Especificaciones Técnicas Detalladas del Sistema RAG Híbrido

Este documento detalla los componentes técnicos, los modelos de Inteligencia Artificial seleccionados y las configuraciones de vectorización necesarias para la implementación del sistema "Experto Inteligente" de InfoSubvenciones.

1. Stack Tecnológico Principal

Componente	Función	Tecnología Específica	Notas
Ingesta de Datos	Extracción, procesamiento y orquestación.	Python (con librerías requests, pypdf, pandas)	Script robusto que se conecta a la API de InfoSubvenciones.
Base de Datos Híbrida	Almacenamiento SQL y Vectorial.	**Supabase** (PostgreSQL) con extensión pgvector	Combina filtros SQL de metadatos (region, deadline) con la búsqueda semántica de vectores.
Orquestación LLM	Gestión de llamadas a la API de Google, *prompting* avanzado y lógica de RAG.	SDK de Google GenAI para Python o Node.js	Maneja la cadena de Ingesta Inteligente y la Consulta del Usuario.

2. Modelos de Lenguaje Grande (LLMs) - Gemini

Utilizamos una estrategia de "Modelos por Tarea" para optimizar la calidad, la velocidad y, crucialmente, el coste, delegando las tareas de gran volumen (lectura) a modelos más ligeros y las tareas de precisión (respuesta) a modelos de alto rendimiento.

2.1. Modelos para la Fase de Ingesta Inteligente (Smart Ingestion)

El objetivo es reducir el texto de un PDF de 400 KB (aprox. 270k tokens) a un Resumen de

Experto de 270 tokens.

Tarea	Modelo Seleccionado	Razón de la Elección
1. Extracción de Resumen de Experto	**Gemini 2.5 Flash**	Excelente para tareas de resumen de textos largos. Ofrece la velocidad de *Flash* con la capacidad de comprender y extraer información compleja de documentos extensos (hasta 1 millón de tokens de contexto), garantizando la calidad del resumen.
2. Conversión a JSON Estructurado	**Gemini 2.5 Flash**	Se utiliza para extraer metadatos estructurados (Ej: beneficiary_type, requirements) en formato JSON a partir del PDF, alimentando las columnas SQL de la base de datos de Supabase.

2.2. Modelos para la Fase de Consulta (RAG y Respuesta)

Tarea	Modelo Seleccionado	Razón de la Elección
3. Generación de Respuesta Final	**Gemini 2.5 Flash**	Combina velocidad con un alto rendimiento en razonamiento. Es ideal para la fase final del RAG, donde debe leer el fragmento de texto original recuperado (la "Verdad") e integrarlo en una respuesta coherente y, obligatoriamente, citada.

3. Especificaciones del Modelo de Embedding

(Vectorización)

La elección del modelo de *embedding* es vital, ya que define el nivel de precisión semántica de la búsqueda y la dimensión de los vectores que se almacenarán en la base de datos pgvector de Supabase.

Parámetro	Valor Recomendado	Impacto en el Proyecto
Modelo de Embedding	**text-embedding-004** (Google)	Modelo de Google de última generación, robusto y optimizado para la comprensión contextual.
Dimensión del Vector	**768 Dimensiones**	Dimensión estándar y eficiente. Afecta directamente al espacio de almacenamiento. Más dimensiones = más precisión, pero más coste de almacenamiento y RAM.
Contenido Vectorizado	**Resumen de Experto** (270 tokens)	**Clave de la optimización**. Evita vectorizar los 14.300 millones de tokens brutos, reduciendo el coste de ingesta de $1,430 USD a $\sim\$3.86 \text{ USD}$.

Configuración en Supabase (pgvector)

El vector de 768 dimensiones debe ser configurado en la tabla principal de Supabase (por ejemplo, convocatorias_embeddings) de la siguiente manera:

```
CREATE TABLE convocatorias_embeddings (
    id UUID PRIMARY KEY,
    numConv VARCHAR(50),
    region VARCHAR(100),
    deadline DATE,
    full_text_reference TEXT,
    summary_embedding VECTOR(768) -- Tipo de dato de pgvector con 768 dimensiones
);
```

4. Opciones Avanzadas de Modelos LLM

Si el presupuesto lo permite o si la complejidad de las bases reguladoras requiere un razonamiento más profundo, se pueden considerar las siguientes opciones:

Tarea	Opción Alternativa	Razón
Generación de Resumen y Respuesta	**Gemini 2.5 Pro**	Mayor capacidad de razonamiento en textos muy densos y largos. Puede mejorar ligeramente la calidad de las respuestas, pero con un coste por token significativamente mayor (especialmente en Output).
Búsqueda (Embedding)	**text-embedding-005**	La versión más reciente del modelo de embedding, potencialmente con mejor rendimiento en la búsqueda semántica. Se recomienda probar *text-embedding-004* primero por su excelente relación rendimiento-coste.

5. Proceso de Ingesta Inteligente Paso a Paso (El Script Python)

El *script* de ingesta debe ejecutar los siguientes pasos de forma secuencial y automatizada para cada uno de los 143.000 documentos:

1. **Descarga y Extracción:** Conexión a la API de InfoSubvenciones, descarga del PDF y extracción de todo el texto nativo (57.2 GB total).
2. **Deduplicación:** Cálculo de un *hash* (MD5) del texto completo. Si el *hash* ya existe en la base de datos (indicando que la Base Reguladora es idéntica), se salta el procesamiento.
3. **Llamada a Gemini (Extracción):** El texto completo se inyecta en **Gemini 2.5 Flash** con un *prompt* que pide la extracción de metadatos en formato JSON y la generación del **Resumen de Experto** (200 palabras).
4. **Vectorización:** El **Resumen de Experto** (38.61 millones de tokens totales) se envía al modelo de **Embedding** para obtener el vector de 768 dimensiones.

5. **Almacenamiento Híbrido:**
    - Los metadatos estructurados (JSON) y el link al PDF se guardan en las columnas SQL de Supabase.
    - El vector (summary_embedding) se guarda en la columna VECTOR(768).