Juego de Roles: Desarrollo Fullstack con React, Fecth, Axios y Ngrok

Julian L.

1. Introducción

Este juego de roles se centra en el uso de React Hooks, Fetch, Axios y ngrok para crear una aplicación web con comunicación eficiente entre frontend y backend. El equipo estará compuesto por:

- Desarrollador Frontend (React)
- Desarrollador Backend (Node.js/Express)
- Ingeniero DevOps

2. Configuración Inicial

2.1. Tarea del Desarrollador Backend

Crear un servidor Express básico con dos endpoints:

```
const express = require('express');
const app = express();
const port = 3001;

app.use(express.json());

app.get('/api/data', (req, res) => {
  res.json({ message: 'Datos del servidor' });
});

app.post('/api/submit', (req, res) => {
  const { data } = req.body;
  res.json({ success: true, receivedData: data });
});

app.listen(port, () => {
  console.log(`Servidor backend corriendo en http://localhost:${
    port}`);
});
```

2.2. Tarea del Ingeniero DevOps

Configurar ngrok para exponer el servidor backend:

```
ngrok http 3001
```

El Ingeniero DevOps debe proporcionar la URL de ngrok al Desarrollador Frontend.

3. Desafíos

3.1. Desafío 1: Fetch con useEffect

Objetivo: Utilizar Fetch y el hook useEffect para obtener datos del backend.
Tarea del Desarrollador Frontend: Implementar un componente que obtenga datos del endpoint `/api/data` usando Fetch y useEffect.

```
import React, { useState, useEffect } from 'react';

function DataFetcher() {
  const [data, setData] = useState(null);

  useEffect(() => {
    // Implementar la lógica de Fetch aquí
    // Usar la URL de ngrok proporcionada por el Ingeniero DevOps
  }, []);

  return (
    <div>
      <h2>Datos del Servidor:</h2>
      {/* Mostrar los datos obtenidos */}
    </div>
  );
}
```

3.2. Desafío 2: Axios con useCallback

Objetivo: Utilizar Axios y el hook useCallback para enviar datos al backend.
Tarea del Desarrollador Frontend: Crear un formulario que envíe datos al endpoint `/api/submit` usando Axios y useCallback.

```
import React, { useState, useCallback } from 'react';
import axios from 'axios';

function DataSubmitter() {
  const [inputData, setInputData] = useState('');

  const handleSubmit = useCallback(() => {
    // Implementar la lógica de Axios aquí
    // Usar la URL de ngrok proporcionada por el Ingeniero DevOps
  }, [inputData]);

  return (
```

```
    <div>
      <input
        type="text"
        value={inputData}
        onChange={(e) => setInputData(e.target.value)}
      />
      <button onClick={handleSubmit}>Enviar</button>
    </div>
  );
}
```

3.3. Desafío 3: Custom Hook para Manejo de Estado

Objetivo: Crear un custom hook que maneje el estado y las operaciones de fetch/axios.

Tarea del Desarrollador Frontend: Implementar un custom hook que combine la funcionalidad de los dos componentes anteriores.

```
import { useState, useEffect, useCallback } from 'react';
import axios from 'axios';

function useDataManager(ngrokUrl) {
  const [data, setData] = useState(null);
  const [inputData, setInputData] = useState('');

  // Implementar useEffect para fetch
  // Implementar useCallback para axios
  // Implementar funciones para manejar cambios de input y submit

  return {
    data,
    inputData,
    handleInputChange,
    handleSubmit
  };
}
```

3.4. Desafío 4: Manejo de Errores y Carga

Objetivo: Mejorar el manejo de errores y estados de carga en todos los componentes.

Tareas:

- Agregar estados de carga (loading) para operaciones asíncronas.
- Implementar manejo de errores para Fetch y Axios.
- Mostrar mensajes apropiados al usuario basados en el estado de la operación.

3.5. Desafío 5: Optimización de Rendimiento

Objetivo: Optimizar el rendimiento de los componentes utilizando memoización.

Tareas:

- Utilizar React.memo para componentes que no necesitan re-renderizarse frecuentemente.
- Implementar useMemo para cálculos costosos.
- Usar useCallback para funciones que se pasan como props a componentes hijos.

4. Evaluación

Después de completar los desafíos, el equipo debe evaluar:

- La eficiencia de la comunicación entre frontend y backend a través de ngrok.
- El uso apropiado de Fetch vs Axios en diferentes situaciones.
- La implementación correcta y eficiente de los Hooks de React.
- El manejo adecuado de estados, efectos secundarios y optimización de rendimiento.