Documentación Prueba Técnica Vambe

Nombre: Max Valenzuela Garafulic

Página: https://vambe-front-end.vercel.app/

Arquitectura del sistema

La arquitectura que se siguió a lo largo del desarrollo de la aplicación fue una arquitectura de microservicios. Este enfoque se puede visualizar en primer lugar en la separación de responsabilidades en Backend y Frontend. Además se utilizó una API externa para manejar las consultas. Por último, se optó por optimizar el rendimiento, a pesar de no ser una aplicación que requiera de mucho, mediante el cuidado en algunos endpoints delicados y su no presentación en el frontend.

Decisiones claves

1. Desarrollar la prueba con dos repositorios, uno para backend y otro para frontend. Esto no fue especificado en el enunciado del desafío, pero se hizo de esta manera para tener un enfoque más detallado de las funciones de cada uno, y para poder diferenciar los trabajos realizados.
2. Utilizar la API de Google Gemini. Está decisión se basó en que, a mi entender, es la que permite más consultas de manera gratuita a diferencia de OpenAI, que se intentó integrar pero no permitía realizar las consultas, dado que había que realizar pagos.
3. Las tecnologías utilizadas fueron Node.js para el backend, React con Vite en el frontend, y SQL para la base de datos. Esto debido a que son las tecnologías que siento que mejor se adecuaban a lo solicitado por el desafío, además de presentar un manejo previo sobre ellas.
4. Las categorías y subcategorías correspondientes se decidieron a través del análisis de las transcripciones de las reuniones con los clientes, tomando en cuenta los patrones de información que se repetía en cada una, y la información que sería relevante manejar para ayudar a los altos directivos de Vambe en la toma de decisiones respecto las reuniones con clientes. Es por esto que se llegó a las categorías: Industria (del cliente), Interés en Funcionalidades, Motivación para Buscar Vambe y Canal de Descubrimiento de Vambe.
5. Se creó un Dashboard (Panel) que lleva a cuatro formas más interactivas de ver diferentes métricas obtenidas de las transcripciones, reuniones y categorías asignadas. Además, se hizo una página más simple que muestra las cuatro categorías, y que cada una lleva a un gráfico básico de como se distribuyen sus respectivas subcategorías.

6. Se hizo el supuesto de que el archivo CSV no se debía subir desde la página. Cabe recalcar que la subida del archivo al sistema está hecho mediante un endpoint tipo POST a través de una aplicación como Postman.
7. Se realizó el supuesto que las asociaciones entre categorías y subcategorías con las transcripciones tampoco se realizaran a través de la página. Esto para no sobrecargar el LLM utilizado. Al igual que el caso anterior, para crear las asociaciones se realizó a través de un endpoint tipo GET existente mediante la aplicación Postman.
8. Se realizó el supuesto que la página se utilizará a través de computadores, por lo que la responsividad de esta no está 100% desarrollada.

Consideración importante

Debido a que los deploy de tanto backend como frontend están hechos mediante las versiones gratuitas de Render y Vercel respectivamente, estos pueden tomar más tiempo del esperado. **Es por esto por lo que al ver el deploy del frontend por primera vez probablemente demorará (hasta 10 minutos) en mostrar la información.**