Arquitectura Técnica para Sistema RAG Empresarial

1. Resumen Ejecutivo

Esta arquitectura propone una solución **Serverless-First y Event-Driven** diseñada en **Google Cloud Platform (GCP)**. El objetivo principal es minimizar la carga operativa (NoOps) mientras se maximiza la escalabilidad para manejar tanto el volumen inicial de 5,000 documentos como un crecimiento exponencial futuro. El sistema desacopla el procesamiento pesado (**Ingesta**) de la interacción en tiempo real (**Inferencia**), garantizando baja latencia y alta disponibilidad para un asistente de documentación técnica.

2. Objetivo y Contexto

Atributo	Descripción
Objetivo	Diseñar y diagramar una arquitectura RAG (Retrieval-Augmented Generation) para un sistema de preguntas y respuestas sobre documentación técnica.
Contexto	Una empresa necesita un asistente que responda preguntas de clientes basándose en su documentación oficial (manuales, guías, FAQs).
Enfoque Arquitectónico	Serverless-First, Event-Driven, NoOps enfocado en GCP.

3. Componentes de la Arquitectura

3.1. Pipeline de Ingesta (Asíncrona y Desacoplada)

El pipeline de ingesta está diseñado para evitar el procesamiento síncrono que bloquea al usuario y es capaz de manejar tareas de larga duración.

Componente	Tecnología GCP	Propósito y Justificación
Trigger	**Eventarc & GCS**	Implementa una arquitectura reactiva. Al subir un archivo (ej. PDF) a Google Cloud Storage

Componente	Tecnología GCP	Propósito y Justificación
		(GCS), Eventarc dispara el proceso sin necesidad de sondeo. Esto reduce costos de cómputo ocioso.
Orquestación	**Cloud Run Jobs**	Se prefiere Jobs sobre Services para la ingesta, ya que permite tareas de larga duración (hasta 24h) sin *timeouts* de HTTP, ideal para procesar PDFs extensos.
Parsing & Extracción	**Google Document AI**	A diferencia de librerías *open-source* (ej. PyPDF), Document AI utiliza modelos pre-entrenados para extraer estructura compleja de tablas y diagramas, información vital en documentación técnica.
Base de Datos Vectorial	**Vertex AI Vector Search**	Elegido por su capacidad de manejar millones de vectores con latencia sub-milisegundo y su integración nativa para escalado horizontal, asegurando que la arquitectura sea *future-proof*, superando la capacidad inicial de 5k documentos.

3.2. API de Inferencia (Servicio en Tiempo Real)

Esta capa se enfoca en la baja latencia y el escalado instantáneo para responder a las consultas de los usuarios.

Componente	Tecnología Clave	Propósito y Justificación
Compute	**Cloud Run + FastAPI**	Contenedores *stateless* que escalan a cero cuando no hay tráfico (ahorro de costos). FastAPI se selecciona por su soporte nativo de concurrencia (`async/await`), crucial para orquestar múltiples llamadas de red (DB, Cache, LLM) simultáneamente.

Componente	Tecnología Clave	Propósito y Justificación
Modelo de Lenguaje (LLM)	**Gemini 2.5 Pro**	Seleccionado por su amplia ventana de contexto, permitiendo inyectar múltiples fragmentos de manuales (recuperados por RAG) sin que el modelo pierda coherencia.

3.3. Capa de Optimización y Monitoreo

Esta capa mejora la eficiencia, reduce costos y permite la mejora continua del sistema.

Componente	Tecnología GCP	Estrategia de Optimización
Estrategia de Caching	**Cloud Memorystore (Redis)**	Implementación de **Semantic Caching**. Antes de llamar al LLM, verifica si una pregunta semánticamente similar ya fue respondida. Esto reduce costos operativos (llamadas al LLM) y mejora la latencia percibida.
Feedback Loop	**BigQuery**	Almacenamiento asíncrono de interacciones y feedback explícito de los usuarios. Esto permite monitorear el *Drift* en los temas de consulta y re-calibrar los *prompts* o la base de conocimiento basándose en datos reales.

4. Estrategia de Escalabilidad, Resiliencia y Seguridad

4.1. Escalabilidad y Resiliencia

Aspecto	Estrategia Implementada
Manejo de Alto Volumen	El uso de **Cloud Load Balancing** y **Cloud Run** permite manejar picos de tráfico repentinos sin intervención manual (Auto-scaling basado en concurrencia de CPU/Requests).

Aspecto	Estrategia Implementada
Crecimiento de Datos	**Vertex AI Vector Search** desacopla el almacenamiento del cómputo. Pasar de 5,000 a 500,000 documentos no degrada el tiempo de respuesta de la búsqueda vectorial.
Tolerancia a Fallos	El pipeline de ingesta es **idempotente**. Si un proceso falla, puede reintentarse (usando Dead Letter Queues en Eventarc) sin duplicar datos en la base vectorial, garantizando la integridad de la base de conocimiento.

4.2. Consideraciones de Seguridad

Principio	Implementación
Identidad y Acceso	Uso estricto de **Service Accounts** con el principio de **menor privilegio (IAM)** para la comunicación entre los servicios (Cloud Run, Storage, Vertex AI).
Comunicaciones	Todas las comunicaciones entre servicios de GCP se realizan a través de redes privadas y puntos finales seguros, minimizando la exposición pública.