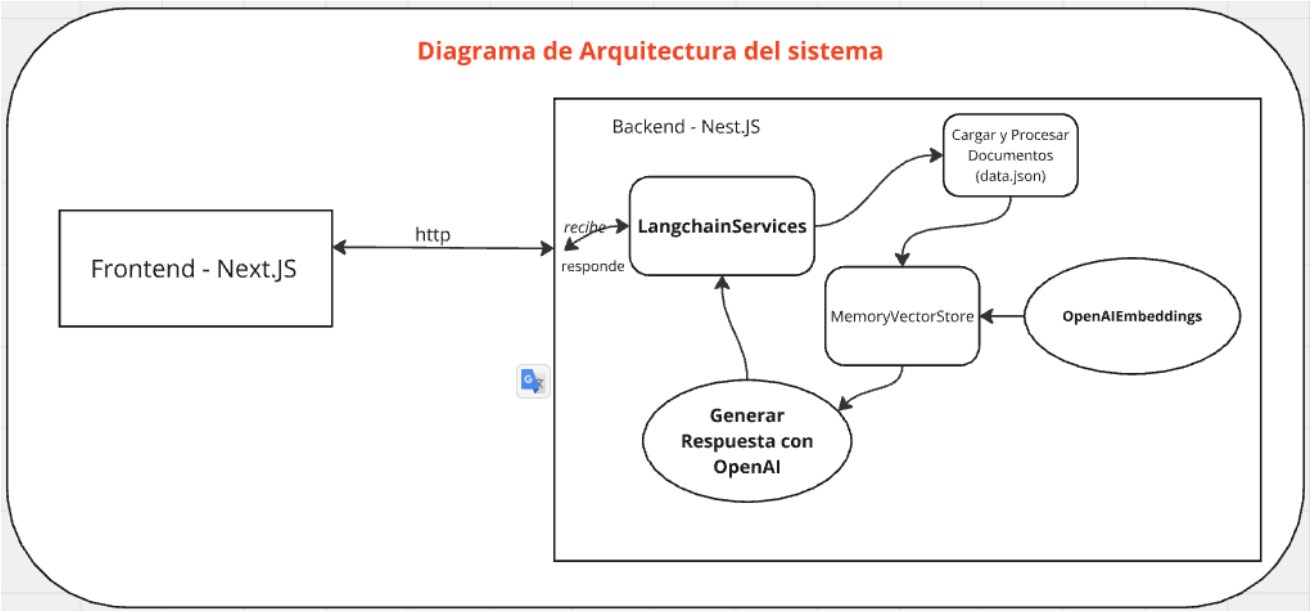
1. Diagrama de Arquitectura del sistema.



2. Listado y selección del stack con su justificación.

Frontend: Next.js 14.1.0 (Proyecto Bootstrapping)

- **Justificación:**
  - **Bootstrapping con Proyecto Open-Source:** Se seleccionó un proyecto de frontend open-source que se adaptaba al formato convencional de chatbots. Esto permitió acelerar el desarrollo al reutilizar componentes y estructuras ya probadas, garantizando una interfaz de usuario coherente y funcional.
  - **Rendimiento y SEO:** Next.js ofrece renderizado del lado del servidor (SSR) y optimizaciones automáticas que mejoran el rendimiento y la optimización para motores de búsqueda.

Backend: NestJS

- **Justificación:**
  - **Comodidad y Experiencia:** La elección de NestJS se basó principalmente en la comodidad y la experiencia previa con el framework, lo que permitió un desarrollo más rápido y eficiente.
  - **Gestión de CLI:** NestJS ofrece una CLI poderosa que facilita la creación y gestión de recursos y servicios diversos, permitiendo probar métodos y funcionalidades de manera sencilla.
  - **Arquitectura Modular:** Facilita la escalabilidad y el mantenimiento del código, permitiendo una estructura organizada y modular.

Procesamiento de Lenguaje Natural: LangChain

- **Justificación:**
  - **Rendimiento Superior:** elegi LangChain sobre alternativas como LlamaIndex debido a su mejor rendimiento y facil adaptacion a sus multiples herramientas funcionales con OpenAI, como en este caso el procesamiento

de archivos Json . LlamaIndex demostró ser lento y menos desarrollado, lo que impactaba negativamente en la eficiencia del sistema y en el tiempo de desarrollo.
    - **Compatibilidad con OpenAI:** LangChain ofrece una integración más fluida y robusta con las APIs de OpenAI, facilitando la generación de respuestas de alta calidad.
    - **Facilidad de Implementación:** Aunque OllamaAI es una opción gratuita, su integración mediante contenedores Docker resultó en una experiencia más lenta y menos eficiente en comparación con el uso directo de OpenAI.

OpenAI API

- **Justificación:**
    - **Avanzada Capacidad de Modelos de Lenguaje:** OpenAI proporciona modelos de lenguaje altamente avanzados y optimizados, ofreciendo respuestas más precisas y coherentes.
    - **Integración Directa con LangChain:** OpenAI se integra de manera más eficiente con LangChain, facilitando la vectorización y procesamiento de archivos JSON.
    - **Desempeño y Fiabilidad:** OpenAI ofrece un rendimiento más consistente y confiable en comparación con alternativas como OllamaAI y Gemini, que presentaron limitaciones en la integración y eficiencia.

Almacenamiento de Datos: JSON y MemoryVectorStore

- **Justificación:**
    - **Simplicidad y Eficiencia:** Utilizar archivos JSON para almacenar datos estructurados es sencillo y eficiente para volúmenes de datos manejables. En este caso elegí JSON porque eran datos de prueba pero también pude haber utilizado MongoDB Atlas.
    - **Rendimiento en Memoria:** `MemoryVectorStore` permite una rápida búsqueda y recuperación de embeddings, adecuado para el volumen actual de datos y usuarios.
    - **Facilidad de Implementación:** Almacenamiento en memoria facilita la gestión y acceso a los datos sin la complejidad adicional de configurar una base de datos persistente.

Despliegue: Docker y Google Cloud Run (GCP)

- **Justificación:**
    - **Portabilidad:** Docker asegura que la aplicación funcione de manera consistente en diferentes entornos, eliminando problemas de compatibilidad.
    - **Escalabilidad Automática:** Google Cloud Run permite escalar automáticamente las instancias de la aplicación según la demanda, manejando eficientemente el tráfico fluctuante.

4. Preguntas adicionales.

Dimensionamiento de la Infraestructura en la Nube

Servidor Backend

- **Especificaciones**:
  - **CPU**: 2 vCPUs
  - **RAM**: 4 GB
- **Requerimientos de Carga**:
  - Estas especificaciones están diseñadas para manejar múltiples interacciones concurrentes, permitiendo que el servidor procese eficientemente las peticiones de hasta 500 usuarios al mismo tiempo.
  - Implementar un **balanceador de carga de GCP** para distribuir el tráfico de manera equitativa entre varias instancias del servidor backend. Esto asegura alta disponibilidad, mejora el rendimiento y permite escalar fácilmente según la demanda de usuarios.

Costos en la Nube

- **Google Cloud Platform (GCP)**:
  - Un servidor con las especificaciones mencionadas podría costar entre **$30 a $50 USD** al mes, dependiendo de la región y el tipo de instancia seleccionada.

Base de Datos (MongoDB Atlas)

- **Especificaciones**:
  - **Cluster M10**, que es adecuado para aplicaciones en desarrollo y producción ligera.
- **Costo**:
  - Aproximadamente **$57 USD** al mes.

Costo de OpenAI

Para calcular el costo mensual de utilizar el modelo GPT-3.5 con 500 usuarios concurrentes, haremos las siguientes suposiciones:

- **Tokens por consulta**: Supongamos que cada usuario realiza aproximadamente 5 consultas al día.
- **Tokens por consulta**: Asumamos que cada consulta utiliza un promedio de **100 tokens para la entrada** y **200 tokens para la salida** (un total de **300 tokens por consulta**).

Cálculo

1. **Tokens diarios por usuario**:
   - 5 consultas/dia×300 tokens/consulta=1,500
2. **Tokens diarios para 500 usuarios**:
   - 1,500 tokens/dia×500 usuarios=750,000 tokens/dia

3. **Tokens mensuales para 500 usuarios**:
    - 750,000 tokens/dia×30 dias=22,500,000 tokens/mes

Costo mensual de OpenAI

- **Costo por 1 millón de tokens**:
    - Entrada: **€0.47**
    - Salida: **€1.40**
- **Costo total por 1 millón de tokens (entrada + salida)**:
    - 0.47+1.40**= €1.87**
    - (22,500,000 tokens/1,000,000)x**€1.87= € 42.075**

Costo Total Estimado

Sumando todos los costos:

- **Servidor Backend**: $30 a $50 USD
- **MongoDB Atlas**: $57 USD
- **OpenAI**: **€42.08** (aproximadamente **$45 USD** al tipo de cambio actual).

Total estimado: **$132 a $152 USD al mes** (con el costo de OpenAI convertido a USD).

Costo Salarial

Esto dependerá de la cantidad de desarrolladores y el tiempo de desarrollo.

Es necesario aclarar que los costos anteriores si bien están basados en supuestos, en los tres casos los costos fueron calculados por la calculadora de pricing de cada empresa.