Prueba Técnica para Desarrollador Full Stack Senior

Objetivo

Diseñar e implementar una solución **escalable y modular** que consuma diferentes protocolos de comunicación (**REST, WebSocket y gRPC**). La prueba evaluará la capacidad del candidato para integrar tecnologías modernas, estructurar el sistema con buenas prácticas de arquitectura, y desplegar la solución en un entorno de producción.

Descripción del Sistema

Un sistema de gestión de **proyectos colaborativos** donde los usuarios pueden:

1. Crear y gestionar proyectos.
2. Añadir tareas a los proyectos.
3. Recibir **notificaciones en tiempo real** cuando se actualizan proyectos o tareas (via WebSocket).
4. Consultar estadísticas generales del sistema (por ejemplo, número de proyectos y tareas, etc.) a través de un servicio **gRPC**.

Instrucciones

Requisitos Técnicos

Backend (Microservicios NodeJS)

1. **Diseño Basado en Microservicios**:
    - Crear servicios separados para manejar:
        - **Gestión de proyectos y tareas**: REST API.
        - **Notificaciones en tiempo real**: WebSocket (usando `Socket.IO`).
        - **Estadísticas del sistema**: gRPC.
    - Cada servicio debe ser independiente y tener su propia lógica.
2. **Servicios requeridos**:
    - **REST API**:
        - `POST /projects`: Crear un proyecto.
        - `POST /projects/:projectId/tasks`: Añadir una tarea a un proyecto.
        - `GET /projects`: Listar todos los proyectos con sus tareas asociadas.
    - **WebSocket**:
        - Enviar notificaciones en tiempo real a los usuarios cuando:
            - Se crea o actualiza un proyecto.
            - Se crea o actualiza una tarea.

    - **gRPC**:
        - Implementar un endpoint para consultar estadísticas generales:
            - Número total de proyectos.
            - Número total de tareas.
3. **Base de Datos**:
    - Usar una base de datos **MongoDB** y Mongoose como ODM.
    - Diseñar una estructura clara y eficiente para las entidades **Proyectos** y **Tareas**.
4. **Autenticación y Autorización**:
    - Usar JWT para proteger los endpoints de la API REST y asegurar que solo usuarios autenticados puedan realizar acciones.
5. **Pruebas Unitarias**:
    - Agregar pruebas unitarias y de integración para los endpoints principales.

Frontend (React o Angular)

1. Crear una interfaz de usuario que permita:
    - **Listar proyectos y tareas** desde la API REST.
    - Crear nuevos proyectos y tareas mediante formularios.
    - Mostrar notificaciones en tiempo real (consumiendo WebSocket).
    - Un dashboard con estadísticas generales (consultadas a través de gRPC).
2. **WebSocket**:
    - Conectar a WebSocket para recibir actualizaciones en tiempo real y mostrarlas en un sistema de notificaciones amigable.
3. **Gestión del Estado**:
    - Usar **Context API** (React) para manejar el estado global.
4. **Estilos**:
    - Usar librería de UI como **Chakra UI** (React).

Producción y Despliegue

1. **Dockerización**:
    - Configurar todos los servicios (frontend, backend y base de datos) en contenedores Docker.
    - Proveer un archivo `docker-compose.yml` para facilitar la ejecución del sistema completo.
2. **Despliegue en Producción**:
    - Publicar la solución en un dominio público usando **Vercel**
    - Backend y WebSocket deben estar accesibles desde el mismo dominio o dominios relacionados.

Criterios de Desarrollo

1. **Complejidad Técnica**:
    - Implementación adecuada de REST, WebSocket y gRPC.
    - Diseño basado en microservicios.
2. **Calidad del Código**:
    - Buenas prácticas (nombres claros, separación de responsabilidades, etc.).
    - Código bien documentado.
3. **Experiencia de Usuario**:
    - Interfaz intuitiva y responsiva.
4. **Escalabilidad y Mantenimiento**:
    - Capacidad de la arquitectura para soportar nuevas funcionalidades.
5. **Despliegue**:
    - Uso adecuado de Docker y servicios en la nube.