Documentación técnica – Proyecto Entrega 1 – SaaS

Integrantes grupo 6:

- Santiago Osorio
- Carlos Peñuela
- David Segura

1. Historias de usuario con criterios de aceptación

Identificador	HU01
Nombre	Subir documentos a la plataforma
Descripción	**Como** usuario registrado **Quiero** cargar documentos en la plataforma **Para** poder analizarlos mediante la IA
Criterios de aceptación	El usuario puede cargar archivos en formatos PDF, TXT, DOCX.
	La plataforma muestra un mensaje de éxito tras la carga.
	Se establece un límite de tamaño para los documentos (ej. 10MB).
	Los archivos cargados son almacenados en la nube. (Próximamente)
	Se notifica al usuario si el formato no es soportado.

Criterio	Comentario
Independiente	No depende de otras historias.
Negociable	El formato y capacidad máxima de los archivos pueden ser ajustados según requisitos técnicos.
Valiosa	Permite al usuario interactuar con la IA mediante sus propios documentos.
Estimable	Puede implementarse en un sprint
Pequeña	Se enfoca solo en la carga de archivos.
Testeable	Se valida con pruebas de carga y visualización de documentos.

Identificador	HU02
Nombre	Generar resúmenes de documentos
Descripción	**Como** usuario registrado **Quiero** obtener un resumen de mis documentos cargados **Para** comprender rápidamente su contenido sin leer todo el texto
Criterios de aceptación	El usuario puede solicitar un resumen del documento cargado.
	El sistema genera un resumen claro y estructurado.

	El resumen es accesible desde la interfaz web.
	Se notificará si ocurre un error en el procesamiento.

Criterio	Comentario
Independiente	Se basa en la HU01 pero no requiere otras funcionalidades.
Negociable	El nivel de detalle del resumen puede ajustarse según configuración.
Valiosa	Ahorra tiempo al usuario en la lectura de documentos extensos.
Estimable	Puede desarrollarse en un sprint usando un LLM preentrenado.
Pequeña	Se enfoca solo en la generación de resúmenes.
Testeable	Se valida con pruebas de entrada/salida y comparación con resúmenes manuales.

Identificador	HU03
Nombre	Explicar conceptos clave del documento
Descripción	**Como** usuario registrado **Quiero** obtener explicaciones detalladas sobre conceptos dentro de mis documentos **Para** mejorar mi comprensión del contenido sin necesidad de investigación externa
Criterios de aceptación	El usuario puede seleccionar términos específicos para obtener explicaciones.
	La IA proporciona explicaciones detalladas y comprensibles.
	El sistema detecta automáticamente conceptos clave y los sugiere al usuario.
	Se permite copiar las explicaciones generadas.

Criterio	Comentario
Independiente	Funciona sin necesidad de la HU02.
Negociable	La cantidad de explicaciones detalladas puede ser configurable.
Valiosa	Enriquece la experiencia de aprendizaje del usuario.
Estimable	Implementable en un sprint usando NLP.
Pequeña	Se limita a explicar términos del documento.
Testeable	Se verifica con consultas específicas y validación de respuestas.

Identificador	HU04
Nombre	Responder preguntas sobre el contenido del documento

Descripción	**Como** usuario registrado **Quiero** hacer preguntas sobre el contenido de mis documentos **Para** recibir respuestas precisas y contextualizadas
Criterios de aceptación	El usuario puede escribir preguntas sobre el contenido del documento.
	La IA responde basándose únicamente en la información del documento cargado.
	Se notificará al usuario si la pregunta no tiene respuesta en el documento.

Criterios INVEST HU04

Criterio	Comentario
Independiente	No depende de la generación de resúmenes ni explicaciones.
Negociable	Se puede ajustar el nivel de profundidad de las respuestas..
Valiosa	Permite al usuario extraer información específica sin leer el documento completo.
Estimable	Implementable en un sprint con un LLM adecuado.
Pequeña	Se enfoca en la funcionalidad de preguntas y respuestas.
Testeable	Se verifica con preguntas de prueba y validación de respuestas.

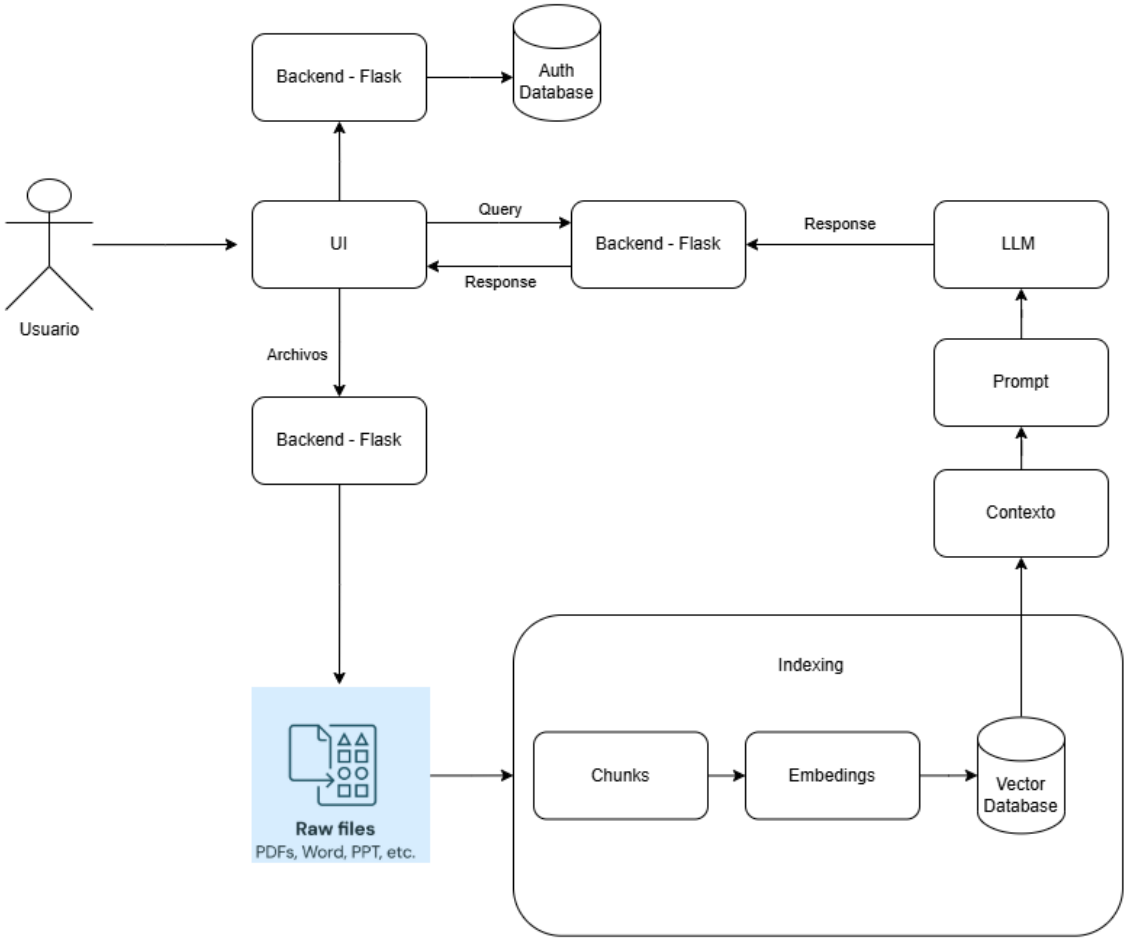
Identificador	HU05
Nombre	Autenticación de usuarios
Descripción	**Como** usuario **Quiero** registrarme e iniciar sesión en la plataforma **Para** gestionar mis documentos de forma segura
Criterios de aceptación	El usuario puede registrarse con email y contraseña.
	Se muestra un mensaje de error si las credenciales son incorrectas.

Criterios INVEST HU05

Criterio	Comentario
Independiente	No requiere la ejecución de procesos de IA.
Negociable	Se pueden incluir filtros o etiquetas.
Valiosa	Facilita la organización de documentos.
Estimable	Implementable en un sprint con una base de datos simple.
Pequeña	Se limita a la gestión de archivos subidos.
Testeable	Se valida con pruebas de UI y consultas de base de datos.

2. Diagramas de despliegue y arquitectura

   A continuación, se presenta de forma breve la arquitectura utilizada para la aplicación.



Ahora, respecto a la arquitectura la aplicación RAG SaaS ha sido desarrollada con una arquitectura basada en contenedores, integrando diversas tecnologías para la autenticación de usuarios, almacenamiento de documentos, generación de embeddings y procesamiento de consultas mediante un modelo de lenguaje de gran escala (LLM). La implementación actual está enfocada en ejecución local, con miras a una transición a una infraestructura en la nube para mejorar el rendimiento y escalabilidad.

Componentes Principales:

Frontend: Aplicación web desarrollada en React para la interacción con los usuarios.

Backend: API en Flask que gestiona autenticación, almacenamiento y procesamiento de documentos.

Autenticación: Sistema basado en JSON Web Tokens (JWT) y almacenamiento en una base de datos relacional.

Base de Datos:

- Relacional (SQLite): Para la autenticación y gestión de usuarios.
- Vectorial (ChromaDB): Para almacenar embeddings de documentos y facilitar la recuperación aumentada generativa (RAG).

Modelo de Lenguaje (LLM): Implementado con LLaMA 8B, operando en un entorno local.

Procesamiento de Archivos: Integración con un servicio que convierte documentos en texto y los indexa en la base de datos vectorial.

Para garantizar que la aplicación pueda atender cientos de usuarios concurrentes, es necesario realizar una migración progresiva a la nube y optimizar los componentes actuales.

Se recomienda implementar los siguientes servicios cloud: PostgreSQL en Amazon RDS o Google Cloud SQL para escalabilidad automática en autenticación y base de datos; Amazon S3 o Google Cloud Storage para reducir la carga en el servidor backend en almacenamiento de documentos; Pinecone, Weaviate o una instancia escalable de ChromaDB en la nube para embeddings y base de datos vectorial; Azure OpenAI, Anthropic Claude o Google Vertex AI en lugar de modelos locales para el procesamiento con LLM; y Kubernetes en Google Kubernetes Engine (GKE) o AWS EKS para manejar escalabilidad horizontal en la orquestación de contenedores.

Para optimizar el procesamiento, se recomienda mejorar la comunicación con el LLM, ya que actualmente las respuestas del modelo son lentas y pierden contexto. Se sugiere implementar cacheo de respuestas con Redis o Memcached, utilizar context windows optimizados con técnicas de Sliding Window o Memory Augmented Generation, y evaluar modelos más robustos con mayor capacidad de contexto como GPT-4 Turbo o Claude 3.

En términos de balanceo de carga, se recomienda implementar un API Gateway como AWS API Gateway, Kong o Nginx para distribuir tráfico eficientemente, y

escalar instancias de backend con Auto Scaling Groups (ASG) en AWS o Horizontal Pod Autoscaler (HPA) en Kubernetes.

Para monitoreo y logging, se sugiere integrar herramientas como Prometheus y Grafana para métricas de rendimiento e implementar centralización de logs con Elasticsearch y Kibana (ELK Stack).

Durante el desarrollo en entorno local, se han identificado varias limitaciones. La comunicación con el LLM es lenta, ya que el modelo LLaMA 8B ejecutado localmente presenta retardos y pérdida de contexto en respuestas largas. La capacidad de cómputo limitada restringe el procesamiento eficiente de múltiples consultas concurrentes. Implementar RAG con modelos pequeños es difícil, ya que el contexto limitado de modelos como LLaMA 8B afecta la precisión y utilidad de las respuestas.

La escalabilidad está restringida, pues la arquitectura actual no permite manejar cientos de usuarios simultáneamente sin degradación del rendimiento. No se han implementado estrategias de almacenamiento en caché para consultas frecuentes, lo que afecta la eficiencia. Además, existe una dependencia de almacenamiento local, ya que la carga de documentos y embeddings actualmente se gestiona en servidores locales, lo cual no es escalable.

Para garantizar un sistema escalable y eficiente, se recomienda migrar progresivamente a la nube, optimizar la comunicación con el LLM y mejorar la infraestructura de almacenamiento y recuperación de información. La próxima fase del desarrollo deberá enfocarse en la implementación de una arquitectura serverless o basada en microservicios en la nube, la optimización del procesamiento con modelos LLM más robustos y con mayor contexto, la mejora de la velocidad de respuesta mediante caching y balanceo de carga, y el monitoreo y observabilidad para garantizar un rendimiento óptimo en producción.

Esta evolución permitirá que RAG SaaS pueda atender eficientemente cientos de usuarios de manera concurrente, asegurando una experiencia fluida y confiable.

3. Documentación de los servicios del API REST

   La documentación de los servicios del API REST se encuentran en la carpeta postman en el repositorio de GitHub, esta contiene una colección de solicitudes para probar los endpoints que provee el backend.

4. Instrucciones para la ejecución y despliegue

Las instrucciones se encuentran en el readme.md del repositorio.