🚀 Reto Técnico: Desarrollo de Plataforma de Mensajería Web 💬🔧

🎯 Objetivo

El objetivo de este reto es diseñar e implementar una plataforma de mensajería en tiempo real **similar a WhatsApp o Messenger**, utilizando **React** para el frontend y **Supabase** como base de datos.

Se evaluará la **infraestructura**, **escalabilidad**, **buenas prácticas** y **optimización del rendimiento** del proyecto.

Pueden usar herramientas de IA para acelerar el desarrollo, recomendados: Copilot, chatgpt, v0. https://v0.dev/chat

---

📌 Requisitos

🖥️ 1. Frontend (React)

✅ Debe ser una **SPA (Single Page Application)**.
✅ Soporte **responsivo** 📱💻 para múltiples dispositivos (desktop, tablet y móvil).
✅ **Autenticación de usuarios** (correo y contraseña) utilizando **Supabase Auth**.
✅ **Mensajería en tiempo real** 🔄 usando **Supabase Realtime**, pero **únicamente para notificaciones** o nuevos mensajes.
✅ **Los datos deben provenir de APIs**, asegurando **flexibilidad** y una **estructura de datos unificada**.
✅ UI **moderna y atractiva** 🎨, similar a las plataformas de mensajería más utilizadas.
✅ **Gestión de estado eficiente** (Context API / Zustand en React).
✅ **Soporte para carga y visualización de imágenes** 🖼️ en los mensajes.
✅ **Seguridad** 🔒: Usar únicamente la key pública de Supabase y configurar correctamente las políticas de seguridad.

🔙 2. Backend (NestJS)

🔹 API para:

- 📜 **Listar contactos**.
- 💬 **Listar mensajes por contacto con paginación**.
- ✉️ **Enviar mensajes**.

- 🔎 **Buscar contactos**.
  - 🔹 **Conexión con Supabase** para gestionar la base de datos.
  - 🔹 **El frontend solo se conectará directamente con Supabase** para eventos en tiempo real, el resto de la información vendrá del backend.
  - 🔹 **Despliegue en Render** 🌍 usando un nuevo correo de prueba.
  - 🔹 **Configurar un cronjob cada 5 minutos en Render** para evitar que el servicio se suspenda.

🏗️ 3. Infraestructura y Escalabilidad

- 🔸 **Arquitectura modular y limpia en NestJS** 🏛️.
- 🔸 **Buenas prácticas en la organización del código** 📂.
- 🔸 **Manejo eficiente de eventos en tiempo real** ⚡ para evitar sobrecarga de peticiones.
- 🔸 **Optimización de recursos y manejo de errores** 🚀.
- 🔸 **Capacidad de manejar múltiples APIs externas** manteniendo una estructura de datos uniforme.

📝 Entrega y Evaluación

🛠️ 1. Código Fuente

📌 **Repositorio en GitHub** con instrucciones claras de instalación y ejecución (Nos pueden compartir acceso o compartir la explicación del codigo mediante un vídeo, generalmente estructura y métodos importantes). **No es obligatorio compartir código ni será tomado en cuenta como criterio de evaluación**

🌍 2. Infraestructura

✔ **Backend desplegado en Render** y accesible desde internet.
✔ **Uso de Supabase** para la base de datos y autenticación.
✔ **Documentación detallada** 📄 sobre la arquitectura y decisiones técnicas tomadas.

🎥 3. Demostración

🎬 **Video corto** mostrando el funcionamiento de la aplicación.

🎁 Bonus (Opcional, pero valorado) 🎯

💥 **Notificaciones push** 🔔 para nuevos mensajes.
💥 **Mensajes multimedia** 📷🎥📄 (imágenes, videos, documentos).
💥 **Uso de Docker** 🐳 para el despliegue en Render.

⏳ Plazo y Entrega

🕐 **Tiempo estimado:** 1 semana.
📩 **Entrega:** Enviar el video de la aplicación funcionando.

🎯 Criterios de Evaluación

Criterio	Peso 📊
✅ **Funcionalidad** (autenticación, contactos, mensajes)	**35%**
🎨 **Escalabilidad / UI / Responsividad**	**40%**
🔒 **Seguridad** (validación de autenticación, políticas en Supabase)	**15%**
🌍 **Despliegue del backend en Render**	**10%**

🔹 **TOTAL:** 100%

🔥 ¡Mucha suerte! 🚀💡

En caso de empate, se evaluará por tiempo de entrega y durante la entrevista.