Prueba técnica front

**Ejercicio Práctico: Solicitar Tarjeta de Crédito y Consultar Saldo**

1. **Requerimientos:**
   * Utilizar React con TypeScript para el desarrollo.
   * Consumir una API bancaria abierta para simular la solicitud de tarjeta de crédito y la consulta de saldo. Puedes usar una API como la de Mocky (<https://www.mocky.io/>) para simular estas funcionalidades.
   * Mostrar un formulario donde el usuario pueda ingresar sus datos para solicitar la tarjeta de crédito, incluyendo nombre, apellido, número de documento, ingresos mensuales, etc.
   * Al enviar el formulario, mostrar un mensaje de éxito o error según la respuesta de la API de solicitud de tarjeta de crédito.
   * Una vez que se ha solicitado la tarjeta de crédito con éxito, permitir al usuario consultar el saldo de la tarjeta ingresando el número de tarjeta.
   * Al consultar el saldo, mostrar el saldo actual de la tarjeta.
   * Estilizar la aplicación de manera adecuada utilizando CSS o alguna librería de componentes como Bootstrap o Material-UI
2. **Tener en cuenta:**
   * Utiliza Create React App para iniciar el proyecto, pero idealmente usa Vite.
   * Utiliza axios o fetch para realizar las peticiones HTTP.
   * Presta atención al diseño y la usabilidad de la aplicación.
   * Organiza tu código de manera modular y sigue las mejores prácticas de desarrollo.
   * Agrega comentarios en el código cuando sea necesario para explicar tu lógica.
   * No olvides incluir un README.md con instrucciones para ejecutar tu aplicación.
   * Gestiona tu código en un repositorio de código (github)

**2. Formulario para el Evaluador:**

* + **Uso efectivo de TypeScript (15%):** ¿El candidato utilizó TypeScript de manera efectiva para gestionar los tipos de datos en la aplicación?
  + **Implementación del proceso de solicitud y consulta de tarjeta (20%):** ¿La aplicación muestra correctamente el proceso de solicitud de tarjeta de crédito y la consulta de saldo?
  + **Manejo adecuado de peticiones HTTP y casos de error (15%):** ¿El candidato manejó adecuadamente las peticiones HTTP y los casos de error?
  + **Diseño e intuitividad de la interfaz de usuario (20%):** ¿La interfaz de usuario es intuitiva y está bien diseñada?
  + **Calidad del código y estructura (15%):** ¿El código está bien estructurado y legible? ¿Se utilizaron componentes de manera eficiente?
  + **Técnica de manejo de estado (10%):** ¿El candidato utilizó alguna técnica de manejo de estado (por ejemplo, Context API, Redux) de manera adecuada?
  + **Aspectos adicionales y sugerencias de mejora (5%):** ¿Hay algún aspecto adicional que consideres destacable o alguna sugerencia de mejora?