E1-Plan de análisis de capacidad

1. Entorno de prueba

Infraestructura de producción

- **Servidor web**: Golang - Echo
- **Servicio de colas:** RabbitMQ
- **Base de datos**: PostgreSQL 15 + pg_vector
- **Sistema operativo**: Manjaro 25.0.0 Zetar
- **Servicio RAG:** Sentence transformer "all-MiniLM-L6-v2"
- **Servicio de asistente:** Ollama 0.5.13

Características de hardware

- **CPU**: AMD Ryzen 7 7840HS w/ Radeon 780M Graphics @ 16x 5.137GHz
- **Memoria RAM**: 15293MiB
- **Almacenamiento**: 469G

Limitaciones identificadas

- Capacidad de CPU durante inferencia de asistente LLM pues el procesamiento de embeddings y las inferencias del modelo serán considerablemente más lentos en CPU.
- Saturación de las colas de RabbitMQ si las consultas llegan más rápido de lo que el RAG puede procesarlas
- Desbalance entre los servicios del backend y el RAG pues el primero, al estar en Go, será considerablemente más eficiente
- El procesamiento de embeddings y recuperación de documentos puede ser intensivo en memoria, especialmente con grandes volúmenes de documentos o documentos extensos

Frameworks utilizados:

Backend: Go - Echo

Procesamiento RAG: Python 3.12.9

Cola de mensajes: RabbitMQ

Base de datos: PostgreSQL 15 + pg_vector

Librerías principales:

- Go: pqtype, bcrypt, echo, jwt, rabbitmq, sqlc

- Python: sentence-transformers (encoder “all-MiniLM-L6-v2”), psycopg_pool, numpy, aio_pika, decouple
- ORM: SQLAlchemy
- LLM utilizado: Ollama phi4 14B
- Contenedores: Docker, Docker Compose

2. Criterios de aceptación

Objetivos de tiempo de respuesta

- Páginas estáticas: Tiempo de carga < 1 segundo
- Operaciones CRUD simples: Tiempo de respuesta < 3 segundos
- Operaciones relacionadas con LLM: Tiempo de respuesta < 2 min

Objetivos de rendimiento

- **Capacidad mínima**: 100 usuarios concurrentes con tiempo de respuesta aceptable
- **Objetivo de escalabilidad**: Soportar picos de hasta 500 usuarios concurrentes
- **Tasa de transacciones**: 50 transacciones por segundo
- **Tasa de respuesta de Ollama:** 25 transacciones por segundo

Objetivos de utilización de recursos

- **CPU**: Utilización < 80% bajo carga máxima esperada
- **Memoria**: Utilización < 85% bajo carga máxima esperada
- **Red**: Utilización < 60% del ancho de banda disponible

Otros criterios de éxito

- **Disponibilidad**: 90% de uptime
- **Porcentaje máximo de error**: 10% bajo carga máxima

3. Escenarios de prueba

Escenario 1: Indexado de documentos

- **Descripción**: Evaluar el rendimiento del sistema cuando los usuarios suben documentos y estos son indexados automáticamente.
- **Perfil de usuario**: 70% sube documentos, 30% navega por documentos ya indexados.
- **Volumen**: 10 peticiones por segundo, aumentando en 2 cada 30 segundos hasta llegar al 80% del uso de CPU durante 30 segundos.
- **Duración**: 10 minutos
- **Métricas a recopilar**: Tiempos de respuesta, throughput, errores, utilización de CPU y memoria.

Escenario 2: Generación de respuestas con LLM

- **Descripción**: Medir el desempeño del sistema cuando múltiples usuarios realizan preguntas simultáneamente al asistente basado en LLM.
- **Perfil de usuario**: 100% generación de respuestas. Los usuarios deben tener cada uno por lo menos cinco documentos ya indexados
- **Volumen**: 50 peticiones concurrentes. De entre 150 y 200 tokens cada una
- **Duración**: 15 minutos
- **Métricas a recopilar**: Throughput en las colas de rabbitmq, uso de CPU y memoria, Tiempo de respuesta promedio de los tasks en RabbitMQ

Escenario 3: Recuperación de información

- **Descripción:** Evaluar la capacidad del sistema para recuperar documentos y chats de usuarios.
- **Perfil de usuario:** 50% solicita lista de documentos, 50% solicitar historial de chats.
- **Volumen:** 10 usuarios concurrentes solicitando datos.
- **Duración:** 5 minutos.
- **Métricas a recopilar:** Tiempos de respuesta, tasa de error, uso de CPU y memoria.

Escenario 4: Prueba de resistencia

- **Descripción:** Evaluar el comportamiento del sistema bajo carga sostenida con alto uso de CPU.
- **Perfil de usuario:** 10% indexa documentos, 10% solicita respuestas del asistente, 40% solicita lista de documentos, 40% solicita chats.
- **Volumen:** 20 usuarios durante 10 minutos.
- **Duración:** 10 minutos.
- **Métricas a recopilar:** Latencia, estabilidad del sistema, consumo de recursos.

Datos de prueba

- **Origen de datos**: Datos sintéticos generados para el entorno de pruebas. Acá se incluyen usuarios ficticios con credenciales de prueba. Documentos de muestra en formatos PDF, DOCX, MD y TXT e historiales de chat pregenerados para simular conversaciones
- **Perfiles de usuario**:
    1. Usuario nuevo: Usuarios realizando registro, configuración inicial y primera carga de documento.
    2. Usuario casual: Usuarios con 1-3 documentos cargados, realizando consultas básicas (1-5 por sesión)
    3. Usuario intensivo: Usuarios con 5+ documentos, realizando múltiples consultas complejas (10+ por sesión)

Configuración de APM

- **Herramienta seleccionada**: Cloud Monitoring, RabbitMQ admin

- **Métricas a monitorear**: CPU, memoria, latencia, tiempos de respuesta, longitud de las colas, *health of the queue system*.

Prerrequisitos técnicos

- Base de datos PostgreSQL con schema idéntico al de producción y datos sintéticos pre-cargados
- RabbitMQ configurado con las mismas colas y parámetros que en producción
- Instancia para el backend Go con capacidad suficiente para manejar conexiones entrantes
- Instancia para el RAG en Python optimizada para procesamiento CPU

4. Escenarios clave para próximas entregas

Escenario relacionado con la capa web

- Descripción: Simulación de flujo de usuario completo de interacción con documentos y chats con IA.
- **Pasos**:
    - Registro de nuevos usuarios.
    - Inicio de sesión de usuario
    - Carga y procesamiento de un nuevo documento.
    - Creación de nuevos chats.
    - Envío de múltiples consultas al chat (preguntas simples y complejas).
    - Visualización de respuestas generadas.
    - Eliminación de chats.
    - Eliminación de documentos.
- **Justificación**: Este escenario representa la ruta crítica principal del usuario en la aplicación, evaluando tanto la gestión de documentos como la capacidad de procesamiento de la IA para generar respuestas, involucrando operaciones intensivas de lectura, escritura y procesamiento.
- **Métricas clave (SLIs)**:
    - Tiempo de respuesta por cada paso.
    - Latencia en la generación de respuestas por parte de la IA.
    - Tasa de éxito en consultas completadas correctamente.
    - Capacidad máxima de usuarios concurrentes antes de degradación del servicio.
    - Tiempo medio entre fallos del servicio de IA.

Escenario relacionado con la capa de procesamiento por lotes

Descripción: Simulación de procesamiento asíncrono de consultas RAG

Pasos:

- Envío de múltiples consultas simultáneas al API
- Procesamiento en background mediante RabbitMQ

- Ejecución de recuperación (retrieval) y generación de respuestas en Python
- Devolución de resultados procesados al usuario

Justificación: Este escenario evalúa la capacidad del sistema para manejar consultas asíncronas mediante el sistema de colas RabbitMQ, que es crítico para mantener la responsividad y escalabilidad cuando múltiples usuarios interactúan simultáneamente con documentos.

Métricas clave (SLIs):

- Tiempo de procesamiento end-to-end (desde consulta hasta respuesta)
- Tasa de procesamiento (consultas RAG/minuto)
- Tamaño y latencia de la cola RabbitMQ bajo carga
- Uso de CPU/memoria en el servicio de procesamiento Python
- Tasa de errores o timeouts en procesamiento

5. Herramienta seleccionada y requerimientos de infraestructura

Herramienta seleccionada

- **Nombre**: Locust
- **Versión**: python 3.12, locust 2.33.1
- **Justificación de selección**:
    - Soporte para protocolos HTTP/HTTPS y diversos formatos de datos
    - Capacidad para simular múltiples usuarios concurrentes
    - Posibilidad de distribuir pruebas en múltiples nodos
    - Soporta pruebas completamente programables por medio de python
    - Amplia documentación y comunidad activa

Infraestructura requerida en Google Cloud Platform

- **Tipo de instancia recomendada**: e2-highcpu-8 (8 vCPU, 8 GB RAM)
    - **Alternativa para mayor carga:** e2-highcpu-16 (16 vCPU, 16 GB RAM) para pruebas con alto volumen de usuarios concurrentes
- **Almacenamiento**:
    - **Persistent Disk**: 50 GB SSD para sistema operativo, logs y resultados de pruebas.
    - **Cloud Storage**: Bucket para almacenar documentos de usuario y respaldos de datos de prueba.
- **Sistema operativo**: Debian 11 o Ubuntu Server 22.04 LTS
- **Red**: Ancho de banda suficiente (al menos 1 Gbps) para generar la carga necesaria
    - Configuración de red premium para reducir la latencia durante pruebas de carga.
- **Configuraciones adicionales**:
    - Reglas de firewall configuradas para permitir tráfico HTTP/HTTPS.
    - IAM role con permisos suficientes para acceder a los recursos necesarios.

    - IP externa estática asignada a la instancia de pruebas para acceso consistente.
    - Cloud NAT configurado si se requieren múltiples instancias para generar carga.
- **Herramientas de Monitoreo:**
    - **Cloud Monitoring** habilitado para seguimiento de métricas durante pruebas de carga.
    - **Cloud Logging** configurado para captura centralizada de logs de pruebas.

Justificación del dimensionamiento

- **CPU**: 16 vCPU permiten suficiente capacidad para generar hasta 500 usuarios virtuales concurrentes con herramientas como Locust.
- **Memoria**: 16 GB proporcionan la capacidad suficiente para:
    - Mantener múltiples hilos de ejecución de pruebas.
    - Gestionar el procesamiento de documentos de tamaño medio (~5 Mb por documento).
    - Mantener resultados en memoria durante la ejecución.
- **Disco**: 50 GB permiten almacenar logs detallados, resultados de múltiples ejecuciones, guardar copias de los documentos de prueba utilizados en los escenarios y espacio para herramientas de análisis y visualización de resultados.
- **Escalabilidad**: Para pruebas más intensivas (>500 usuarios concurrentes o documentos de gran tamaño):
    - Escalar verticalmente a instancias e2-highcpu-16 (16 vCPU, 16GB RAM)