CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERÍAS

DEPARTAMENTO DE CIENCIAS COMPUTACIONALES

Programación para internet

Sección: D03

Proyecto Final – Better Essay

Profesor: Michel Emanuel Lopez Franco

Alumna: Katia Marlene Salcedo Huerta

Contenido

Introducción

“Better Essay” consiste en una aplicación web orientada a la mejora de trabajos de escritura, la cual permite a los usuarios corregir ensayos y generar resúmenes automáticos de manera eficiente. Esta aplicación aprovecha el uso de modelos avanzados de Inteligencia Artificial para analizar el contenido textual, detectar errores, proponer mejoras y sintetizar información relevante a partir de textos extensos.

La aplicación fue desarrollada utilizando un conjunto de tecnologías modernas tanto en el frontend como en el backend. Para la interfaz de usuario se empleó React, permitiendo crear una experiencia dinámica e intuitiva. El servidor fue construido con Node.js y Express, encargándose de la lógica de negocio y la gestión de peticiones, mientras que la base de datos se implementó en MongoDB Atlas para el almacenamiento seguro de la información de los usuarios. La autenticación, recuperación de contraseña y manejo de sesiones se integraron mediante el uso de tokens y envío de correos electrónicos.

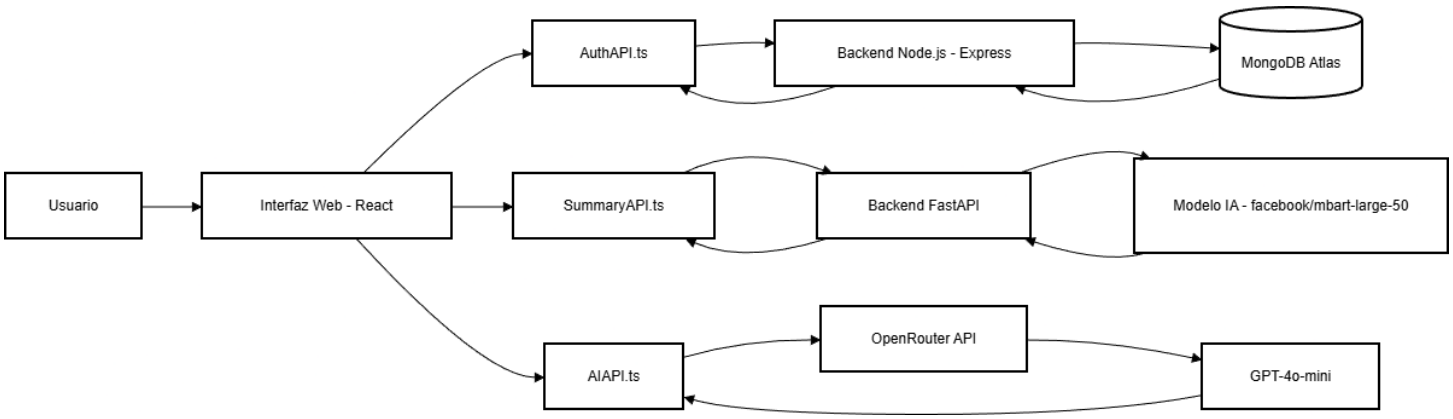
Adicionalmente, se incorporó un microservicio desarrollado en FastAPI (Python) para la generación de resúmenes automáticos, empleando el modelo facebook/mbart-large-50 de Hugging Face. Por otro lado, la corrección de ensayos se realizó mediante la integración de OpenRouter con el modelo GPT-4o Mini, el cual permitió analizar y mejorar ensayos de forma precisa, identificando errores gramaticales, problemas de coherencia y oportunidades de mejora en la redacción.

Este proyecto tiene como objetivo principal facilitar el proceso de escritura y revisión de ensayos, así como la generación de resúmenes automáticos, ofreciendo a los usuarios una herramienta accesible, rápida y eficaz que les permita optimizar la calidad de sus textos. De esta manera, se busca apoyar principalmente a estudiantes, docentes y cualquier persona interesada en perfeccionar sus habilidades de escritura mediante el uso de tecnologías basadas en Inteligencia Artificial.

Arquitectura y Programación de Sistemas

El proyecto “BetterEssay” es una aplicación web cuya finalidad es apoyar a los usuarios en la generación de resúmenes automáticos a partir de textos extensos, utilizando tecnologías de Inteligencia Artificial. Para lograr esto, el sistema está estructurado en diferentes módulos que trabajan de forma independiente pero coordinada.

La arquitectura implementada se basa en una separación de responsabilidades, utilizando dos backends principales: uno destinado al manejo de usuarios y autenticación, y otro especializado en el procesamiento de textos mediante Inteligencia Artificial. Adicionalmente, el frontend cuenta con una API que se comunica directamente con OpenRouter para la generación de respuestas basadas en modelos de lenguaje como GPT-4.



Modelado y estructuración del sistema

El proyecto se encuentra estructurado bajo una arquitectura distribuida de tres capas principales:

1. Capa de presentación: Interfaz web desarrollada en React, desde donde el usuario realiza todas las acciones: iniciar sesión, enviar ensayos y solicitar resúmenes.
2. Capa de procesamiento:
    - Backend 1: Node.js + Express (autenticación y control de usuarios).
    - Backend 2: FastAPI (procesamiento de resúmenes automáticos).

3. Capa de datos e inteligencia:
    - MongoDB Atlas (almacenamiento de datos).
    - OpenRouter + GPT-4o-mini (corrección de ensayos).
    - facebook/mbart-large-50 (generación de resúmenes).

La comunicación entre capas se realiza mediante protocolos HTTP/HTTPS usando APIs REST.

Esta organización permite un sistema robusto, escalable y listo para integrarse en entornos reales.

Componentes principales del sistema

El sistema está compuesto por los siguientes elementos:

Frontend (React)

Es la interfaz de usuario del sistema. Está desarrollada en React y permite al usuario:

- Registrarse e iniciar sesión.
- Ingresar textos a resumir y ensayos.
- Visualizar los resultados generados por la IA.
- Comunicarse tanto con el backend de autenticación como con el backend de IA.

Además, el frontend cuenta con una API interna encargada de comunicarse con OpenRouter, que es el servicio utilizado para acceder a modelos avanzados como GPT-4.

Backend de Autenticación (Node.js + Express)

Este backend es responsable de gestionar todo lo relacionado con:

- Registro de usuarios.
- Inicio de sesión.
- Encriptación y verificación de contraseñas.
- Generación de tokens JWT para sesiones seguras.
- Envío de correos de confirmación y recuperación de contraseña mediante Nodemailer.

Se conecta directamente con la base de datos para almacenar y validar la información de los usuarios.

Base de Datos (MongoDB)

La base de datos almacena:

- Información de los usuarios.
- Correos electrónicos.
- Contraseñas encriptadas.
- Tokens de recuperación y confirmación.
- Información necesaria para la gestión de sesiones.

MongoDB fue elegido por su flexibilidad y buen rendimiento en aplicaciones modernas basadas en JSON.

Backend de Inteligencia Artificial (FastAPI)

El sistema cuenta con un segundo backend independiente desarrollado en Python utilizando FastAPI, cuya función principal es la generación de resúmenes automáticos mediante Inteligencia Artificial.

Este backend se encarga de recibir textos enviados desde la interfaz web y procesarlos para producir un resumen coherente, reducido y estructurado. Para lograrlo, se apoya en un modelo de procesamiento de lenguaje natural especializado en tareas de resumen automático, el cual es ejecutado localmente a través de la librería Transformers.

De forma general, este servicio realiza las siguientes tareas:

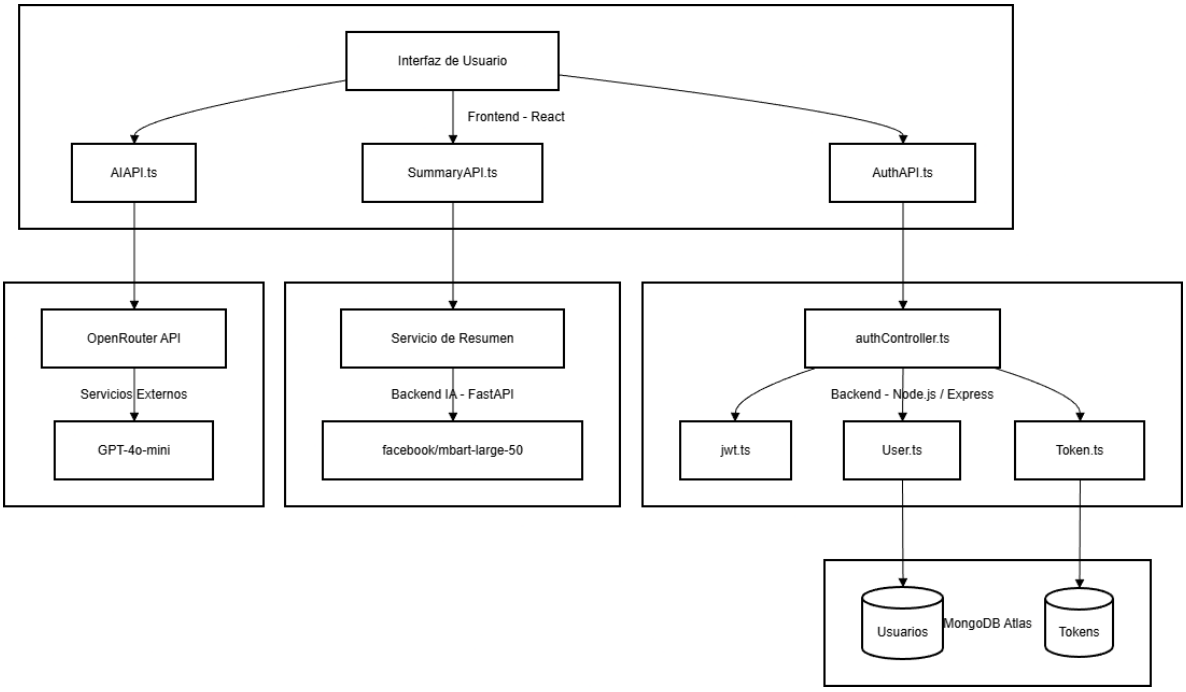
- Recibe el texto proporcionado por el usuario por medio de una petición HTTP.
- Realiza una limpieza básica del contenido para eliminar caracteres innecesarios.
- Procesa la información con el modelo.
- Genera y devuelve un resumen final al frontend.

Este backend funciona de manera independiente al sistema de autenticación, lo cual mejora la organización, la escalabilidad y el mantenimiento del sistema en general.

OpenRouter + GPT-4

OpenRouter funciona como intermediario para acceder a modelos de lenguaje como GPT-4. Este servicio:

- Recibe las solicitudes del frontend.
- Procesa el texto utilizando modelos avanzados.
- Devuelve las correcciones del ensayo al sistema.



Elementos de Ingeniería de Software

El sistema cumple con:

- Cohesión: cada módulo cumple una función específica
- Bajo acoplamiento: los módulos son independientes entre sí
- Seguridad: datos sensibles protegidos con hash y tokens

Seguridad implementada:

1. Encriptación de contraseñas (bcrypt): Todas las contraseñas son cifradas antes de ser almacenadas.
2. Autenticación por JWT: Se genera un token que permite acceder a rutas protegidas sin exponer información sensible.
3. Tokens temporales (6 dígitos): Se utilizan para confirmar cuentas y restablecer contraseñas, con vencimiento automático de 10 minutos.
4. Envío de correo electrónico: Mediante Nodemailer se envían notificaciones seguras con tokens únicos.

Sistemas Inteligentes

El sistema desarrollado es capaz de analizar un texto extenso en español y generar un resumen coherente y significativo. Para lograr esto, se utiliza un modelo de red neuronal tipo Transformer, el cual ha sido entrenado previamente para realizar tareas de resumen automático en múltiples idiomas.

Aunque se utilizó un modelo preentrenado (facebook/mbart-large-50), el desarrollo del sistema implicó un trabajo de integración, adaptación y optimización. Se diseñó una API propia en FastAPI, encargada de realizar el preprocesamiento del texto, la segmentación inteligente en fragmentos, la ejecución del modelo de resumen y un proceso adicional de refinamiento. Asimismo, se implementaron validaciones, control de longitud y métricas de evaluación, lo que convirtió la solución en un sistema inteligente funcional diseñado específicamente para la generación automática de resúmenes.

El backend (FastAPI) actúa como un microservicio inteligente que recibe información desde el frontend, la procesa mediante el modelo de IA y devuelve un resumen optimizado.

La tarea principal que realiza este sistema es la generación de resúmenes abstractive, lo que significa que no se limita a copiar frases del texto original, sino que comprende el significado y genera nuevas oraciones que representan la esencia del contenido.

Modelo matemático del sistema de resumen automático

El modelo utilizado, facebook/mbart-large-50, está basado en la arquitectura Transformer, la cual se compone principalmente por un codificador (encoder) y un decodificador (decoder).

Representación del texto

Sea un texto de entrada definido como una secuencia de palabras:

$$X = (x_1, x_2, x_3, \dots, x_n)$$

Cada palabra es convertida a un vector numérico mediante un proceso llamado embeddings, obteniendo:

$$E = (e_1, e_2, e_3, \dots, e_n)$$

Estos vectores contienen información semántica del texto.

Además, se agregan codificaciones posicionales para que el modelo conozca el orden de las palabras:

$$Z = E + P$$

Donde:

- E = embeddings de palabras
- P = codificaciones posicionales
- Z = entrada final al modelo

Mecanismo de Atención (Self-Attention)

El Transformer utiliza un mecanismo llamado self-attention, el cual permite calcular la importancia de cada palabra con respecto a las demás:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Donde:

- Q = Query (consulta)
- K = Key (clave)
- V = Value (valor)
- d_k = dimensión de los vectores

Este mecanismo permite que el modelo entienda el contexto global del texto, identificando qué palabras son más relevantes al momento de generar el resumen.

Función objetivo

Durante el entrenamiento, el modelo busca maximizar la probabilidad de generar el resumen correcto:

$$P(Y|X)$$

Donde:

- X es el texto original
- Y es el resumen generado

La función de pérdida utilizada suele ser la entropía cruzada:

Esto permite ajustar los pesos del modelo para que cada vez genere resúmenes más precisos y coherentes.

Justificación del modelo

Para este proyecto se eligió el modelo facebook/mbart-large-50 debido a sus numerosas ventajas frente a otras alternativas disponibles en el área del procesamiento de lenguaje natural.

Este modelo fue entrenado con más de 50 idiomas, incluido el español, lo que lo convierte en una opción ideal para trabajar con textos en este idioma sin necesidad de realizar un entrenamiento adicional. Además, cuenta con una alta capacidad para generar resúmenes del tipo *abstractive*, es decir, no solo selecciona frases del texto original como ocurre en los enfoques extractivos, sino que crea nuevas oraciones más naturales, coherentes y representativas de las ideas principales.

Su arquitectura basada en Transformers le permite utilizar mecanismos de atención y comprensión contextual profunda, lo que facilita el entendimiento de relaciones semánticas complejas, ideas completas y estructuras gramaticales dentro del texto.

Comparación con otras opciones

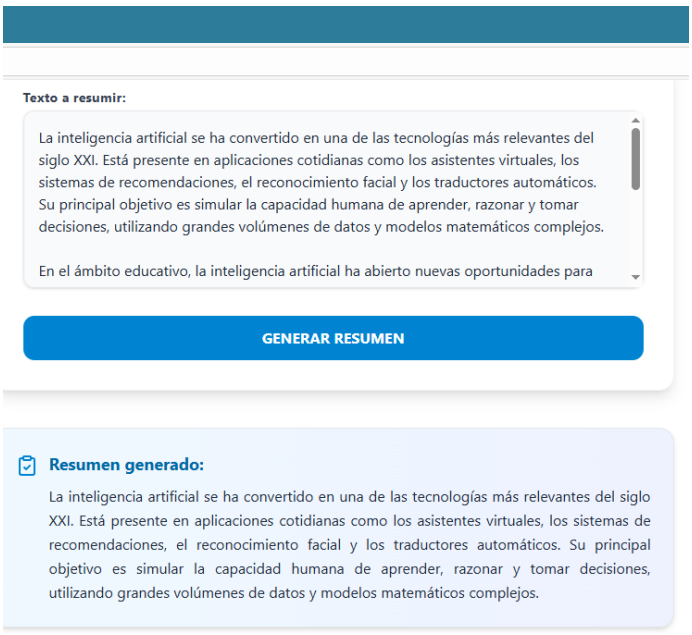
Modelo	**Desventajas frente a facebook/mbart-large-50**
BERT	No está diseñado para generación de texto, solo para comprensión (clasificación, análisis, embeddings, etc.). No puede crear resúmenes por sí mismo.
GPT básico	Es un modelo generativo general, pero no está optimizado específicamente para tareas de resumen y puede generar texto innecesario o poco enfocado.
T5	Tiene excelente desempeño en resumen, sin embargo, requiere mayor capacidad de cómputo y configuración más compleja para obtener resultados óptimos
TextRank	Método extractivo sin redes neuronales profundas. Solo selecciona frases del texto original, sin generar contenido nuevo ni reformular ideas.
DistilBART (Hugging Face)	Es una versión comprimida de BART. Aunque es más rápida, pierde precisión y calidad en textos largos o complejos comparado con mBART-50, especialmente en contextos multilingües, como se observó en la versión anterior del proyecto.

mBART ofrece un balance entre calidad, rendimiento y facilidad de implementación, lo que lo convierte en la opción más adecuada para el sistema desarrollado.

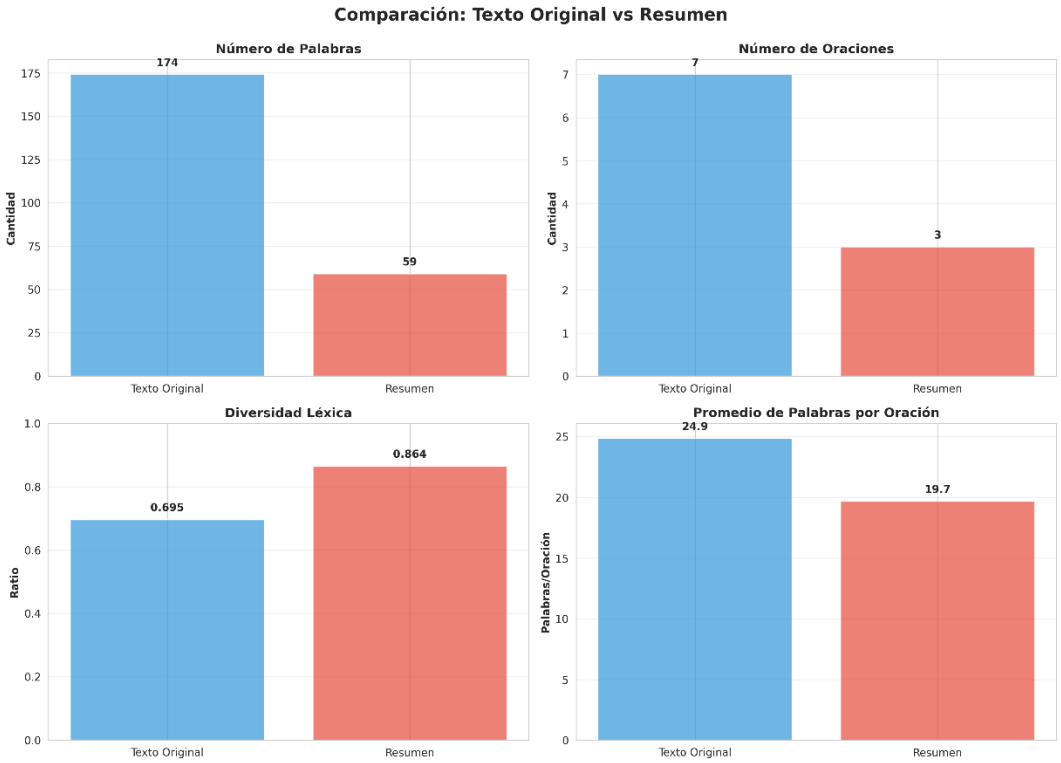
Resultados

Tras realizar pruebas con diferentes textos de distintas longitudes (pequeño, medio largo), el sistema obtuvo las siguientes métricas:

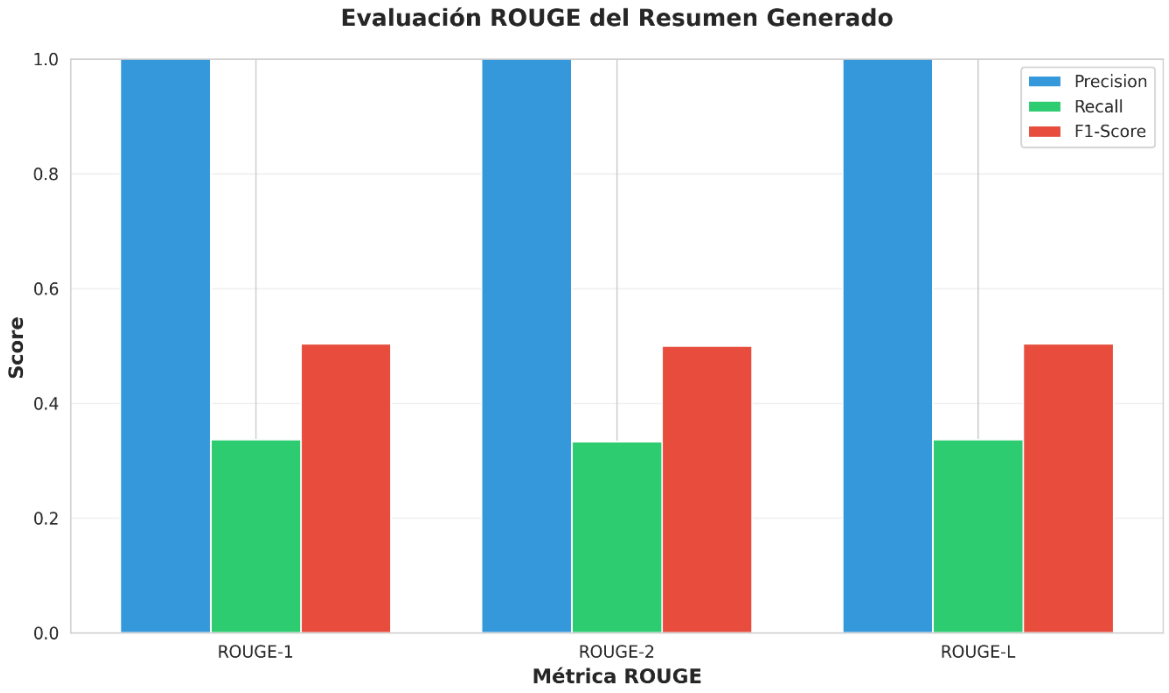
Texto Corto (≈ 100–200 palabras)

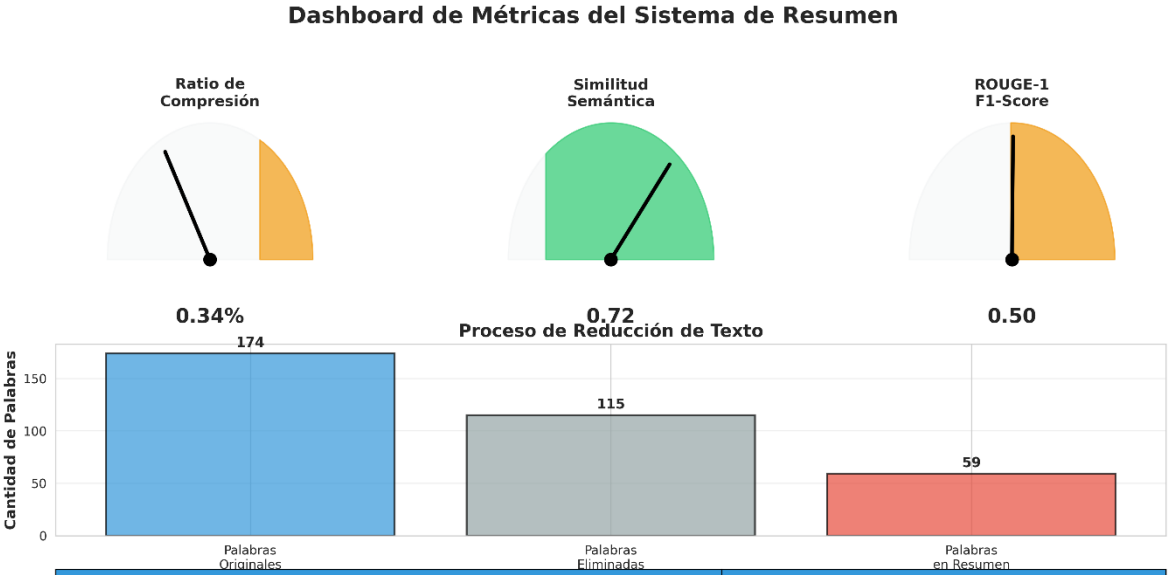


En la prueba realizada con un texto corto, el modelo logró una reducción significativa del contenido sin perder la coherencia general del mensaje. El texto original contenía 174 palabras distribuidas en 7 oraciones, mientras que el resumen generado fue reducido a 59 palabras y 3 oraciones, lo que representa una compresión del 66.1%. Esto demuestra la capacidad del sistema para sintetizar información de forma eficiente.



La diversidad léxica aumentó de 0.695 en el texto original a 0.864 en el resumen, lo que indica que, a pesar de ser más breve, el contenido generado mantiene una riqueza de vocabulario considerable y evita la repetición excesiva de palabras.





Métrica	Valor
Tiempo de procesamiento	43.24 seg
Velocidad	4.0 palabras/seg
Chunks procesados	1
Reducción de texto	66.1%
ROUGE-2 F1	0.500
ROUGE-L F1	0.504

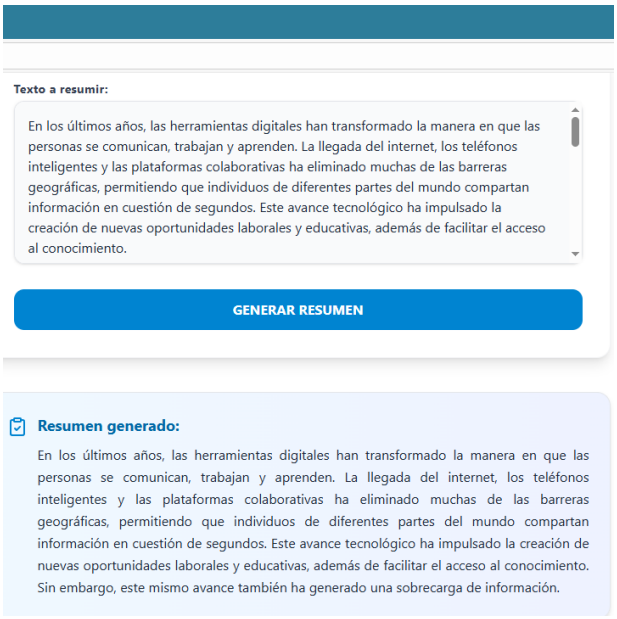
En cuanto a las métricas ROUGE, se obtuvieron valores de F1 cercanos a 0.50 en ROUGE-1, ROUGE-2 y ROUGE-L, lo cual indica una relación aceptable entre el contenido del

resumen y el texto original, logrando capturar las ideas principales sin copiar grandes fragmentos textuales. Además, la similitud coseno de 0.719 confirma que el resumen conserva un alto nivel de relación semántica con el contenido fuente.

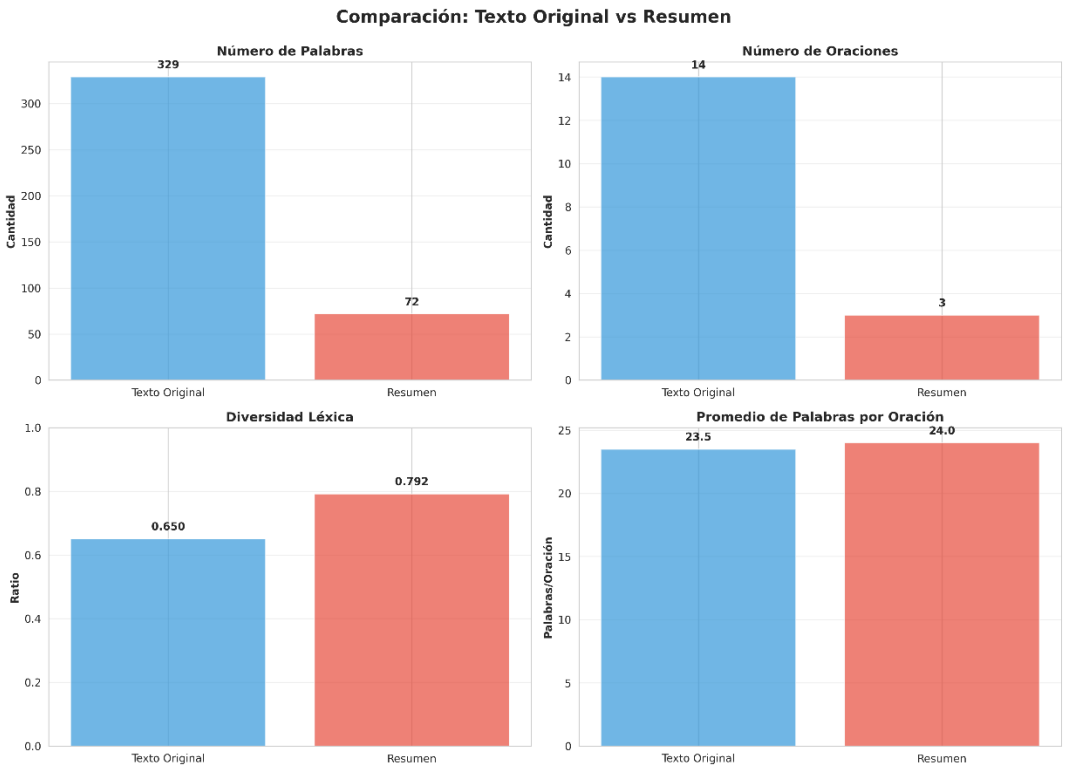
Respecto al rendimiento, el proceso tardó 43.24 segundos, con una velocidad aproximada de 4 palabras por segundo, lo que, aunque puede considerarse moderado, es aceptable considerando el uso de un modelo de arquitectura Transformer de gran tamaño y alta capacidad semántica.

En conjunto, estos resultados muestran que el sistema es capaz de generar resúmenes concisos, coherentes y semánticamente relevantes, cumpliendo con el objetivo principal del proyecto: la automatización de la generación de resúmenes mediante inteligencia artificial.

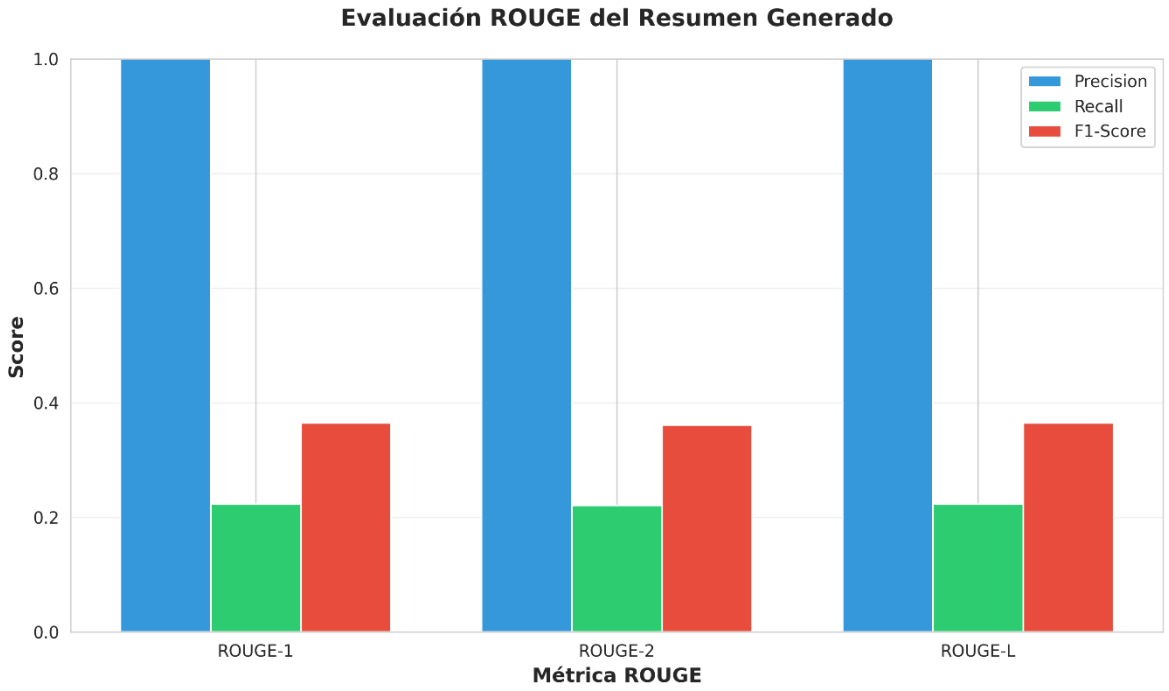
Texto Medio (≈ 250–500 palabras)

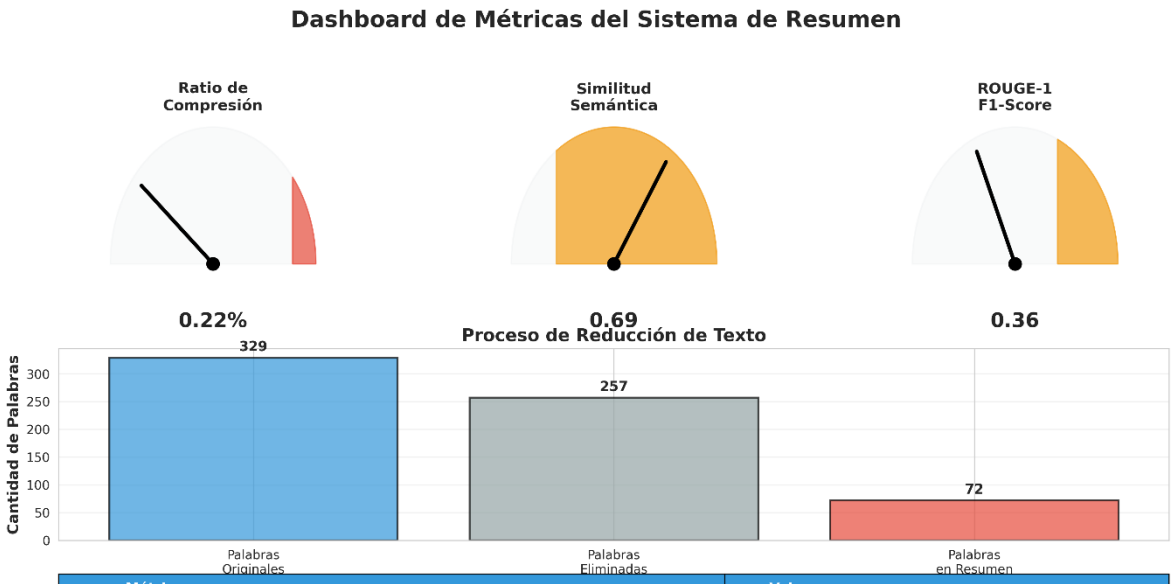


En la prueba realizada con un texto de longitud media, el sistema mostró un desempeño eficiente en la reducción y síntesis de la información. El texto original contenía 329 palabras distribuidas en 14 oraciones, mientras que el resumen generado fue reducido a 72 palabras y 3 oraciones, lo que representa una compresión del 78.1%. Esto evidencia la capacidad del modelo para eliminar información redundante y conservar únicamente los puntos más relevantes.



La diversidad léxica pasó de 0.650 en el texto original a 0.792 en el resumen, lo que indica que, a pesar de la reducción significativa en tamaño, el resumen mantiene una variedad adecuada de vocabulario y no cae en repeticiones excesivas.





Métrica	Valor
Tiempo de procesamiento	60.16 seg
Velocidad	5.5 palabras/seg
Chunks procesados	1
Reducción de texto	78.1%
ROUGE-2 F1	0.362
ROUGE-L F1	0.364

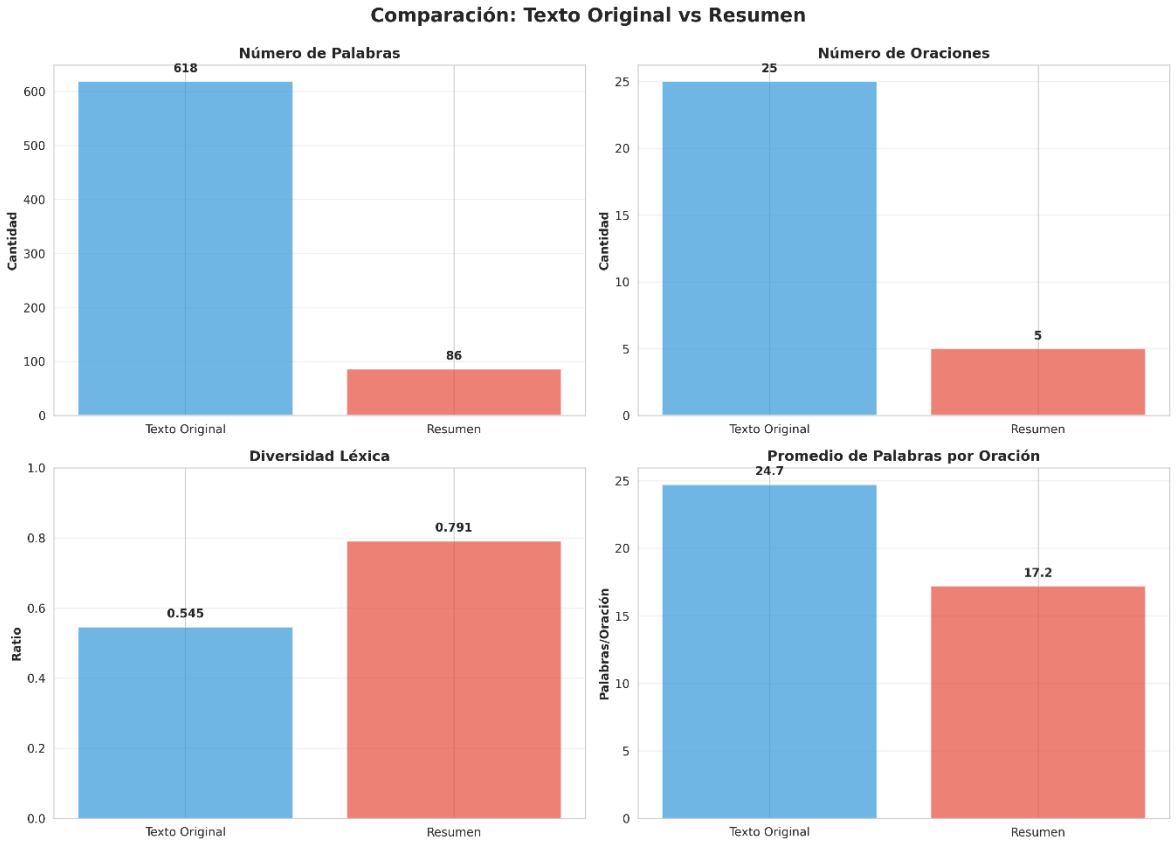
En cuanto a las métricas ROUGE, se obtuvieron valores de F1 alrededor de 0.36 para ROUGE-1, ROUGE-2 y ROUGE-L. Estos resultados indican que el modelo logra capturar las ideas principales del texto, aunque con menor coincidencia exacta de términos. La similitud coseno de 0.690 confirma que el contenido del resumen conserva una relación semántica considerable con el texto original.

Respecto al rendimiento, el proceso tuvo una duración de 60.16 segundos, con una velocidad aproximada de 5.5 palabras por segundo, mostrando un desempeño ligeramente superior en términos de velocidad comparado con la prueba del texto corto, lo cual puede atribuirse a una mejor distribución del contexto en el procesamiento del modelo.

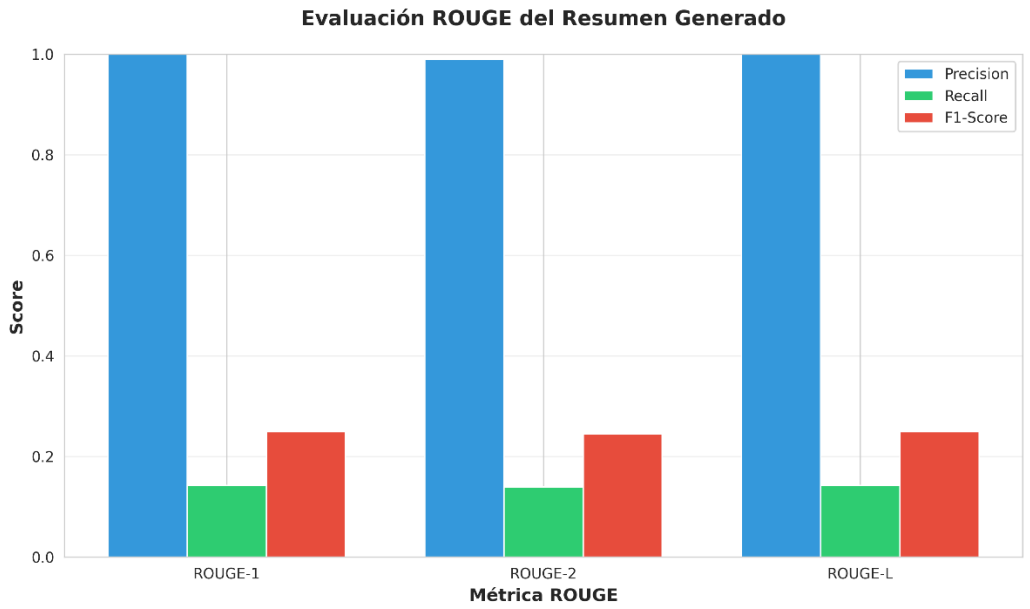
En conjunto, estos resultados demuestran que el sistema se mantiene estable y eficaz al trabajar con textos de mayor longitud, generando resúmenes concisos, coherentes y semánticamente relevantes, lo que valida su utilidad para documentos de tamaño medio en escenarios reales.

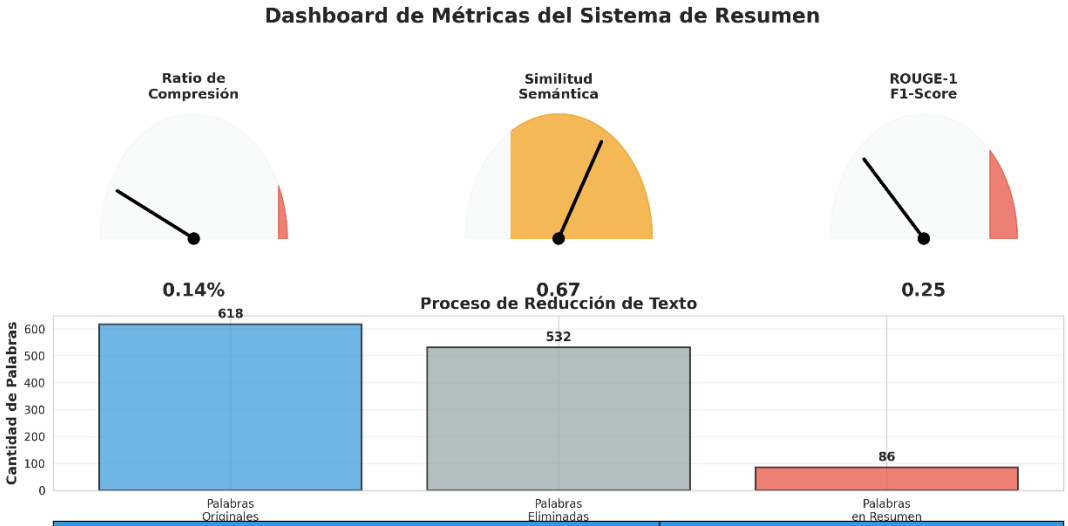
Texto Largo (≈ 600–1000 palabras)

En la prueba realizada con un texto largo, el sistema demostró una alta capacidad de compresión manteniendo la coherencia general del contenido. El texto original contenía 618 palabras distribuidas en 25 oraciones, mientras que el resumen generado se redujo a 86 palabras en 5 oraciones, lo que representa una reducción del 86.1%.



La diversidad léxica aumentó de 0.545 en el texto original a 0.791 en el resumen, lo que indica una mejora significativa en la variabilidad del vocabulario utilizado. Esto indica que el modelo no solo acorta el texto, sino que también reorganiza la información de forma más compacta y eficiente, utilizando términos más variados.





Métrica	Valor
Tiempo de procesamiento	148.47 seg
Velocidad	4.2 palabras/seg
Chunks procesados	2
Reducción de texto	86.1%
ROUGE-2 F1	0.245
ROUGE-L F1	0.250

En cuanto a las métricas ROUGE, los valores obtenidos se situaron alrededor de 0.25 en F1 para ROUGE-1, ROUGE-2 y ROUGE-L. Aunque estos valores son más bajos en comparación con los textos corto y medio, esto es esperable debido al mayor nivel de abstracción aplicado al resumir textos más extensos. A pesar de ello, la similitud coseno de 0.670 confirma que el resumen mantiene una relación semántica sólida con el contenido original, preservando las ideas principales del documento.

En términos de rendimiento, el tiempo de procesamiento fue de 148.47 segundos, con una velocidad aproximada de 4.2 palabras por segundo. Este incremento en el tiempo es coherente con la longitud del texto de entrada y el proceso de división en fragmentos (chunks), necesario para evitar los límites de entrada del modelo. Aun así, el sistema se mantuvo estable durante todo el proceso.

En conjunto, estos resultados demuestran que el sistema basado es capaz de manejar eficientemente textos largos, generando resúmenes altamente comprimidos, coherentes y semánticamente representativos del contenido original, lo que valida su implementación para escenarios donde se requiera sintetizar grandes volúmenes de información de forma automatizada.

Sistemas Distribuidos

El sistema desarrollado corresponde a una aplicación web con una arquitectura distribuida, ya que sus componentes principales se encuentran separados física y lógicamente, estando desplegados en distintas plataformas en la nube. Esta estructura permite dividir las responsabilidades del sistema en múltiples servicios independientes que se comunican entre sí mediante protocolos de red estandarizados.

El frontend, desarrollado con React, se encuentra desplegado en la plataforma Render y es el encargado de ofrecer la interfaz gráfica para los usuarios, permitiéndoles registrarse, iniciar sesión, enviar textos para resumir y solicitar la corrección de ensayos. Este componente funciona como el cliente principal del sistema, desde donde se originan las peticiones hacia los distintos servicios.

El backend principal, construido con Node.js y Express y también desplegado en Render, se encarga de la lógica de negocio de la aplicación. Entre sus funciones se encuentran la

autenticación de usuarios, la administración de sesiones mediante tokens, la recuperación de contraseñas a través de correo electrónico y la conexión con la base de datos alojada en MongoDB Atlas, la cual se encuentra en un servidor independiente en la nube. Esta separación convierte a la base de datos en otro nodo distribuido dentro del sistema.

Adicionalmente, se implementó un microservicio independiente en FastAPI, desarrollado en Python y desplegado en Hugging Face Spaces, el cual utiliza el modelo de inteligencia artificial facebook/mbart-large-50 para la generación de resúmenes automáticos. Este servicio no forma parte directa del backend principal, sino que opera como un componente desacoplado, especializado únicamente en tareas de procesamiento de lenguaje natural. De esta manera, el cálculo pesado es delegado a un servidor externo, distribuyendo así la carga computacional del sistema.

Por otro lado, la corrección de ensayos se realiza a través de la integración de OpenRouter, utilizando el modelo GPT-4o Mini, el cual funciona como un servicio externo adicional. Este componente también representa un nodo distribuido, ya que se encuentra alojado en una infraestructura distinta y se accede a él mediante su API.

En conjunto, el sistema está compuesto por los siguientes nodos distribuidos:

- Frontend en React desplegado en Render
- Backend en Node.js / Express desplegado en Render
- Base de datos en MongoDB Atlas
- Microservicio de resúmenes en FastAPI desplegado en Hugging Face Spaces
- Servicio de corrección de ensayos mediante OpenRouter (GPT-4o Mini)

Este diseño permite distribuir las tareas del sistema, favoreciendo la escalabilidad, la tolerancia a fallos ya que no todo depende de un solo servidor y la posibilidad de modificar o mejorar cada componente de manera independiente sin afectar a los demás.

Justificación de los protocolos de comunicación involucrados

La comunicación entre los distintos componentes del sistema distribuido se realiza principalmente mediante el protocolo HTTP/HTTPS, siguiendo el modelo de arquitectura REST (Representational State Transfer).

La interacción entre el frontend (React) y el backend (Node.js y Express) se lleva a cabo mediante solicitudes HTTP/HTTPS del tipo GET, POST, PUT y DELETE. Estas peticiones permiten realizar acciones como el registro de usuarios, el inicio de sesión, el envío de textos, la solicitud de resúmenes y la corrección de ensayos. Este esquema responde al modelo cliente-servidor, en el cual el navegador actúa como cliente y el backend como servidor encargado de procesar la información.

La comunicación entre el frontend y el microservicio en FastAPI (Hugging Face Spaces) también se realiza mediante una API REST sobre HTTP/HTTPS. Cuando un usuario solicita un resumen, se envía el texto al microservicio, este procesa la información utilizando el modelo facebook/mbart-large-50 y devuelve el resumen generado. Esta separación permite que el procesamiento de inteligencia artificial se realice en un servidor independiente, lo que es un ejemplo de distribución del procesamiento de cálculos.

De manera similar, la conexión entre el frontend y OpenRouter (GPT-4o Mini) se realiza a través de peticiones HTTPS hacia su API. Esto permite que la corrección de ensayos se lleve a cabo en una infraestructura externa, sin necesidad de ejecutar modelos complejos dentro del propio servidor, y esto optimiza el uso de recursos locales.

El uso de HTTPS en lugar de HTTP es importante para garantizar la seguridad del sistema, ya que los datos transmitidos entre los distintos servicios son cifrados, evitando que terceros puedan interceptarlos o modificarlos.

Además, se emplea el mecanismo de CORS (Cross-Origin Resource Sharing) en el backend para permitir que el frontend, aun estando alojado en un dominio distinto (Render), pueda realizar peticiones de manera segura al servidor.