Prueba Técnica para Desarrollador Full Stack

Descripción:

El objetivo de esta prueba es construir una pequeña aplicación de gestión de tareas (To-Do List) que permita a los usuarios agregar, actualizar, eliminar y ver tareas. La aplicación debe estar desarrollada en **Next.js** para el frontend y utilizar una API REST para interactuar con una base de datos **PostgreSQL** en el backend.

Requisitos Técnicos:

- **Frontend:** Construido en **Next.js** con **React.js**.
- **Backend:** API REST que maneje las operaciones CRUD (Crear, Leer, Actualizar, Eliminar).
- **Base de Datos:** Utilizar **PostgreSQL** para almacenar los datos de las tareas.
- **Control de versiones:** Usar **Git** para el control de versiones y entregar el proyecto en un repositorio.

Instrucciones del Proyecto

1. **Configuración de la Base de Datos**
    - Crear una base de datos en PostgreSQL con una tabla tasks que contenga los siguientes campos:
        - id: Identificador único de la tarea (UUID o serial).
        - title: Título de la tarea (cadena de texto).
        - description: Descripción de la tarea (cadena de texto).
        - status: Estado de la tarea (pendiente, en progreso, completada).
        - created_at: Fecha de creación de la tarea (timestamp).
        - updated_at: Fecha de última actualización de la tarea (timestamp).
2. **Desarrollo del Backend (API REST)**
    - Crear una API REST con las siguientes rutas:
        - GET /api/tasks: Devuelve una lista de todas las tareas.
        - GET /api/tasks/:id: Devuelve una tarea específica.
        - POST /api/tasks: Crea una nueva tarea.
        - PUT /api/tasks/:id: Actualiza una tarea existente.
        - DELETE /api/tasks/:id: Elimina una tarea.

- Asegúrate de manejar correctamente los códigos de estado HTTP y los mensajes de error.

3. **Desarrollo del Frontend (Next.js)**

   - Crear una interfaz simple que permita al usuario:
     - Ver todas las tareas en una lista.
     - Crear una nueva tarea mediante un formulario.
     - Editar una tarea existente.
     - Cambiar el estado de la tarea (de pendiente a en progreso o completada).
     - Eliminar una tarea.
   - Utilizar **React Hooks** para el manejo del estado y **Axios** o **Fetch API** para realizar las solicitudes a la API.

4. **Control de Versiones (Git)**

   - Utilizar Git para el control de versiones. Crea commits significativos a lo largo del desarrollo, documentando los cambios principales.
   - Subir el proyecto a un repositorio en GitHub, GitLab o Bitbucket y compartir el enlace del repositorio.

5. **Despliegue**

   - Subir el proyecto a un servidor gratuito (como **Vercel**, **Heroku**, **Render**, o similar) y proporcionar el enlace al proyecto en funcionamiento como parte de la entrega.

Criterios de Evaluación

1. **Funcionalidad**: La aplicación cumple con los requisitos de CRUD en el backend y las funcionalidades de la interfaz en el frontend.
2. **Calidad del Código**: Código limpio, bien estructurado y con convenciones de nombres claras.
3. **Control de versiones**: Uso adecuado de Git, con commits descriptivos.
4. **Eficiencia y Escalabilidad**: Buen manejo de las operaciones en la base de datos y la API, optimizando el rendimiento y la escalabilidad.
5. **Buenas Prácticas de Desarrollo**: Manejo adecuado de errores, códigos de estado HTTP, y manejo de dependencias.
6. **Despliegue**: Proyecto desplegado en un servidor gratuito y accesible, con un enlace funcional para evaluar la aplicación en línea.

Entrega

Comparte el enlace al repositorio y el enlace de la aplicación en funcionamiento en el servidor, junto con cualquier instrucción adicional para instalar y ejecutar el proyecto en un entorno local si fuera necesario.