó Supabase, un sistema backend como servicio que ofrece compatibilidad con PostgreSQL y una API RESTful que facilita su integración con entornos Python.

La plataforma fue configurada para establecer conexión con Supabase utilizando variables de entorno definidas en un archivo .env, garantizando así la seguridad y la portabilidad del sistema. Una vez que cada publicación había sido limpiada, clasificada y categorizada temáticamente, el sistema ejecutaba la inserción del registro en la tabla analisis_sentimientos, que actúa como contenedor central de todos los datos procesados.

Cada fila de la tabla almacenó información relevante que incluyó: el texto original, el texto limpio, la etiqueta de sentimiento, la calificación del modelo, el nivel de confianza de la predicción, el tipo de transporte identificado y la fecha de creación del contenido original. Este diseño permitió consolidar en una sola estructura toda la información necesaria para el análisis posterior, eliminando la necesidad de múltiples tablas y relaciones complejas, lo cual resultó ideal para las condiciones de prototipado académico.

La inserción de datos se llevó a cabo mediante operaciones insert proporcionadas por la biblioteca supabase-py, lo que permitió una interacción fluida entre la aplicación desarrollada

en Streamlit y la base de datos en la nube. Este mecanismo no solo facilitó el almacenamiento seguro y remoto, sino que además garantizó que los datos estuvieran disponibles para su visualización inmediata dentro de los dashboards del sistema.

4.4.6 Visualización y exploración de resultados

Una vez que los registros fueron almacenados en la base de datos Supabase, el sistema permitió su consulta y exploración mediante una interfaz gráfica desarrollada con Streamlit. Este componente cumplió un rol clave en la interpretación y análisis de los datos procesados, facilitando el acceso a los resultados para usuarios sin conocimientos técnicos avanzados.

La visualización se implementó a través de dashboards interactivos que integraron múltiples filtros, permitiendo al usuario explorar los resultados según diferentes criterios como el tipo de transporte mencionado, la polaridad del sentimiento o el rango de fechas de publicación. Esta exploración dinámica fue posible gracias a la recuperación automática de los datos desde la base de datos, que se cargaban en tiempo real en la sesión activa de la aplicación.

Los gráficos generados fueron construidos utilizando la librería Plotly, que ofrece capacidades avanzadas de visualización interactiva. Entre los tipos de representación utilizados se incluyeron: gráficos de barras para observar la distribución de sentimientos por mes; gráficos circulares para mostrar proporciones por categoría temática; gráficos de líneas para identificar tendencias temporales; y nubes de palabras para resaltar los términos más frecuentes según la polaridad del texto.

Figura 16

Dashboards.

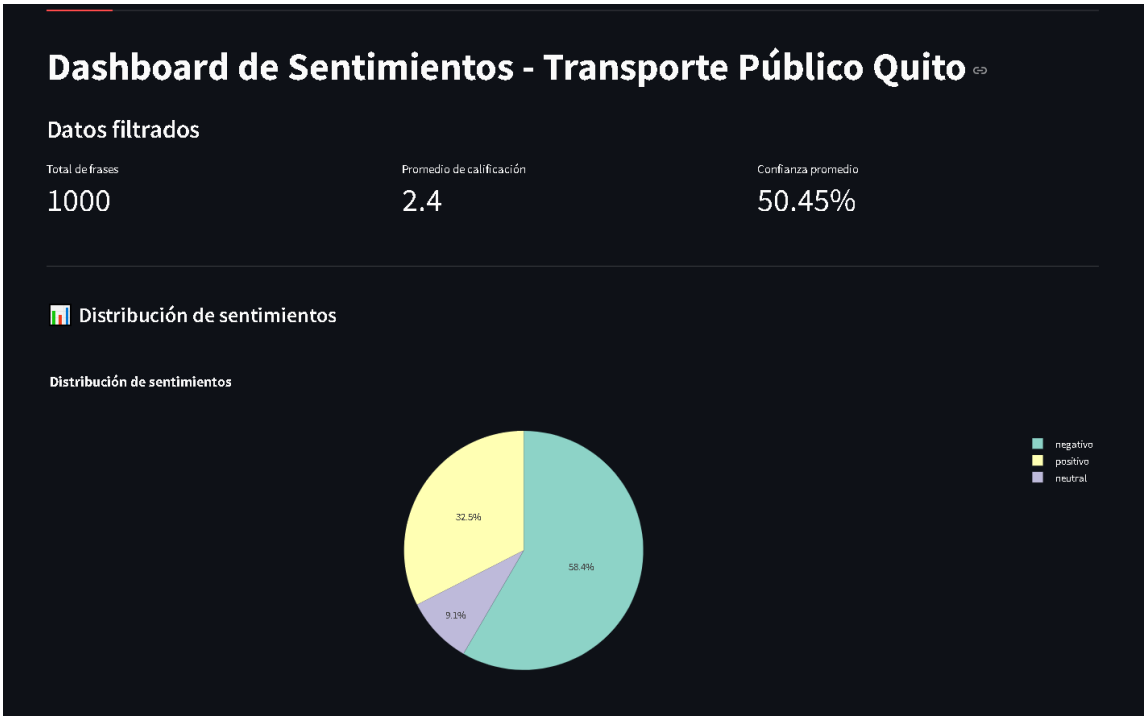


Figura 17

Dashboards.

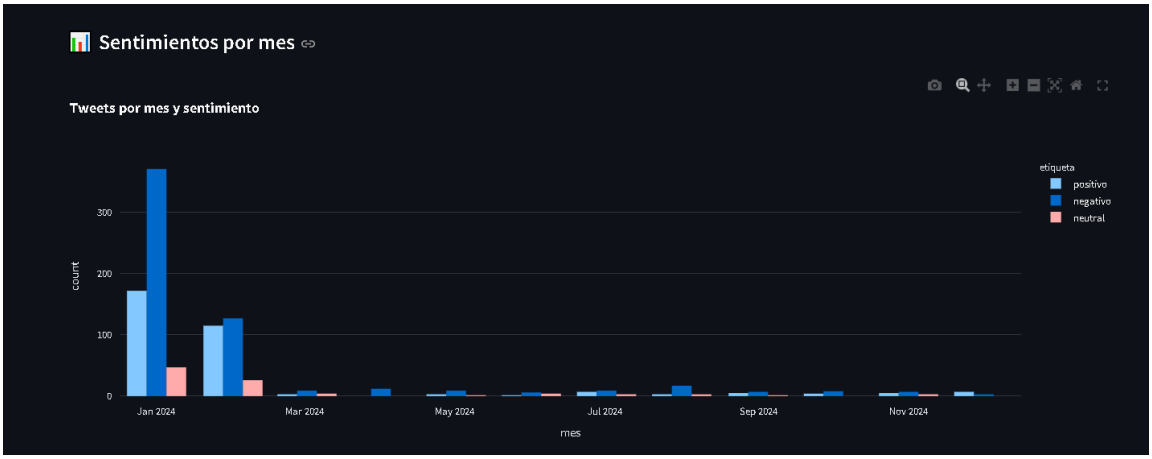
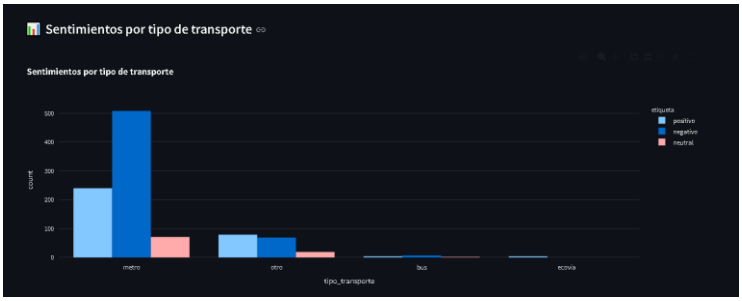


Figura 18

Dashboards.



Además, la interfaz incluyó tablas dinámicas que permitieron examinar publicaciones específicas, así como botones para aplicar o eliminar filtros, actualizando automáticamente los gráficos visibles. Esta interacción intuitiva convirtió al sistema en una herramienta de análisis exploratorio ágil, accesible y fácilmente comprensible.

4.4.7 Generación de reportes automáticos

Como parte final del flujo de análisis, se desarrolló una funcionalidad destinada a la exportación de los resultados procesados en formatos que faciliten su uso externo, almacenamiento o integración con otros sistemas. Esta etapa permitió al usuario generar reportes estructurados de manera automática, sin requerir conocimientos técnicos ni procesamiento adicional.

El sistema fue programado para generar reportes al finalizar cada análisis masivo, o a demanda del usuario mediante un botón dedicado en la interfaz. Estos reportes fueron exportados en dos formatos principales: archivos de Excel (.xlsx), organizados por hojas mensuales; y archivos de texto plano (.txt), que contenían resúmenes narrativos con los principales hallazgos del período evaluado.

El archivo Excel incluía columnas detalladas con todos los campos procesados para cada publicación: texto original, texto limpio, etiqueta de sentimiento, puntuación de intensidad, nivel de confianza, tipo de transporte detectado y fecha de publicación. Gracias al uso de la biblioteca pandas, fue posible estructurar esta información de forma tabular y exportarla con un formato amigable para ser reutilizada en otras plataformas o presentaciones.

Por otro lado, el archivo de texto plano se generaba automáticamente a partir de estadísticas agrupadas, tales como la cantidad total de publicaciones analizadas, distribución porcentual de los sentimientos, promedio de puntuaciones y términos más frecuentes para cada polaridad. Este documento era especialmente útil como insumo preliminar para informes institucionales o análisis de medios.

Figura 19

Generador de reportes por mes.

	Mes	Total	Positivos	Negativos	Neutrales	Calificación Promedio	Confianza Promedio (%)
0	2024-01	590	172	371	47	2.24	53.19
1	2024-02	268	115	127	26	2.82	46.95
2	2024-03	16	3	9	4	2.25	40.74
3	2024-04	12	0	12	0	1	48.95
4	2024-05	13	3	9	1	1.92	56.21
5	2024-06	11	1	6	4	2.09	43.03
6	2024-07	19	7	9	3	2.79	43.48
7	2024-08	22	3	17	2	1.77	51.39
8	2024-09	13	5	7	1	2.69	35.7
9	2024-10	12	4	8	0	2.25	46.21

4.5 Estructura de la plataforma y organización del código fuente

4.5.1 Estructura de la plataforma y organización del código fuente

La organización del código fuente del sistema se diseñó siguiendo un enfoque modular y funcional, el cual es ampliamente recomendado en proyectos de análisis de datos y desarrollo de aplicaciones web con Python. Este tipo de estructuración permite dividir la lógica del sistema en bloques independientes según su propósito principal, promoviendo la escalabilidad, la legibilidad del código y la facilidad de mantenimiento.

En este sentido, se adoptó una estructura de proyecto basada en la separación por responsabilidad funcional, también conocida como modular functional structure. Esta

disposición permite que cada carpeta represente una categoría lógica del sistema (por ejemplo, análisis, visualización, almacenamiento), agrupando los archivos que ejecutan tareas afines.

En el directorio raíz del proyecto se encuentran archivos de configuración y entorno como .env, donde se almacenan variables sensibles como las claves de conexión a la base de datos, y streamlit_app.py, el archivo que inicializa la aplicación principal de Streamlit. Además, se incluyen carpetas como tweets_2024/, que almacena el conjunto de datos base, y capturas/, utilizada para evidencias gráficas.

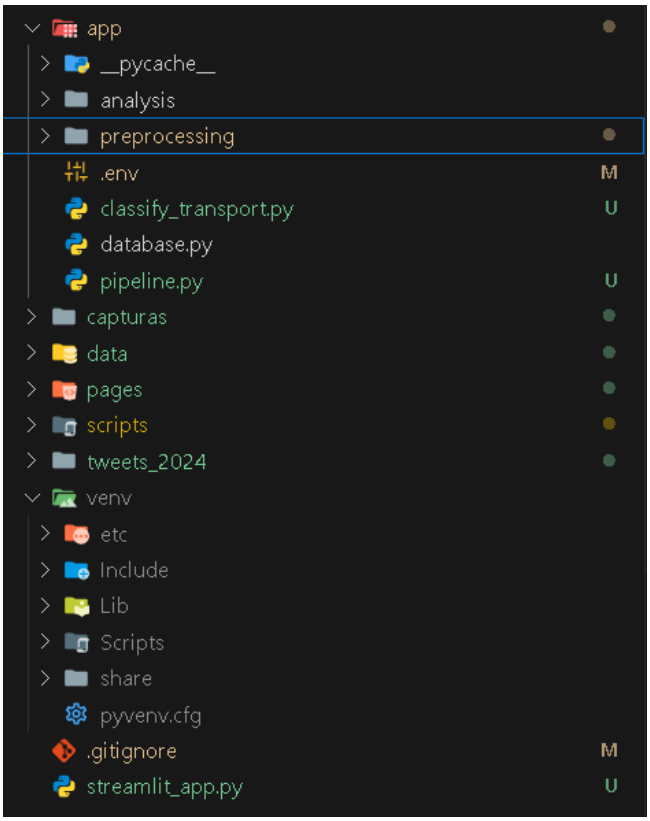
La carpeta principal app/ contiene la lógica funcional del sistema, dividida en submódulos como analysis/ y preprocessing/, así como archivos que implementan el flujo de análisis (pipeline.py), conexión con la base de datos (database.py) y detección semántica (classify_transport.py). Esta segmentación permite trabajar de forma aislada en cada componente y facilita la integración continua.

Complementariamente, se utilizó la carpeta pages/ para ubicar los scripts asociados a las distintas secciones de la interfaz gráfica, siguiendo la convención de multipáginas de Streamlit. Cada archivo representa una vista o sección del sistema: análisis manual, carga CSV, dashboards, validación del modelo y generación de reportes.

Por último, se creó la carpeta scripts/ para alojar utilidades, validaciones y procesos experimentales. Estos archivos fueron utilizados principalmente durante el desarrollo y exploración inicial de los datos, así como para pruebas del rendimiento del modelo.

Figura 20

Estructura de Carpetas



4.5.2 Módulos funcionales principales

Los módulos funcionales centrales del sistema se encuentran agrupados dentro de la carpeta app/, la cual representa el núcleo lógico del proyecto. Esta carpeta incluye los scripts encargados del preprocesamiento de texto, la clasificación del sentimiento, la categorización temática del transporte, la conexión con la base de datos y la ejecución del flujo general de procesamiento.

Uno de los archivos clave en esta sección es text_cleaner.py, que contiene funciones específicas para normalizar, limpiar y transformar el texto antes de ser analizado. Este script aplica las operaciones descritas en la etapa de preprocesamiento, tales como la eliminación de signos innecesarios, enlaces, menciones y la lematización de términos relevantes.

El archivo sentiment_classifier.py, ubicado dentro del subdirectorio analysis/, se encarga de implementar la interfaz con el modelo de análisis de sentimiento. Este módulo toma como

entrada el texto limpio y devuelve las etiquetas de polaridad, la calificación numérica y el valor de confianza que serán almacenados y visualizados.

Figura 21

Código de sentiment_classifier.

```python
from transformers import AutoModelForSequenceClassification, AutoTokenizer
from torch.nn.functional import softmax
import torch

model_name = "nlptown/bert-base-multilingual-uncased-sentiment"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForSequenceClassification.from_pretrained(model_name)

def rating_to_label(rating):
    if rating <= 2:
        return "negativo"
    elif rating == 3:
        return "neutral"
    else:
        return "positivo"

def classify_sentiment(text):
    inputs = tokenizer(text, return_tensors="pt", truncation=True, padding=True)
    outputs = model(**inputs)
    scores = softmax(outputs.logits, dim=1)
    scores = scores.detach().numpy()[0]
    rating = scores.argmax() + 1  # Escala de 1 a 5
    return {
        "texto": text,
        "calificacion": rating,
        "etiqueta": rating_to_label(rating),
        "confianza": round(float(scores[rating - 1]), 4)
    }
```

Por su parte, classify_transport.py implementa la lógica para la detección del tipo de transporte mencionado en cada publicación, usando reglas basadas en palabras clave. El script database.py contiene las funciones necesarias para realizar operaciones de inserción en Supabase, utilizando la biblioteca supabase-py y variables de entorno para garantizar la seguridad de las credenciales.

Figura 22

Script-Conexión a la BDD

```python
from supabase import create_client, Client
import os
from dotenv import load_dotenv

load_dotenv()  # Cargar variables desde .env

# Conexión
url: str = os.getenv("SUPABASE_URL")
key: str = os.getenv("SUPABASE_KEY")
supabase: Client = create_client(url, key)

def insertar_sentimiento(data: dict):
    """Inserta un análisis de sentimiento en la tabla de Supabase"""
    try:
        response = supabase.table("analisis_sentimientos").insert(data).execute()
        return response
    except Exception as e:
        print("Error al insertar en la base de datos:", e)
        return None
```

Finalmente, el archivo pipeline.py actúa como orquestador del flujo general. Este script centraliza la llamada a los distintos módulos de procesamiento, ejecutando el flujo completo desde la entrada del texto hasta su análisis, clasificación temática y posterior almacenamiento.

Esta distribución modular no solo mejora la mantenibilidad del sistema, sino que también permite desarrollar pruebas unitarias para cada componente, facilitando la detección de errores y asegurando la estabilidad de la plataforma en su conjunto.

4.5.3 Interfaz de usuario y navegación

El sistema de navegación de la plataforma se estructuró utilizando el esquema de multipáginas que ofrece Streamlit, con el propósito de distribuir las funcionalidades en vistas separadas y organizadas lógicamente. Esta estrategia permite mejorar la experiencia del usuario, al evitar la sobrecarga visual y funcional en una única interfaz, y facilita el mantenimiento del código al mantener cada sección en un archivo independiente.

Cada archivo ubicado en la carpeta pages/ corresponde a una sección específica del sistema y se integra automáticamente en el menú lateral de la interfaz web. De esta forma, se habilita una

navegación intuitiva por los distintos módulos: análisis de frases individuales, carga masiva de datos, visualización de resultados, validación del modelo y generación de reportes.

Por ejemplo, el archivo 1_Analisis_manual.py implementa la funcionalidad que permite analizar frases ingresadas por el usuario, mientras que 2_Carga_CSV.py gestiona la carga y el procesamiento de archivos .csv con publicaciones colectivas. La exploración de resultados se lleva a cabo mediante 3_Dashboards.py, que presenta visualizaciones dinámicas y filtros interactivos. La generación de reportes automáticos se implementa en 4_Reportes.py, y finalmente, 5_Validacion_modelo.py permite comparar predicciones del sistema con etiquetas manuales para evaluar la precisión del modelo.

Esta organización modular no solo permite mantener una estructura clara de navegación para el usuario final, sino que también favorece el desarrollo incremental del sistema, ya que nuevos módulos pueden ser integrados fácilmente sin alterar los existentes. Asimismo, facilita futuras adaptaciones a nuevas funcionalidades o cambios en los requerimientos, conservando la cohesión y escalabilidad del proyecto.

4.5.4 Scripts auxiliares y procesos adicionales

Durante el desarrollo del sistema, fue necesario complementar la arquitectura principal con scripts auxiliares que sirvieron de apoyo para tareas específicas, tanto en etapas iniciales como en procesos de validación y mantenimiento. Estos scripts fueron agrupados dentro de la carpeta scripts/, manteniéndose separados de los módulos de producción para evitar interferencias en la lógica del sistema principal.

Entre los scripts más relevantes se encuentran aquellos orientados al tratamiento inicial de los datos, como los utilizados para importar, limpiar y combinar archivos extraídos por scraping. También se desarrollaron scripts para evaluar el desempeño del modelo de análisis de sentimiento, aplicándolo sobre muestras etiquetadas manualmente y calculando métricas de precisión. Otros archivos ejecutaban exportaciones personalizadas, pruebas de carga, verificación de estructura de columnas y respaldos locales de los resultados.

Si bien estos scripts no forman parte directa del flujo interactivo expuesto al usuario, fueron fundamentales durante el desarrollo para garantizar la calidad, consistencia y rendimiento del sistema. Su existencia responde a una lógica de ingeniería de software donde las tareas

recurrentes o automatizables son encapsuladas como herramientas reutilizables y controladas desde una ubicación común.

4.5.5 Variables de entorno y configuración

Para garantizar la seguridad, portabilidad y flexibilidad del sistema, se implementó el uso de variables de entorno como mecanismo de gestión de información sensible y parámetros de configuración. Esta técnica es ampliamente utilizada en entornos de desarrollo profesional, ya que permite separar la lógica del código fuente de las credenciales, rutas y claves necesarias para la operación del sistema.

En este proyecto, las variables de entorno fueron definidas en un archivo oculto denominado .env, ubicado en la raíz del proyecto. Este archivo contiene valores como la URL del proyecto en Supabase, la clave pública de acceso (API key), y otras variables auxiliares que determinan aspectos de conectividad, entorno de ejecución o nombres de tablas.

Figura 23

Archivo .env

Para acceder a estas variables desde los distintos módulos del sistema, se utilizó la biblioteca python-dotenv, que permite cargar automáticamente las claves definidas en el entorno y acceder a ellas mediante el módulo estándar os.environ. Esta práctica asegura que los datos confidenciales no se expongan en el código fuente ni se incluyan en versiones compartidas del

proyecto (por ejemplo, en repositorios GitHub), reduciendo riesgos de seguridad y facilitando el cumplimiento de buenas prácticas de desarrollo.

Además, el uso de este sistema de configuración posibilita la migración fluida del proyecto entre distintos entornos (desarrollo, pruebas, producción), sin necesidad de modificar el código interno. Basta con cambiar el contenido del archivo .env para que el sistema se conecte a una nueva base de datos, utilice una API diferente o ejecute funciones bajo otro contexto. Este principio de desacoplamiento entre lógica y entorno mejora significativamente la mantenibilidad del proyecto.

Desde el punto de vista académico, la incorporación de variables de entorno también favorece la replicabilidad del sistema. Otros investigadores o desarrolladores pueden ejecutar el proyecto en sus propios equipos simplemente configurando su archivo .env sin requerir acceso a claves internas, asegurando así un uso controlado y adaptable.

4.6 Diagramas UML y arquitectónicas

Con el fin de complementar la documentación técnica del desarrollo de la plataforma, se incluyen a continuación diversos diagramas UML y de arquitectura. Estos elementos visuales permiten representar de forma estructurada las funcionalidades, componentes, clases y procesos involucrados en el sistema. Los diagramas seleccionados facilitan la comprensión del funcionamiento interno del prototipo, evidencian las decisiones de diseño tomadas y respaldan la organización del código y la interacción con los usuarios. La incorporación de estos esquemas responde a las buenas prácticas de ingeniería de software, y es particularmente relevante en contextos académicos donde se requiere claridad, trazabilidad y replicabilidad de los desarrollos realizados.

4.6.1 Diagrama de Caso de Uso

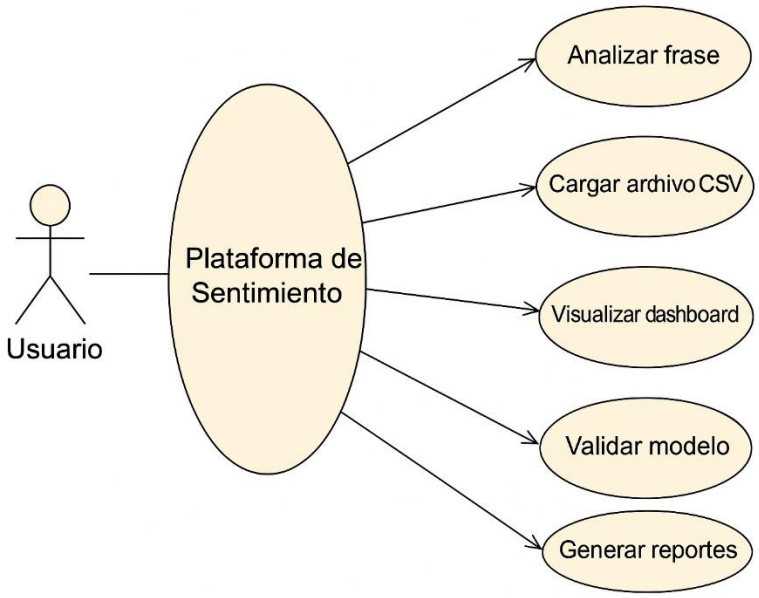
El diagrama de caso de usos presentado en la figura 24, resume las interacciones principales que un usuario puede realizar dentro de la aplicación; en este caso el usuario puede desempeñar el rol de analista, evaluador, investigador o un usuario convencional y mediante el acceso al sistema se despliega un conjunto de funcionalidades organizadas en módulos y páginas diferentes e interactivas.

Los casos de uso identificados son los siguientes:

- **CU1: Analizar frase**: permite al usuario ingresar manualmente una oración o mensaje y obtener de forma inmediata su clasificación de sentimiento (positivo, negativo o neutro). Esta opción es útil para realizar pruebas rápidas o explorar casos específicos.
- **CU2: Cargar archivo CSV**: habilita la carga de archivos que contienen múltiples publicaciones o frases. El sistema procesa automáticamente los registros mediante un pipeline que incluye limpieza de texto, clasificación de sentimiento y almacenamiento de resultados.
- **CU3: Visualizar dashboard**: da acceso a paneles interactivos donde se presentan gráficas dinámicas sobre la distribución del sentimiento, los tipos de transporte mencionados, la evolución temporal y otros indicadores relevantes derivados del análisis.
- **CU4: Validar modelo**: permite ejecutar pruebas de validación sobre el clasificador de sentimiento, comparando sus resultados con una muestra etiquetada manualmente. Esta funcionalidad sirve para evaluar el desempeño técnico del modelo.
- **CU5: Generar reportes**: ofrece la posibilidad de exportar informes mensuales con estadísticas y visualizaciones, facilitando su revisión o presentación ante terceros.

Figura 24

Diagrama de Casos de Uso



Cada uno de estos casos de uso responde a un objetivo específico dentro del sistema y fue diseñado en correspondencia con los requerimientos funcionales definidos previamente. El

enfoque modular y progresivo de las funciones busca garantizar una experiencia de usuario clara, flexible y adecuada tanto para uso académico como institucional.

4.6.1.1 Tabla de relación entre historias de usuario y casos de uso

A continuación se presenta una tabla que representa la relación entre las historias de usuario y los casos de uso modelados anteriormente.

Tabla 13

Tabla de relación entre historias de usuario y casos de uso

Historia de Usuario	Caso de Uso Relacionado
HU1	CU1 - Analizar frase individual
HU2	CU2 - Cargar archivo CSV
HU3	CU3 - Visualizar dashboard
HU4	CU4 - Validar modelo
HU5	CU5 - Generar reportes

4.6.2 Diagrama de Componentes

El diagrama de componentes representa la estructura lógica del sistema propuesto, evidenciando la organización modular de la aplicación, así como las relaciones entre sus distintas partes. Este diagrama permite visualizar cómo se distribuyen las responsabilidades entre los diferentes archivos y cómo interactúan tanto entre sí como con elementos externos.

En la Figura X se puede observar que el núcleo de la aplicación está contenido dentro de un componente principal denominado Plataforma de Sentimiento, el cual agrupa los scripts internos encargados del procesamiento, análisis y almacenamiento de la información.

Dentro de este contenedor se identifican los siguientes módulos:

- **pipeline.py**: es el componente que se encarga de coordinar el flujo general de la aplicación. Recibe las solicitudes desde la interfaz de usuario, y a partir de ello gestiona el envío de los datos hacia los módulos de limpieza y clasificación, además de registrar los resultados en la base de datos.

- **text_cleaner.py:** se ocupa del preprocesamiento del texto. Su función es limpiar y normalizar los mensajes recolectados para que puedan ser analizados correctamente por el modelo de sentimiento.
- **sentiment_classifier.py**: contiene el modelo de análisis de sentimiento, previamente entrenado. Este componente se encarga de recibir el texto limpio y determinar su polaridad.
- **database.py**: maneja la conexión con la base de datos externa, permitiendo guardar y recuperar información relacionada con los análisis realizados.

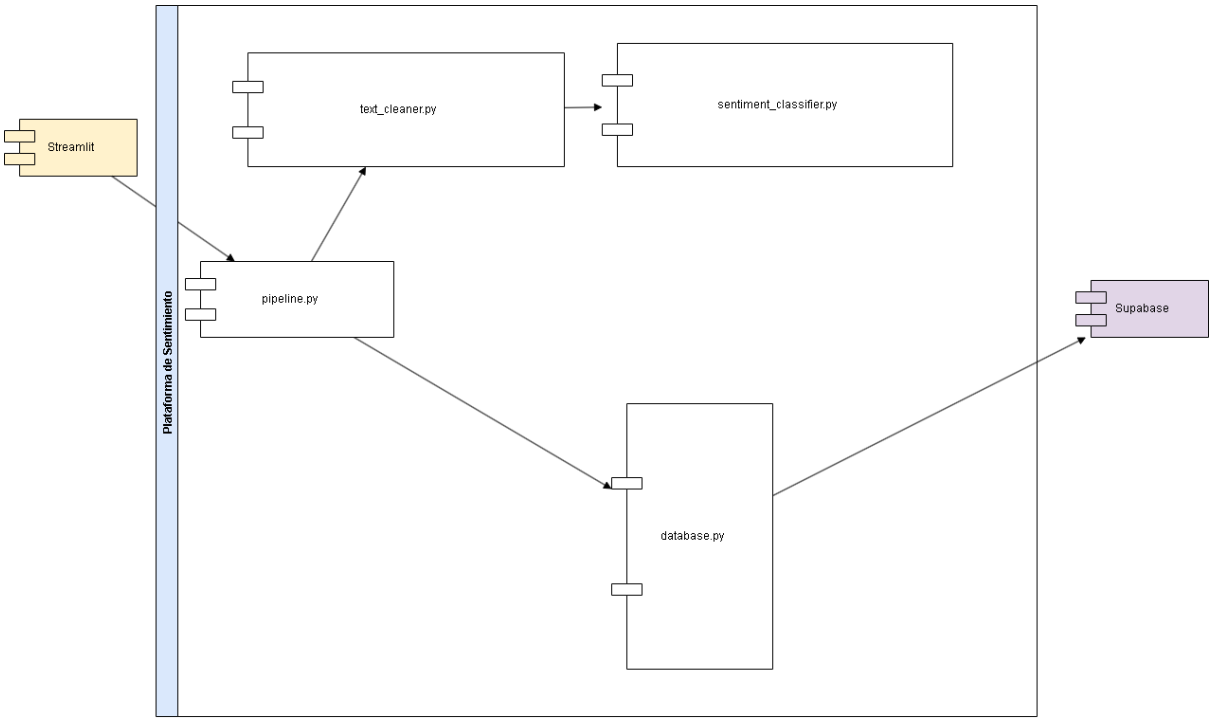
Fuera del componente principal se identifican dos módulos externos:

- **Streamlit:** es la herramienta utilizada para construir la interfaz de usuario. Desde esta capa, el usuario puede interactuar con la plataforma, cargar datos, realizar análisis y consultar los resultados.
- **Supabase:** servicio externo utilizado como base de datos. La comunicación con esta plataforma se realiza mediante el componente database.py, que contiene las funciones necesarias para la escritura y lectura de datos.

La interacción entre estos componentes se da de la siguiente manera: el usuario inicia una acción desde la interfaz de Streamlit, lo cual es recibido por pipeline.py. Este último llama a text_cleaner.py para limpiar el texto, luego a sentiment_classifier.py para analizar el sentimiento, y finalmente a database.py para almacenar los resultados. A su vez, database.py establece la conexión con Supabase para persistir la información.

Figura 25

Diagrama de Componentes.



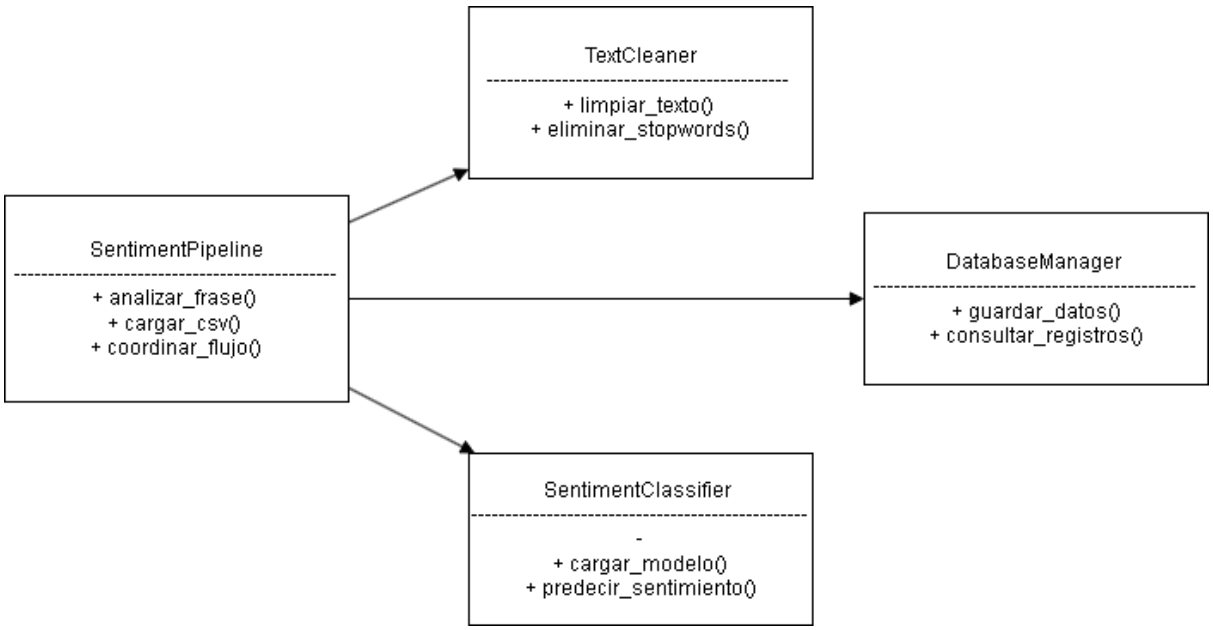
4.6.3 Diagrama de Clases

El diagrama de clases tiene como finalidad representar de forma estructurada los principales elementos funcionales que componen el sistema, organizándolos en bloques que agrupan responsabilidades similares. En este caso, debido a que la implementación no sigue un enfoque orientado a objetos, se ha optado por elaborar un diagrama de clases lógico, tomando como base los módulos desarrollados y las funciones que estos contienen.

En al Figura 26, se puede observar el diagrama como tal, el cual presenta la abstracción de clases que representa cada uno de los componentes centrales de la plataforma, cada bloque contiene sus propios métodos correspondientes a las funciones definidas dentro del proyecto, permitiendo visualizar como se distribuye las tareas principales del sistema.

Figura 26

Diagrama de Calces



- **SentimentPipeline:** Representa el módulo principal desde el cual se orquesta el flujo de ejecución. Contiene métodos para el análisis de frases, la carga de archivos CSV y la coordinación general del procesamiento de los datos.
- **TextCleaner:** Agrupa las funciones responsables del preprocesamiento del texto. Incluye métodos para limpiar y normalizar el contenido textual, como la eliminación de símbolos, caracteres especiales y palabras irrelevantes.

- **SentimentClassifier:** En esta clase lógica se encuentran las funciones que permiten cargar el modelo de análisis de sentimiento y ejecutar las predicciones sobre el texto previamente procesado.
- **DatabaseManager:** Define los métodos relacionados con la persistencia de los datos. Contempla funciones para guardar los resultados del análisis y para consultar registros almacenados en la base de datos externa.

Cada una de estas clases mantiene una relación directa con SentimentPipeline, ya que este módulo actúa como intermediario entre el ingreso de datos por parte del usuario y los distintos pasos del procesamiento interno. Así, el diseño modular reflejado en este diagrama facilita el mantenimiento y la comprensión del sistema.

4.6.4 Diagrama de Actividades

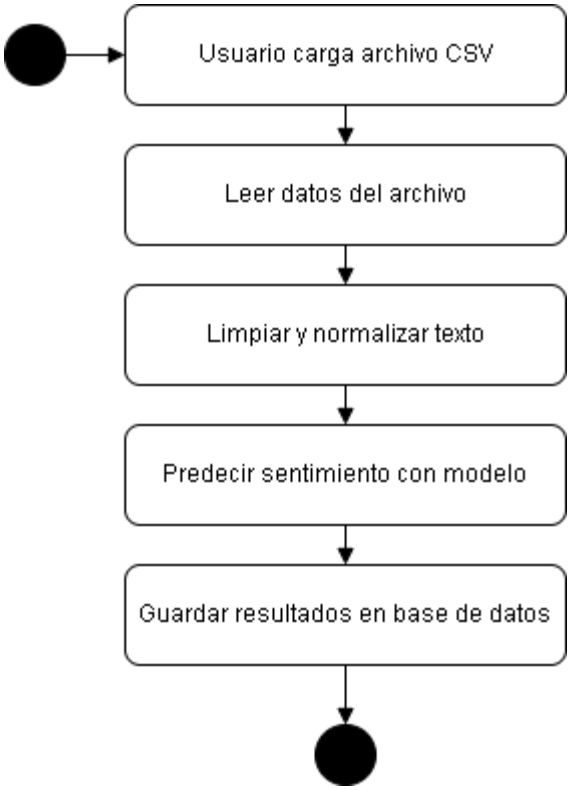
El diagrama de actividad es una herramienta que nos permite modelar la secuencia de acciones que se llevan a cabo dentro de un proceso del sistema, representando de manera visual el flujo de trabajo desde que inicia el proceso hasta que finaliza, en este caso se ilustró un diagrama acerca del procedimiento de carga y análisis de un archivo CSV, pasando con la Figura 27, esta muestra el flujo de trabajo de esta actividad la cual inicia con la acción del usuario de cargar el archivo CSV para posteriormente continuar con los siguientes pasos:

- **Lectura del archivo CSV:** el sistema extrae las frases o publicaciones contenidas en cada fila del documento.
- **Preprocesamiento del texto:** cada contenido es limpiado y normalizado, eliminando elementos no deseados como signos de puntuación, mayúsculas o palabras irrelevantes.
- **Clasificación de sentimiento:** una vez procesado el texto, se lo envía al modelo entrenado para predecir el sentimiento asociado (positivo, negativo o neutro).
- **Almacenamiento de resultados:** los datos enriquecidos con su respectiva clasificación se almacenan en la base de datos externa Supabase.

El flujo se considera terminado cuando el proceso de análisis del archivo ha sido completado, esta secuencia muestra el funcionamiento de la plataforma y evidencia el diseño modular de los componentes internos que permiten que el flujo se complete.

Figura 27

Diagrama de actividades.



CAPÍTULO 5: RESULTADOS Y VALIDACIÓN DEL SISTEMA

5.1 Validación funcional del sistema

Esta etapa representa una pieza fundamental que permitirá comprobar el cumplimiento de los objetivos establecidos en el proyecto, el enfoque se encuentra en verificar el funcionamiento operativo de la plataforma y sus módulos, desde la carga de datos hasta la generación de los reportes, incluyendo también el procesamiento, clasificación, almacenamiento y visualización de los resultados que son las principales y más importantes etapas. La validación se realizó en un entorno controlado, ejecutando flujos completos de uso que simulan el comportamiento de usuarios reales, con el fin de demostrar la estabilidad, integridad y coherencia del prototipo.

5.1.1 Flujo completo de uso del sistema

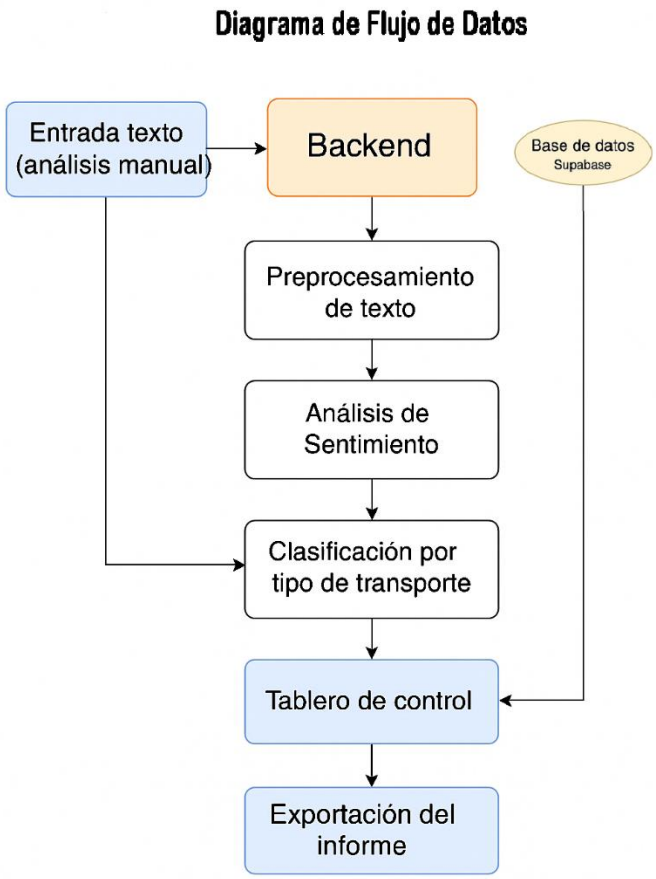
El flujo funcional de la plataforma se diseñó para ser ejecutado en su totalidad desde una interfaz web desarrollada con Streamlit, accesible mediante navegador, sin necesidad de conocimientos técnicos por parte del usuario. La navegación se organiza mediante una barra lateral que permite ingresar a los distintos módulos: análisis manual, carga masiva CSV, dashboards, validación del modelo y generación de reportes automáticos.

El proceso comienza con la entrada de datos, ya sea a través de una frase individual ingresada manualmente o mediante la carga de un archivo .csv. En ambos casos, el sistema activa automáticamente el pipeline completo que incluye la limpieza del texto, el análisis de sentimiento, la detección temática por tipo de transporte y la inserción del registro procesado en la base de datos remota en Supabase.

Una vez almacenados, los resultados pueden ser explorados a través del módulo de visualización, donde se presentan dashboards interactivos que permiten filtrar los registros por sentimiento, tipo de transporte, rango de fechas o palabras clave. Estos dashboards generan gráficos dinámicos que ofrecen al usuario una interpretación visual inmediata del estado de opinión ciudadana respecto al transporte público. Finalmente, desde el módulo de reportes, se puede exportar la información procesada en formatos estructurados como Excel o TXT, organizados por mes.

Figura 28

Diagrama de flujo de datos.



5.1.2 Comprobación de requerimientos funcionales

Durante la ejecución del sistema en su entorno de validación, se observó el comportamiento de los distintos módulos en relación con los requerimientos funcionales definidos previamente. Cada etapa fue verificada en condiciones controladas, replicando escenarios de uso orientados al flujo real del sistema por parte de un usuario final.

En primera instancia, el módulo de entrada manual de texto permite registrar opiniones ciudadanas mediante una interfaz simple desplegada en navegador. Esta vista fue desarrollada con Streamlit y presenta un campo de texto junto a un botón de análisis. Al ingresar una frase, se activa el proceso automatizado de preprocesamiento y clasificación, cuyos resultados se muestran inmediatamente en pantalla. Esta funcionalidad está orientada a usuarios que deseen explorar frases individuales sin necesidad de conocimientos técnicos. La interfaz responde

también ante entradas vacías, mostrando mensajes informativos que evitan errores en la ejecución.
Por otro lado, la sección de carga masiva admite archivos en formato .csv que contengan columnas de texto. Al cargar el archivo, se inicia automáticamente el pipeline de procesamiento, que analiza cada fila y envía los resultados a la base de datos remota. La validación se realizó utilizando archivos de prueba, verificando la interpretación de múltiples registros y su incorporación estructurada en el almacenamiento en Supabase. Esta funcionalidad resulta clave para el procesamiento por lotes y la escalabilidad del análisis.

En relación con el análisis semántico, el sistema muestra resultados de sentimiento categorizados en positivo, negativo o neutral, acompañados de una calificación numérica y un valor de confianza. Estos datos pueden visualizarse posteriormente en el módulo de dashboards. Allí, se presentan visualizaciones interactivas que permiten al usuario filtrar por sentimiento, tipo de transporte, fechas o palabras clave. Estas visualizaciones facilitan la exploración de los resultados y su análisis desde una perspectiva comparativa y temporal.

El módulo de reportes, también integrado en la plataforma, permite generar exportaciones en formatos estructurados. Se diseñó con la capacidad de segmentar los datos por mes, entregando archivos en Excel o texto plano, de modo que los resultados puedan ser utilizados para análisis posteriores o informes externos. Adicionalmente, se observaron opciones de descarga directa de los subconjuntos filtrados en la misma interfaz.

La interacción entre módulos y la transición fluida entre entrada, procesamiento y visualización fueron aspectos observados con detenimiento. El diseño modular adoptado permitió identificar claramente cada etapa del sistema, facilitando su análisis funcional. Asimismo, se contemplaron mecanismos de retroalimentación ante errores, como la validación del tipo de archivo cargado o el aviso ante registros sin contenido.

Todas estas funcionalidades fueron desarrolladas con el objetivo de responder a los lineamientos funcionales planteados en etapas anteriores del proyecto. La validación técnica de los flujos, respaldada por evidencias visuales de ejecución, constituye un insumo relevante para los procesos de revisión y evaluación establecidos por el tribunal o dirección académica. En las siguientes secciones se presentan capturas que ilustran el comportamiento de cada uno de los módulos descritos, así como ejemplos del resultado final generado por la plataforma.

5.1.3 Evidencias de funcionamiento por módulo

La validación del sistema se extendió a una revisión funcional detallada de cada módulo implementado en la plataforma, permitiendo observar cada interacción entre los componentes y su comportamiento en condiciones reales de uso, teniendo en cuenta la carga masiva de datos e individuales.

El módulo de análisis manual mostró un comportamiento estable al ingresar frases diversas, procesándolas de forma inmediata y devolviendo las etiquetas de sentimiento junto a valores numéricos de calificación y nivel de confianza. El sistema responde adecuadamente ante entradas incompletas o inválidas, mediante mensajes de advertencia que mejoran la experiencia de uso. Esta funcionalidad, orientada a la exploración rápida, evidencia la integración entre la interfaz de usuario y el backend de análisis semántico.

En el módulo de carga masiva se utilizaron archivos .csv que contenían publicaciones reales recolectadas anteriormente con una cantidad considerada adecuada para el análisis. Al ser cargados, el sistema activó automáticamente las funciones de preprocesamiento, clasificación de sentimiento y detección temática por palabras clave. Posteriormente, los registros procesados fueron verificados en la base de datos relacional en Supabase, mostrando que los campos generados (sentimiento, tipo de transporte, calificación, etc.) eran consistentes con el flujo de procesamiento establecido.

La interfaz de visualización presentó resultados organizados en gráficos dinámicos y filtros interactivos, permitiendo al usuario explorar los datos desde distintas perspectivas. Se observaron gráficas de distribución por sentimiento, líneas de tiempo por mes, y comparaciones por tipo de transporte. Esta interfaz está orientada a facilitar la interpretación de los patrones encontrados, y su comportamiento visual fue estable durante las sesiones de prueba.

El módulo de generación de reportes se activó desde el menú lateral de la aplicación. Al seleccionar el mes deseado, se obtuvo un archivo descargable en formato Excel con los registros procesados correspondientes a ese período. Este archivo incluyó tanto los textos originales como las etiquetas generadas por el modelo, permitiendo su uso en análisis externos. También se generaron versiones en texto plano para su revisión rápida.

A nivel de backend, se evidenció la conexión estable con Supabase mediante variables de entorno protegidas, así como la existencia de registros correctamente estructurados en la tabla principal analisis_sentimientos. Se accedió a estos datos desde el dashboard y desde la propia consola de administración de Supabase, observando que las fechas, etiquetas y valores numéricos coinciden con lo reportado en la interfaz de usuario.

5.2 Análisis de resultados obtenidos

Una vez desarrollados los módulos funcionales del sistema y procesadas las publicaciones ciudadanas recolectadas, fue posible explorar los resultados desde una perspectiva cuantitativa y visual. El análisis se enfocó en identificar patrones de comportamiento en las opiniones emitidas en torno al transporte público en Quito, categorizadas por polaridad del sentimiento, tipo de transporte mencionado y distribución temporal a lo largo del año 2024.

La plataforma permitió acceder a un conjunto de 10.996 registros procesados, los cuales fueron analizados mediante los dashboards interactivos implementados. Las visualizaciones generadas por la aplicación ofrecieron un panorama general de la percepción ciudadana, con métricas acumuladas y segmentadas que facilitaron una lectura integral del fenómeno.

5.2.1 Tendencias generales del sentimiento

La primera dimensión explorada en los resultados fue la polaridad general del sentimiento expresado por la ciudadanía en sus publicaciones, analizadas por el sistema a lo largo del año 2024. A partir de las etiquetas generadas por el modelo de análisis de sentimiento, fue posible identificar cómo se distribuyen las opiniones clasificadas como positivas, negativas y neutras, sobre un total de 10.996 registros válidos procesados por la plataforma.

Los gráficos generados a partir de este conjunto de datos evidenciaron una predominancia de publicaciones negativas, que superaron ampliamente a las positivas en la mayoría de los meses. Este comportamiento puede estar relacionado con una tendencia frecuente en redes sociales a expresar disconformidad, quejas o frustraciones, especialmente frente a servicios públicos, o a la gestión de las autoridades de turno. La polaridad negativa aparece con mayor claridad y carga emocional, lo que puede facilitar su clasificación por parte del modelo, en contraste con las publicaciones de tono positivo, que tienden a ser más sutiles o ambiguas.

Las opiniones clasificadas como neutras ocuparon un espacio intermedio en volumen, generalmente asociadas a publicaciones informativas, descripciones objetivas o expresiones sin carga emocional evidente. En algunos casos, estas opiniones podrían corresponder también a mensajes que, aunque emocionales, no contienen indicadores léxicos suficientes para que el modelo los clasifique con certeza como positivos o negativos.

Durante la visualización, se notaron variaciones en la proporción de cada polaridad entre diferentes semanas o meses, aunque la tendencia general de predominio negativo se mantuvo estable. Este comportamiento refuerza una hipótesis común en el análisis de redes: la opinión negativa se expresa con más frecuencia y con mayor intensidad emocional que la positiva, especialmente en contextos de movilidad urbana donde los usuarios enfrentan incomodidades cotidianas.

Cabe señalar que la interpretación de estos resultados debe realizarse con cautela, considerando que el modelo de análisis no ha sido entrenado específicamente con lenguaje ecuatoriano ni con datos sobre transporte. Por tanto, existe la posibilidad de que ciertas formas de ironía, localismos o expresiones ambiguas hayan sido clasificadas como neutras o en categorías opuestas a su intención original.

Sin embargo, la magnitud de datos procesados y la distribución polar consistente en distintas muestras temporales, ofrece un marco válido para considerar estas tendencias como representativas dentro de los límites de un análisis automatizado. El uso de herramientas visuales, como gráficas de barras y tortas, facilitó la comprensión de estas proporciones y ofreció al usuario final una interpretación directa de los sentimientos ciudadanos expresados en línea.

5.2.2 Segmentación por modo de transporte

Una de las funcionalidades integradas en el sistema fue la capacidad de identificar, dentro de cada publicación, referencias explícitas a distintos modos de transporte público en Quito. Esta clasificación temática se realizó mediante un sistema de detección por palabras clave, adaptado a los nombres y términos comúnmente utilizados por la ciudadanía en redes sociales. Gracias a este mecanismo, se logró segmentar los datos procesados en categorías como “Metro de Quito”, “Ecovía”, “Trole”, “Bus” y “Otro”.

El análisis de esta segmentación mostró que el sistema de buses convencionales es el más mencionado dentro del conjunto de datos recolectado, superando con amplitud y claridad al resto de los medios de transporte. Esta observación puede explicarse por el hecho de que se trata del sistema de uso más extendido y, al mismo tiempo, uno de los más problemáticos en cuanto a percepción ciudadana. En muchos casos, las publicaciones que mencionan a los buses están asociadas con sentimientos negativos vinculados a demoras, condiciones del servicio o incomodidades experimentadas por los usuarios.

La categoría Metro de Quito, por su parte, obtuvo una proporción importante de menciones, tanto en tono positivo como negativo. Si bien se identificaron publicaciones con críticas, también se registraron expresiones de expectativa, aprobación o valoración favorable respecto a su infraestructura o funcionamiento reciente, lo que sugiere una percepción pública más matizada. La Ecovía y el Trolebús, en cambio, fueron mencionados con menor frecuencia relativa, pero con un porcentaje significativo de opiniones negativas, especialmente relacionadas con la experiencia de viaje, capacidad y tiempos de espera.

La categoría “Otro” agrupa aquellas publicaciones donde no se logró identificar una referencia específica a un medio de transporte definido por el sistema. En algunos casos, estas publicaciones hacían mención a “movilidad” en general, al tráfico urbano, o al uso de taxis y plataformas de transporte privado, que no estaban contemplados en el listado principal de palabras clave. Esta categoría también fue útil para captar publicaciones que expresaban opiniones globales sobre el estado del sistema de transporte sin asociarlas a una entidad puntual.

Las visualizaciones generadas permitieron observar cómo se distribuyen las menciones por tipo de transporte y cómo varía la polaridad del sentimiento dentro de cada categoría. Por ejemplo, en el caso de los buses, predominan ampliamente las opiniones negativas, mientras que en el Metro se aprecia una distribución más equilibrada. Esta capacidad de segmentación permite un análisis más específico y dirigido, facilitando la detección de focos de malestar o de aprobación dentro del sistema urbano.

Cabe destacar que el enfoque utilizado para esta clasificación se basa en coincidencias directas con palabras clave predefinidas, por lo que no considera sinónimos, errores ortográficos o expresiones indirectas. A pesar de ello, el volumen y variedad de datos analizados permite

identificar patrones válidos para el análisis exploratorio, cumpliendo con el propósito de ofrecer una primera aproximación temática automatizada desde el procesamiento textual.

5.2.3 Variación temporal del sentimiento

Una de las funcionalidades integradas en los dashboards de la plataforma fue la capacidad de segmentar los resultados por meses, permitiendo observar la evolución del sentimiento ciudadano en torno al transporte público a lo largo del año 2024. Esta segmentación temporal permitió identificar fluctuaciones, patrones estacionales y posibles correlaciones con hechos o eventos coyunturales ocurridos en determinados períodos.

La exploración visual de los datos, mediante gráficos de líneas y barras por mes y polaridad, evidenció que las publicaciones negativas mantuvieron una presencia constante durante todo el año, con picos significativos en ciertos meses, como marzo, junio y septiembre. Estos aumentos podrían corresponder a momentos específicos de crisis o eventos relacionados con fallos en el servicio, cambios tarifarios, protestas, implementación de nuevas rutas o percepciones negativas amplificadas en redes sociales.

En contraste, las publicaciones positivas se mantuvieron relativamente estables y en volúmenes inferiores, con ligeros aumentos en meses como enero y agosto. Estas variaciones podrían estar relacionadas con momentos de apertura de nuevos servicios, anuncios institucionales positivos o simplemente con menor exposición mediática de eventos negativos. No obstante, su volumen reducido en comparación con las publicaciones negativas refleja, en parte, la tendencia general observada en entornos digitales, donde las experiencias desfavorables tienden a expresarse con mayor intensidad y frecuencia.

Las publicaciones neutras, por otro lado, mostraron una distribución más homogénea a lo largo del año, sin picos marcados. Esto puede explicarse por su carácter informativo o por la ambigüedad en su carga emocional, lo que evita que sean clasificadas en los extremos de la polaridad.

La herramienta de filtros implementada en la plataforma permitió examinar estos datos por segmentos temporales específicos, permitiendo una exploración más granular al nivel de semanas o días si se requería. Esta característica ofrece al usuario una herramienta útil para

correlacionar cambios en la percepción ciudadana con la aparición de eventos urbanos, decisiones políticas o incidentes operativos en los sistemas de transporte.

Aunque este análisis no se complementa con datos externos sobre acontecimientos concretos, las fluctuaciones detectadas constituyen un punto de partida para estudios longitudinales más profundos, en los que se pueda relacionar la percepción digital con variables externas como clima, tráfico, decisiones municipales, accidentes u operativos de control.

En síntesis, la variación temporal del sentimiento refleja que el análisis de opinión pública en redes sociales no es estático, sino que responde a ciclos de atención y percepción social. El sistema desarrollado permite capturar esta dinámica de forma automatizada, brindando un recurso valioso para el seguimiento continuo de la calidad percibida del transporte público en Quito.

5.2.4 Interpretación de resultados

La interpretación de los resultados procesados por la plataforma requiere considerar no solo las cifras y visualizaciones generadas, sino también el contexto social, lingüístico y tecnológico en el que fueron recolectados los datos. Las publicaciones ciudadanas extraídas desde la red social X reflejan percepciones espontáneas, sin control sobre su redacción, estructura ni intencionalidad explícita. Por tanto, si bien el sistema permitió clasificar y organizar estas opiniones, es importante asumir que toda interpretación está sujeta a limitaciones semánticas y contextuales propias del análisis automatizado.

Uno de los principales hallazgos fue la alta proporción de publicaciones con sentimiento negativo en relación al transporte público. Esta tendencia se manifestó tanto en la distribución general como en la segmentación por tipo de transporte, y se mantuvo de forma consistente en la mayoría de los meses analizados. Este resultado puede interpretarse como una muestra del descontento ciudadano respecto a la calidad, puntualidad o cobertura del sistema, aunque también puede estar influenciado por la naturaleza crítica que caracteriza a los espacios digitales, donde es común que los usuarios expresen frustraciones con mayor énfasis que valoraciones positivas.

En contraposición, las publicaciones positivas fueron menos frecuentes y más dispersas. Esto puede estar relacionado con varios factores: la estructura informal y ambigua del lenguaje

positivo, la menor urgencia de comunicar experiencias satisfactorias en redes sociales, o incluso con limitaciones del modelo de clasificación, que puede enfrentar dificultades para detectar ironía, doble sentido o elogios indirectos. En este sentido, una parte del contenido etiquetado como neutro podría incluir opiniones implícitamente positivas que el sistema no logró identificar como tales.

Por otro lado, la clasificación temática por tipo de transporte ofreció un panorama más específico, revelando que el bus convencional es el sistema más mencionado, seguido por el Metro de Quito y la Ecovía. Las diferencias en la carga emocional de las opiniones por cada categoría invitan a reflexionar sobre cómo la experiencia ciudadana varía según el medio de transporte utilizado, y como estas diferencias pueden ser analizadas para orientar políticas de movilidad.

El análisis temporal aportó otra dimensión valiosa, mostrando fluctuaciones mensuales en el volumen y polaridad de las publicaciones. Aunque este trabajo no se centró en correlacionar eventos externos, los picos detectados permiten identificar momentos clave de la conversación pública, y podrían ser asociados a medidas, incidentes o anuncios institucionales si se integraran datos contextuales adicionales.

Finalmente, es importante destacar que los resultados obtenidos no representan la totalidad de la opinión pública, sino una muestra filtrada por palabras clave y recolectada en una red social con características propias. Sin embargo, la automatización del análisis y la visualización estructurada de estos datos constituyen un primer paso relevante para convertir la voz digital ciudadana en insumo para el diagnóstico urbano, especialmente en un entorno complejo como el del transporte público de Quito.

5.3 Validación del modelo de análisis de sentimiento

El sistema de análisis de sentimiento utilizado en esta plataforma se basa en un modelo preentrenado, específicamente nlptown/bert-base-multilingual-uncased-sentiment, el cual fue seleccionado por su capacidad de clasificar textos en diferentes idiomas —incluido el español— sin necesidad de entrenamiento adicional. No obstante, dada la naturaleza crítica de este trabajo de titulación, fue necesario validar de forma independiente el comportamiento del

modelo al aplicarse sobre un corpus específico, compuesto por opiniones ciudadanas sobre transporte público en Quito.

Con ese fin, se diseñó una prueba de validación basada en un conjunto de control compuesto por 100 publicaciones seleccionadas aleatoriamente del total de datos recolectados. Estas publicaciones fueron etiquetadas manualmente por un evaluador humano, aplicando criterios de interpretación subjetiva basados en la polaridad percibida: positivo, negativo o neutro. Posteriormente, se compararon las etiquetas generadas manualmente con las asignadas automáticamente por el modelo, lo que permitió construir una matriz de confusión y calcular métricas de evaluación comunes en tareas de clasificación.

5.3.1 Evaluación cuantitativa del modelo

La evaluación cuantitativa del modelo se llevó a cabo con el objetivo de estimar su desempeño sobre el corpus de publicaciones ciudadanas relacionadas con el transporte público en Quito. Para ello, se seleccionó aleatoriamente una muestra de 100 publicaciones procesadas por el sistema, las cuales fueron clasificadas manualmente en tres categorías: positivo, negativo y neutro. Esta clasificación humana fue luego comparada con las etiquetas generadas automáticamente por el modelo nlptown/bert-base-multilingual-uncased-sentiment.

El análisis de esta comparación permitió construir una matriz de confusión multiclase. Los datos revelaron que la clase “negativo” fue la que obtuvo la mayor tasa de aciertos, lo que podría atribuirse a la presencia de léxico emocional explícito en publicaciones de tono crítico. Por el contrario, las clases “positivo” y “neutro” presentaron mayor cantidad de errores relativos, con frecuentes confusiones entre sí.

A partir de la matriz, se calcularon métricas cuantitativas de desempeño. El modelo alcanzó una precisión global de 70%, con un F1-score superior al 0.80 en la categoría negativa, pero más bajo en las restantes. Estos resultados sugieren que, aunque el modelo ofrece un rendimiento aceptable en tareas de clasificación automática, su efectividad varía de manera importante dependiendo del tono y claridad del texto.

Además, se observó que textos breves o escritos con ambigüedad semántica tienden a ser mal clasificados, lo cual es consistente con la literatura sobre análisis de sentimiento en redes sociales. A pesar de estas limitaciones, la evaluación cuantitativa confirma que el modelo puede

ser utilizado como herramienta exploratoria para detectar tendencias agregadas, especialmente en análisis con gran volumen de datos.

5.3.2 Limitaciones observadas en el modelo

Durante el proceso de validación manual y análisis de las predicciones, se identificaron diversas limitaciones inherentes al uso de modelos preentrenados aplicados sobre lenguaje no controlado, como el de las redes sociales. Uno de los principales desafíos observados fue la dificultad del modelo para interpretar ironía, sarcasmo y lenguaje figurado, frecuentes en publicaciones ciudadanas. En estos casos, la polaridad atribuida automáticamente puede diferir sustancialmente de la intención real del autor.

Asimismo, el modelo no está adaptado específicamente al registro lingüístico ecuatoriano ni al dominio temático del transporte urbano. Esto limita su capacidad para reconocer expresiones locales, abreviaciones comunes o apelativos informales empleados para referirse a los medios de transporte. Términos como “trole”, “full”, “colapsado”, entre otros, podrían no estar correctamente ponderados dentro del contexto del modelo.

Otra limitación significativa radica en el enfoque basado exclusivamente en texto. El sistema no considera elementos multimedia ni señales paratextuales, como emojis o signos repetidos, que en muchos casos complementan o refuerzan el sentimiento de una publicación. De igual manera, no se incorporan variables como el autor, el contexto temporal o la interacción (likes, retweets), que podrían enriquecer el análisis semántico.

Por último, se identificó que el modelo tiende a sobrerrepresentar la categoría “neutro” en textos ambiguos o poco expresivos, lo cual reduce su sensibilidad a matices emocionales suaves. Este comportamiento podría afectar el balance final en análisis longitudinales si no se ajusta mediante estrategias de calibración o incorporación de modelos alternativos.

Estas limitaciones no invalidan el uso del modelo, pero sí deben ser consideradas al interpretar sus resultados. En escenarios aplicados, como la toma de decisiones públicas o el diseño de políticas de movilidad, se sugiere acompañar el análisis automático con mecanismos de validación humana o con el entrenamiento de modelos adaptados al contexto local.

5.4 Evaluación de usabilidad del sistema

Dado que la plataforma fue diseñada para ser utilizada por usuarios sin conocimientos técnicos, resulta relevante analizar su comportamiento desde la perspectiva de la usabilidad y la interacción. Aunque no se ejecutó una prueba formal con usuarios externos, se aplicaron principios de diseño centrado en el usuario durante el desarrollo de la interfaz, orientando decisiones visuales, de navegación y de retroalimentación.

La plataforma fue implementada utilizando Streamlit, lo que facilitó una navegación intuitiva mediante una barra lateral organizada por secciones. Cada módulo de análisis (entrada manual, carga CSV, dashboards, validación del modelo y reportes) está claramente delimitado y accesible desde el menú principal. Las instrucciones integradas en cada sección, el uso de iconos representativos y las alertas contextuales fueron incorporadas como estrategias para facilitar la comprensión de las acciones requeridas, especialmente en la carga de datos o la exportación de reportes.

Adicionalmente, se realizaron pruebas internas de usabilidad empleando escenarios de uso simulados, con el fin de identificar potenciales bloqueos o ambigüedades en la interacción. Estas pruebas revelaron que la experiencia de uso se mantiene estable en distintos navegadores modernos y sistemas operativos (Windows y Linux), sin requerir configuraciones adicionales. En cuanto al desempeño, la respuesta del sistema fue fluida incluso al procesar archivos CSV con más de 500 registros, lo que refuerza su aplicabilidad en contextos reales de análisis ciudadano o académico.

Si bien no se aplicaron cuestionarios estandarizados como SUS (System Usability Scale) ni se recogieron evaluaciones de usuarios finales, la retroalimentación obtenida durante el desarrollo sugiere que el sistema cumple con criterios básicos de accesibilidad, consistencia visual y eficiencia operativa. Se priorizó una curva de aprendizaje baja, permitiendo que cualquier usuario con conocimientos básicos en informática pueda utilizar la herramienta sin necesidad de formación adicional.

Como mejora futura, se recomienda la inclusión de pruebas de usabilidad participativas, donde usuarios externos interactúen con la plataforma y proporcionen valoraciones mediante métricas

objetivas y observación directa. Esto permitirá ajustar detalles del flujo de navegación, contenido informativo o posibles puntos de fricción no identificados durante el desarrollo.

CAPÍTULO 6: CONCLUSIONES Y RECOMENDACIONES

6.1 Conclusiones generales del trabajo

1. El desarrollo del prototipo evidenció que es técnicamente viable construir una plataforma web accesible que permita procesar y analizar publicaciones ciudadanas sobre el transporte público en Quito. Mediante el uso de herramientas de código abierto, como Streamlit para la interfaz y Supabase como sistema de almacenamiento, fue posible diseñar un flujo completo de análisis sin depender de soluciones propietarias ni entornos complejos de despliegue. Este enfoque demostró que incluso en contextos académicos, donde los recursos suelen ser limitados, es factible generar herramientas con aplicaciones prácticas reales, permitiendo a usuarios sin experiencia técnica interactuar con los datos, explorar visualizaciones y obtener reportes útiles sin complicaciones.

2. El diseño modular del sistema permitió integrar múltiples componentes de procesamiento de lenguaje natural en una misma aplicación, lo que facilitó el flujo entre limpieza de texto, análisis de sentimiento y clasificación temática sin fricciones técnicas. La existencia de una sola fuente de verdad —la tabla principal de la base de datos— contribuyó a la eficiencia en la consulta y visualización de resultados. Esta arquitectura monoestructural simplificó tanto la implementación como las pruebas funcionales, permitiendo lograr un funcionamiento estable durante sesiones prolongadas y con distintos volúmenes de datos.

3. A nivel de procesamiento semántico, el sistema fue capaz de analizar de forma automatizada un volumen representativo de publicaciones, identificando patrones en las opiniones de los ciudadanos sobre los distintos modos de transporte. Aunque no se trató de un modelo entrenado desde cero, su adaptación mediante técnicas de clasificación y limpieza específicas para el español permitió lograr resultados consistentes, útiles para la exploración de tendencias y para la construcción de reportes mensuales estructurados. Esta capacidad de síntesis fue esencial para transformar texto libre en información estructurada y visualizable.

4. Desde la perspectiva funcional, la plataforma respondió de forma adecuada a los distintos escenarios de entrada previstos: análisis individual y carga masiva. Ambos flujos activaron el procesamiento completo sin intervención técnica, lo que evidencia un grado aceptable de autonomía del sistema para su uso potencial en contextos ciudadanos o institucionales.

Además, las visualizaciones integradas permitieron representar con claridad los datos almacenados, segmentándolos por polaridad, tiempo y tipo de transporte, facilitando así la comprensión de los resultados incluso para usuarios no especializados.

5. En términos de validación, la implementación del módulo de evaluación comparativa permitió verificar el comportamiento del modelo mediante una muestra etiquetada manualmente. Esta estrategia reveló que, si bien el modelo no alcanza precisión perfecta, su rendimiento es suficiente para cumplir con el propósito del sistema: ofrecer una visión general sobre el sentimiento ciudadano, más que una clasificación exacta en todos los casos. El uso de métricas como precisión, recall y F1-score en la validación cuantitativa fue clave para reconocer sus limitaciones sin comprometer la funcionalidad global del prototipo.

6. Otro aspecto destacable fue la incorporación de mecanismos de exportación y generación automática de reportes, los cuales consolidan los datos recolectados mes a mes y permiten su análisis posterior fuera del entorno de la plataforma. Esta funcionalidad mejora la utilidad práctica del sistema, permitiendo compartir hallazgos en contextos académicos o institucionales sin depender de conexiones activas al sistema o conocimientos técnicos en visualización de datos. Su integración directa a la interfaz refuerza el carácter autoexplicativo y autónomo de la herramienta desarrollada.

7. Finalmente, el proyecto en su conjunto demuestra que la ciencia de datos puede aplicarse de forma efectiva a problemas locales mediante soluciones accesibles y reproducibles. El caso del transporte público en Quito sirvió como un ejemplo concreto para validar el potencial de este enfoque, permitiendo visibilizar opiniones ciudadanas que usualmente se diluyen entre grandes volúmenes de publicaciones. La posibilidad de adaptar y escalar la plataforma en el futuro abre el camino hacia soluciones más completas, que incluyan detección de emociones, análisis geográfico o integración con otras fuentes de información urbana.

6.2 Recomendaciones técnicas y metodológicas

1. Para futuros trabajos que busquen replicar o extender esta propuesta, se recomienda mantener un enfoque modular en la arquitectura del sistema. La separación clara entre preprocesamiento, análisis semántico, almacenamiento y visualización permitió desarrollar, validar y mejorar cada componente de forma aislada, reduciendo errores y facilitando las pruebas. Este diseño no solo mejora la mantenibilidad del sistema, sino que además favorece su escalabilidad, ya que nuevos módulos —como geolocalización, análisis de emociones o extracción de entidades— pueden integrarse sin afectar el núcleo funcional existente.

2. Desde el punto de vista técnico, es aconsejable considerar el entrenamiento de modelos de análisis de sentimiento adaptados específicamente al dominio del transporte público en el contexto ecuatoriano. Aunque los modelos preentrenados empleados ofrecieron resultados funcionales, su precisión puede mejorarse mediante el uso de datos limpios de publicaciones locales, lo que permitiría capturar con mayor fidelidad las expresiones lingüísticas, modismos y referencias culturales del entorno. Esto es especialmente relevante si se desea escalar la herramienta para uso institucional o tomar decisiones en base a los datos generados.

3. En cuanto al almacenamiento, si bien el uso de Supabase facilitó el manejo de datos y su consulta desde la interfaz, se recomienda evaluar alternativas que permitan mayor control de la infraestructura y mejor manejo de permisos si el sistema fuera a implementarse en un entorno de producción. Para proyectos con requisitos de seguridad elevados, convendría explorar opciones autogestionadas de bases de datos o servicios más robustos en términos de autenticación y auditoría de acceso.

4. Metodológicamente, se sugiere documentar con mayor detalle la limpieza y transformación de datos. Si bien este proyecto incluyó un módulo claro de preprocesamiento, el análisis reproducible y transparente se vuelve clave a medida que el sistema evoluciona. Incluir bitácoras automáticas o registros intermedios del procesamiento ayudaría a depurar posibles errores y a explicar mejor cómo se llega desde un texto original hasta su etiqueta final de sentimiento.

5. En cuanto a la interacción con el usuario, se recomienda incorporar validaciones más robustas y mensajes de retroalimentación más descriptivos. Esto no solo mejora la experiencia

de uso, sino que también ayuda a evitar entradas incorrectas, fallos silenciosos o resultados inconsistentes. También podría explorarse la posibilidad de integrar un sistema de ayuda o tutorial guiado que oriente al usuario en cada sección del sistema, especialmente si se espera que sea utilizado por personas sin experiencia previa en análisis de datos.

6. Finalmente, como recomendación general, se plantea la conveniencia de realizar estudios de usabilidad más formales con usuarios reales en fases tempranas de validación. Si bien el diseño actual fue pensado para ser intuitivo, contar con retroalimentación estructurada desde el inicio puede fortalecer el desarrollo iterativo y asegurar que el sistema no solo sea funcional, sino también útil, comprensible y apropiado para su público objetivo. Esta recomendación es clave si se proyecta migrar la solución hacia un entorno productivo o institucional, donde la aceptabilidad del usuario final es crítica para el éxito del sistema.

Bibliografía

Bird, S., Klein, E., & Loper, E. (2009). *Natural Language Processing with Python.* Sebastopol: O'Reilly.

Developers, N. (n.d.). *NumPy User Guide.* Retrieved from NumPy.org: https://numpy.org/doc/stable/user/

Explosion AI. (n.d.). *spaCy.io.* Retrieved from https://spacy.io/usage

Hall, J. &. (2022). Rapid dashboard development with Streamlit for COVID-19 visualization. *Data Science Journal*.

Han, J., Pei, J., & Kamber, M. (2011). *Data Mining: Concepts and Techniques.* Boston: Morgan Kaufmann.

Jurafsky, D., & Martin, J. H. (2021). *Speech and Language Processing.* Boston: Pearson.

Lewis Tunstall, L. v. (2021). *Transformers for Natural Language Processing.* Birmingham, UK: Packt Publishing.

Microsoft. (2016). *Visual Studio Code Documentation.* Retrieved from https://code.visualstudio.com/docs

Schröer, C. (2019). A Systematic Literature Review on Applying CRISP-DM Process Model. *Procedia Computer Science* (pp. 716–725). Elsevier.

Sun, Y. (2021). Topic modeling-based bibliometric analysis of transportation research. *Journal of Transport and Land Use*, 107–122.

Verma, V. (2023). Sentiment Analysis and Topic Modeling in Transportation. *Applied Sciences*, 6576.