**Prueba Técnica: CRUD Completo en React.js y Node.js**

**Descripción del Proyecto**

El objetivo de esta prueba técnica es evaluar la capacidad del candidato para desarrollar una aplicación CRUD (Crear, Leer, Actualizar, Eliminar) utilizando React.js para el frontend y Node.js para el backend. La aplicación gestionará una lista de productos y categorías, donde cada producto estará asociado a una categoría.

**Requisitos**

1. **Backend (Node.js)**:
   * Crear un servidor en Node.js que ofrezca una API RESTful para gestionar productos y categorías.
   * Implementar relaciones entre productos y categorías.
2. **Frontend (React.js)**:
   * Crear una interfaz en React.js que permita interactuar con la API para realizar las operaciones CRUD.
   * El usuario debe poder asociar productos a categorías desde la interfaz.
3. **Base de Datos (MySQL)**:
   * Diseñar una base de datos relacional con las tablas necesarias para productos y categorías.
4. **Validaciones**:
   * Los formularios deben incluir validaciones para asegurar la calidad de los datos. Puedes implementar las validaciones como consideres más conveniente.
5. **Uso de Git**:
   * Subir el código a un repositorio público en GitHub.
6. **Investigación**:
   * Se permite y fomenta el uso de documentación y recursos externos.

**Especificaciones Técnicas**

**Backend (Node.js)**

1. **Estructura del Proyecto**:
   * Organizar el proyecto en carpetas como controllers, routes y config.
2. **Dependencias**:
   * Express.js para el servidor web.
   * mysql2 para conectarse a la base de datos MySQL.
   * dotenv para la gestión de variables de entorno.
3. **Modelos de Datos**:
   * **Productos**:
     + id (INT, PRIMARY KEY, AUTO\_INCREMENT)
     + name (VARCHAR, NOT NULL)
     + description (TEXT, NOT NULL)
     + price (FLOAT, NOT NULL)
     + category\_id (INT, FOREIGN KEY hacia la tabla de categorías)
   * **Categorías**:
     + id (INT, PRIMARY KEY, AUTO\_INCREMENT)
     + name (VARCHAR, NOT NULL)
4. **Relaciones**:
   * Cada producto debe estar asociado a una categoría.
5. **Endpoints de la API**:
   * Productos:
     + GET /products: Obtener todos los productos.
     + GET /products/:id: Obtener un producto por su ID.
     + POST /products: Crear un nuevo producto (asociado a una categoría).
     + PUT /products/:id: Actualizar un producto existente.
     + DELETE /products/:id: Eliminar un producto.
   * Categorías:
     + GET /categories: Obtener todas las categorías.
     + GET /categories/:id: Obtener una categoría por su ID.
     + POST /categories: Crear una nueva categoría.
     + PUT /categories/:id: Actualizar una categoría existente.
     + DELETE /categories/:id: Eliminar una categoría.

**Frontend (React.js)**

1. **Estructura del Proyecto**:
   * Puedes usar **Create React App** o **Vite** para inicializar el proyecto.
   * Organizar el proyecto en carpetas como components y services.
2. **Dependencias**:
   * Axios para las llamadas a la API.
   * React Router para la navegación.
3. **Componentes**:
   * **CategoryList**: Mostrar la lista de categorías.
   * **CategoryForm**: Formulario para crear o editar categorías.
   * **ProductList**: Mostrar la lista de productos, indicando a qué categoría pertenece cada uno.
   * **ProductForm**: Formulario para crear o editar productos, con la opción de seleccionar una categoría.
   * **ProductDetail**: Vista detallada de un producto específico, mostrando su categoría asociada.
4. **Validaciones**:
   * Asegurar que los formularios validen correctamente los datos ingresados (e.g., campos obligatorios, validación de precios, etc.).
   * Libertad para implementar las validaciones como consideres más conveniente (JavaScript puro, bibliotecas adicionales, etc.).

**Instrucciones para la Entrega**

1. **Repositorio GitHub**:
   * Subir el código a un repositorio público en GitHub.
   * Incluir un archivo README.md con:
     + Descripción del proyecto.
     + Instrucciones para instalar dependencias y configurar variables de entorno.
     + Pasos para ejecutar el backend y el frontend.
2. **Configuración y Ejecución**:
   * Proveer instrucciones claras para configurar y ejecutar tanto el servidor como la interfaz.
3. **Plazo**:
   * El proyecto debe completarse en un plazo máximo de 2 HR.

**Criterios de Evaluación**

1. **Funcionalidad**:
   * La aplicación debe cumplir con todos los requisitos funcionales (CRUD para productos y categorías, relación entre tablas, etc.).
2. **Calidad del Código**:
   * El código debe ser modular, limpio y bien organizado.
3. **Validaciones**:
   * Los formularios deben incluir validaciones adecuadas para garantizar la calidad de los datos.
4. **Documentación**:
   * La documentación del proyecto debe ser clara y detallada.

**Notas**

* Tienes total libertad para investigar y utilizar recursos externos para completar esta prueba.
* Se valorará la creatividad y la calidad en la implementación de las relaciones entre productos y categorías.

¡Buena suerte!