Prueba Técnica de JavaScript

A continuación, se tiene la prueba técnica, la cual debe de hacer un fork al repositorio, y tendra 2 dias para completarlo.

Lea y complete lo solicitado, puede usar cualquier recurso que le pueda facilitar completar la prueba tecnica.

Para el caso de la parte teorica, es mejor que investigue en dado caso no sepa del tema y de lo que entienda pueda dar respuesta
a las preguntas.

Parte Teórica

APIs

1. **¿Qué es una API RESTful y cuáles son sus principales características?**
2. **Explica la diferencia entre los métodos HTTP GET y POST.**
3. **¿Qué es CORS y por qué es importante en el desarrollo de APIs?**

Express

1. **¿Qué es Express.js y cuáles son sus ventajas en el desarrollo de aplicaciones web?**
2. **Explica cómo manejar errores en una aplicación Express usando middleware.**

Objetos en JavaScript

1. **¿Cuál es la diferencia entre `Object.create()` y el operador `new` en JavaScript?**
2. **Explica el concepto de prototipos en JavaScript y cómo se relacionan con la herencia.**

Frontend (React y Next.js)

1. **¿Qué son los hooks en React y menciona al menos tres hooks comúnmente utilizados?**
2. **¿Qué es Next.js y cuáles son sus principales características?**

Sequelize

1. **¿Qué es Sequelize y para qué se utiliza en Node.js?**
2. **Explica cómo definir un modelo en Sequelize y cómo realizar una consulta básica.**

Parte Práctica

Backend (Node.js, Express, APIs)

1. Crea una API RESTful simple utilizando Express que realice operaciones CRUD (Crear, Leer, Actualizar, Eliminar) sobre una colección de "usuarios". La API debe tener las siguientes rutas:

    - **GET /users** (obtener todos los usuarios)
    - **GET /users/:id** (obtener un usuario específico)
    - **POST /users** (crear un nuevo usuario)
    - **PUT /users/:id** (actualizar un usuario existente)
    - **DELETE /users/:id** (eliminar un usuario)

    Implementa manejo de errores y validación básica de datos.

Frontend (React, Next.js)

2. Desarrolla una aplicación simple de lista de tareas (todo list) utilizando React o Next.js. La aplicación debe permitir:

    - Añadir nuevas tareas
    - Marcar tareas como completadas
    - Eliminar tareas
    - Filtrar tareas por estado (completadas/pendientes)

    Utiliza hooks de React para manejar el estado y, si usas Next.js, implementa al menos una ruta dinámica.

Integración Backend-Frontend

3. Conecta la aplicación frontend con la API backend que creaste en el ejercicio 1. Implementa las operaciones CRUD para los usuarios desde la interfaz de usuario.

Created by: *Carlos Soto*.