Sistema RAG de Asistencia a Codebase
Ejercicio 2

73.64 Temas Avanzados en Deep Learning

Bengolea, Iñaki (63515), ... ibengolea@itba.edu.ar
Casella, Nicolas Dario (62311), ... ncasella@itba.edu.ar
Magliotti, Gianfranco (61172), ... gmagliotti@itba.edu.ar

13 de Octubre, 2025

1. Responsible AI y Safety

Al manipular código fuente provisto por usuarios, el sistema debe garantizar protección, seguridad e intimidad mediante principios de Responsible AI. Se excluyen del procesamiento archivos con información sensible (archivos .env, credenciales) mediante análisis estático que redacta y anonimiza el código, eliminando secretos, PII y código bajo licencias restrictivas. Esta medida preventiva se ejecutara antes de la persistencia en la base vectorial. El código se almacenara exclusivamente con propósito de retrieval contextual, sin utilizarse para entrenamiento de modelos. El acceso a repositorios privados requerirá consentimiento explícito del usuario, respetando propiedad intelectual y confidencialidad.

Debemos asegurarnos de que las respuestas sean puramente relacionadas a las preguntas del usuario que referencien a la codebase del mismo. Se hará mucho énfasis de esto en el prompt del sistema, remarcando que solo debe responder en relación a los fragmentos de código conseguidos. Adicionalmente, se le especificara 2 tokens especiales, uno que determina el comienzo y fin del prompt del usuario, y otro el comienzo y fin de los fragmentos de código recuperados, evitando así prompts maliciosos como intentos de jailbreaks y de prompt injection, mejorando así la robustez del sistema.

Las respuestas generadas se complementaran sistemáticamente con referencias al archivo y número de línea del código utilizado como contexto, fortaleciendo la explicabilidad y la trazabilidad del modelo. La persistencia de logs captura queries, fragmentos consultados y respuestas anonimizadas, posibilitando auditorías exhaustivas y permitiendo identificar patrones problemáticos o degradación en la calidad de respuestas.

El sistema implementara advertencias explícitas respecto a limitaciones inherentes de la inteligencia artificial. Los usuarios deben comprender que el modelo es una herramienta de asistencia, no sustituto del juicio humano para decisiones de implementación. Las salidas deben tratarse como sugerencias que requieren revisión humana, especialmente en el caso que la salida del sistema sea código generado con intención de usarse en un entorno de producción. El sistema incorporara auditoría periódica de sus propias dependencias, mitigando riesgos de supply-chain attacks mediante el principio de seguridad por diseño.

2. Estrategias de Deployment

La implementación en producción requiere una arquitectura basada en contenedores estructurada en tres componentes desplegados como servicios independientes. Al recibir una codebase por parte del usuario, se dispara el indexing pipeline, extrayendo el código brindado por el usuario. En caso de que el usuario provea una URL de un repositorio de Github, se realizara una petición a la API del mismo para obtener el archivo .zip del repositorio.

El servicio de ingesta ejecuta el pipeline de detección de lenguaje, parseo y fragmentación del código. El resultado de la fragmentación sera guardada en un bucket S3 y enviado a una queue para que pueda ser procesado por el embedder, que generara los embeddings y la metadata que sera persistida en la base de datos vectorial, de forma asincrónica. La base

de datos vectorial ChromaDB se configurara en modo servidor persistente con volúmenes que garanticen supervivencia de colecciones, permitiendo compartir la base vectorial entre múltiples instancias del servicio de query. El modulo de LLM orquesta las llamadas a los modelos de lenguaje externos (GPT o Gemini). Una vez generada la respuesta del modelo de lenguaje, se la envía al usuario a través de la API Gateway, para poder mostrarse en el frontend.

Usando Amazon Cloudwatch, se loggearan principalmente los prompts de los usuarios de forma anónima, fragmentos consultados y respuestas de la LLM, excluyendo todo dato considerado privado o sensible. Ademas, usaremos Cloudwatch para mantener métricas adicionales relacionadas con la aplicación, como uso de recursos, latencia y tokens utilizados.

El análisis estático y detección de secretos mediante herramientas como truffleHog se ejecuta antes de persistir fragmentos, rechazando procesamiento de repositorios comprometidos. Para repositorios privados, se implementa multi-tenancy estricto mediante colecciones segregadas con controles de acceso que impidan queries cross-tenant, complementándolo con autenticación mediante OAuth 2.0 o SAML.

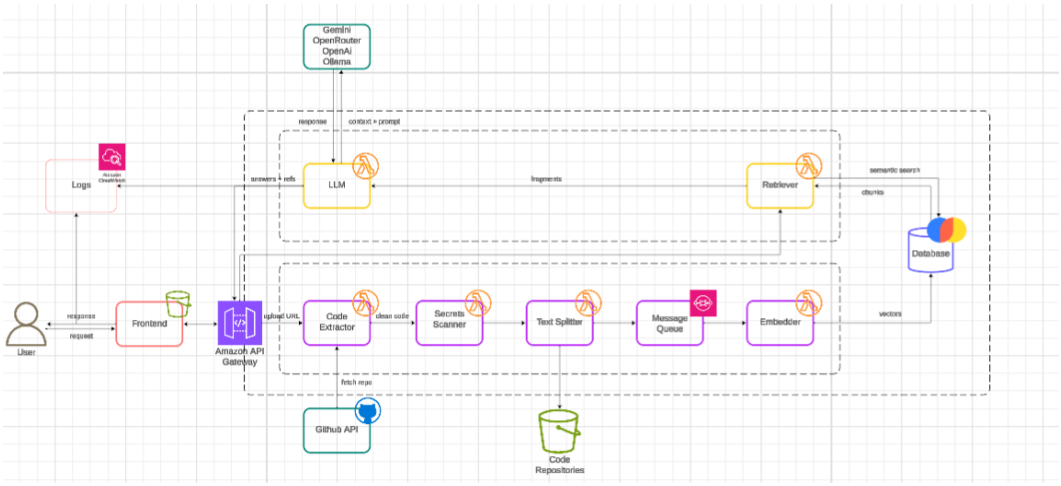

Figura 1: Diagrama detallado de la arquitectura