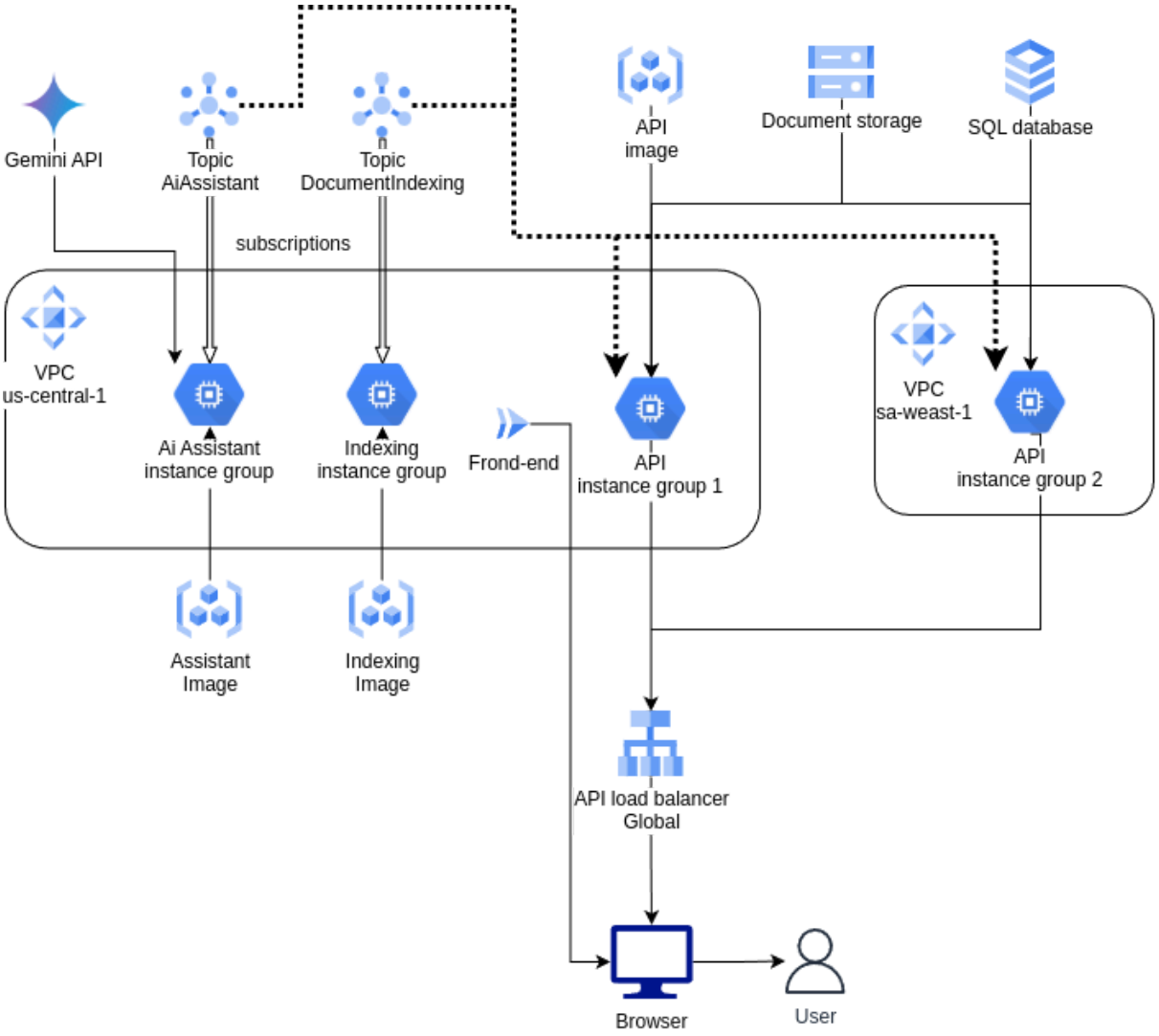
Documentación Entrega 3

Arquitectura de solución

A continuación se presenta un diagrama con la arquitectura de la solución implementada



Justificación de arquitectura

1. Uso de instance groups en dos regiones (us-central-1 y sa-west-1):

Esta decisión se basa en alta disponibilidad y reducción de latencia. Al tener grupos de instancias distribuidos en distintas regiones:

Se garantiza la tolerancia a fallos regionales: si una región (por ejemplo, us-central-1) sufre una interrupción, el sistema puede seguir operando desde sa-west-1.

Se mejora el rendimiento para usuarios geográficamente distribuidos, reduciendo la latencia al enviar solicitudes a la región más cercana.

Esto es esencial en sistemas críticos donde se necesita asegurar continuidad operativa y experiencia de usuario consistente.

2. Uso de Pub/Sub:

Se utiliza Pubsub para desacoplar los componentes del sistema y permitir una arquitectura basada en eventos:

Las instancias de AI Assistant y Indexing están suscritas a estos temas y procesan mensajes de manera asíncrona y escalable.

Esto mejora la escalabilidad, ya que cada componente puede procesar los mensajes a su ritmo y se facilita el manejo de cargas variables.

Video

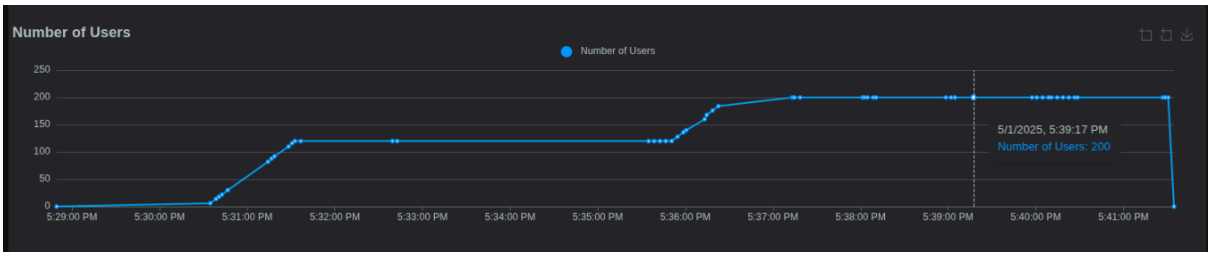
Para ver un video mostrando la configuración en GCP pueden ver el siguiente video

Análisis de capacidad

Escenario 1

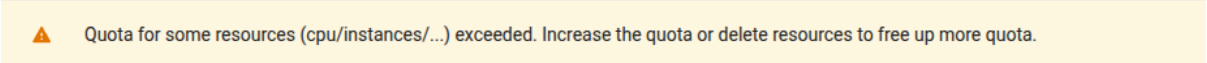
Descripción

Este escenario consiste en una prueba de indexación de documentos, en este los usuarios hacen requests entre 0.7 y 1.3 segundos. Los requests estuvieron distribuidos en un 70% indexado de nuevos documentos, y en 30% lectura de los documentos creados. En total la prueba duró aproximadamente 10 minutos en los cuales hubo un periodo con 120 usuarios y otro en el que se llegó a 200.



Resultados

En esta prueba el api pudo escalar exitosamente para suplir la demanda ejecutada por los usuarios, sin embargo se encontró un problema en cuanto al escalamiento del worker de indexado de documentos. Esto es debido a una limitación en el número de CPU disponibles para el proyecto. Sin embargo no se presentó ningún error en la prueba

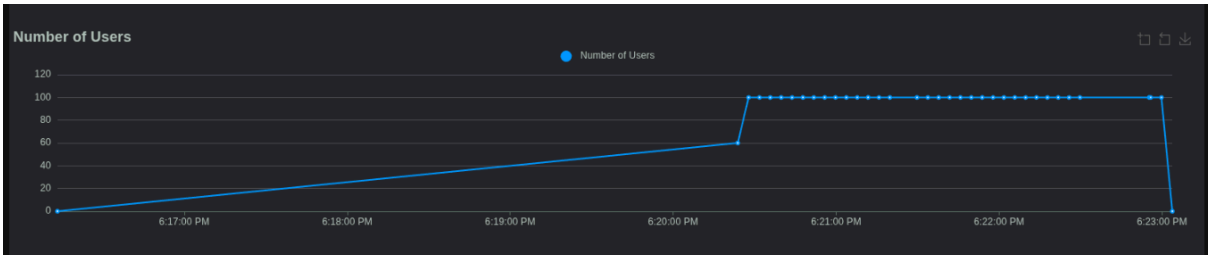


Para más información puede referirse a las imágenes y el archivo README.md en el siguiente link

Escenario 2

Descripción

Este es un caso de generación de respuestas por parte del asistente para esto se tienen usuarios que generan un request creando un mensaje en un chat cada 0.1 segundos. La creación del mensaje desencadena el proceso que finaliza en la respuesta del agente siendo introducida en el chat. En total la prueba duró alrededor de 7 minutos y se alcanzaron 100 usuarios concurrentes.



Resultados

En esta prueba parece que el verdadero factor limitante fue el *rate-limit* de el modelo Geminis. Se llega a esta conclusión ya que en ningún momento hubo acumulacion de *unacked-messages.* Aun así la utilización del cpu tanto del API como del worker que se encarga de la creación del contexto para la IA.

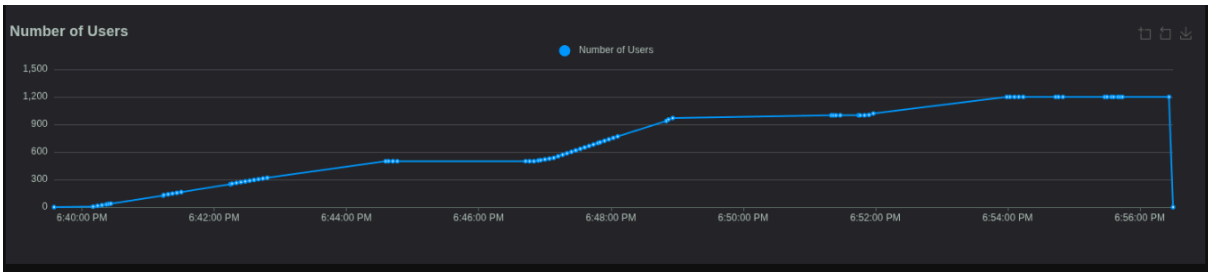
Para más información puede referirse a las imágenes y el archivo README.md en el siguiente link

Escenario 3

Descripción

En el escenario 3 hacemos una prueba de recuperación de información, Esta consta de usuarios que hacen un request entre 0.7 y 1.7 segundos. Los requests están divididos de

forma equitativa entre hacer un retrival de los chats del usuario y de sus documentos. La prueba duró alrededor de 16 minutos y se llegaron a 1200 usuarios concurrentes en tres escalones.



Resultados

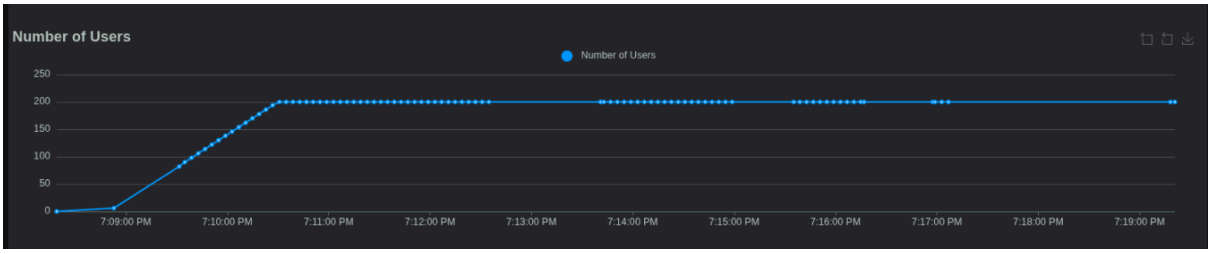
Para esta prueba pudimos observar unos pocos errores a causa de time-outs. Ya que pudimos observar que tanto el api como los otros servicios no pasaron por ningún escalamiento y que tanto los usos de cpu como de red se mantenían bajos (en el lado del servidor), concluimos que el factor limitante en esta prueba fue la red desde la cual se realizaron las pruebas; la cual se vio saturada.

Para más información puede referirse a las imágenes y el archivo README.md en el siguiente link

Escenario 4

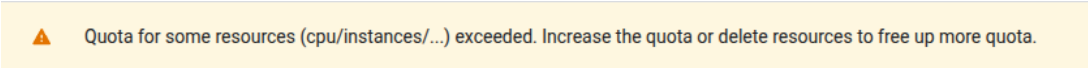
Descripción

Esta prueba corresponde con una prueba completa de las funcionalidades del sistema. Consta de usuarios que producen entre 0.7 y 1.3 requests por segundo. Los requests están divididos de la siguiente manera: 40% lectura de documentos, 40% lectura de chats, 10% crear mensajes (lo que arranca el proceso del asistente IA). En esta prueba se llegó a 200 usuarios concurrentes.



Resultados

En este caso también se observaron dos fallos del sistema. El primero fue la observación de errores por time-out de los cuales aun así solo se observaron 54. Y también ocurre el mismo problema de escalamiento donde el proceso de indexado no pudo escalar su número de CPUs porque el escalamiento del api ya cubría la cuota máxima



Para más información puede referirse a las imágenes y el archivo README.md en el siguiente link