Actividad 4 - Tecnología front end en la construcción de una aplicación web II

Presentado por:

Francisco Jose Mangones Anaya.

Presentado a:

Joaquin Sanchez.

Ingeniería de software.

Electiva disciplinar II

Corporación Universitaria Iberoamericana.

Lorica-Córdoba.

2025

1. Introducción

La construcción de aplicaciones web modernas requiere integrar de manera coherente tecnologías de front-end y back-end que permitan entregar soluciones escalables, eficientes y centradas en la experiencia del usuario. En continuidad con el trabajo realizado en la Actividad 2, donde se realizó la propuesta inicial del proyecto informático y el análisis del problema organizacional, el presente documento desarrolla los conceptos y herramientas necesarios para plasmar la capa de interacción gráfica que conectará a los usuarios con la lógica de negocio previamente definida.

Para este propósito se explican conceptos fundamentales como REST y Swagger, ReactJS, Hooks, Context API, Axios, rutas y navegación, así como los lineamientos para realizar el despliegue final de la aplicación. Estos conceptos se acompañan de ejemplos prácticos y fragmentos de código que evidencian la articulación entre back-end y front-end.
Finalmente, se presenta un ejemplo de implementación en React que integra cada una de las tecnologías mencionadas.

2. CONTENIDO

2.1. REST con Swagger

¿Qué es REST?

REST (Representational State Transfer) es un estilo de arquitectura que define cómo deben comunicarse los sistemas a través de HTTP. Se basa en métodos como GET, POST, PUT, DELETE, y permite que clientes (front-end) consuman servicios del servidor (back-end).

¿Qué es Swagger?

Swagger es una herramienta que permite documentar y probar APIs REST de manera visual e interactiva. Facilita la comprensión del servicio por parte del equipo y permite verificar su funcionamiento.

Ejemplo básico de endpoint REST documentado con Swagger (Node Express)

```
/**
 * @swagger
 * /api/users:
 *   get:
 *     summary: Obtiene todos los usuarios
```

```
 *     responses:
 *       200:
 *         description: Lista de usuarios
 */
app.get('/api/users', (req, res) => {
  res.json([{ id: 1, name: "Francisco" }]);
});
```

2.2. ReactJS

ReactJS es una librería de JavaScript diseñada para construir interfaces de usuario basadas en componentes. Permite crear aplicaciones dinámicas, modulares y escalables.

Ejemplo básico

```
function App() {
  return <h1>Bienvenido a mi aplicación Web II</h1>;
}
```

2.3. Hooks en React

Los Hooks permiten manejar estado, ciclo de vida y lógica reutilizable.

useState – Maneja estados locales

```
const [contador, setContador] = useState(0);
```

useEffect – Ejecuta acciones cuando un componente cambia

```
useEffect(() => {
  console.log("El componente se montó");
}, []);
```

useContext – Acceso a valores globales

```
const usuario = useContext(UserContext);
```

useReducer – Estados complejos

```
const [state, dispatch] = useReducer(reducer, initialState);
```

2.4. Context API

Permite compartir estados globales sin necesidad de pasar props entre múltiples componentes.

Ejemplo

Contexto global

```
export const UserContext = createContext();

export function UserProvider({ children }) {
  const [user, setUser] = useState(null);
  return (
    <UserContext.Provider value={{ user, setUser }}>
      {children}
    </UserContext.Provider>
  );
}
```

2.5. Peticiones HTTP con Axios

Axios permite consumir APIs REST desde React.

Ejemplo de petición GET

```
axios.get("http://localhost:3000/api/users")
  .then(res => setUsuarios(res.data))
  .catch(err => console.log(err));
```

2.6. Rutas y navegación (React Router)

Permite moverse entre páginas sin recargar la aplicación.

Ejemplo

```
import { BrowserRouter, Routes, Route } from "react-router-dom";

function App() {
  return (
    <BrowserRouter>
      <Routes>
        <Route path="/" element={<Inicio />} />
        <Route path="/usuarios" element={<Usuarios />} />
      </Routes>
    </BrowserRouter>
  );
}
```

3. CÓDIGO DE LA APLICACIÓN WEB (INTEGRADO)

Este ejemplo demuestra el uso de:

✓ React

✓ Hooks

✓ Axios

✓ Context API

✓ Rutas

index.js

```
import React from "react";
import ReactDOM from "react-dom/client";
import App from "./App";
import { UserProvider } from "./context/UserContext";

ReactDOM.createRoot(document.getElementById("root")).render(
```

```
  <UserProvider>
    <App />
  </UserProvider>
);
```

App.js

```
import { BrowserRouter, Routes, Route } from "react-router-dom";
import Home from "./pages/Home";
import Usuarios from "./pages/Usuarios";

function App() {
  return (
    <BrowserRouter>
      <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/usuarios" element={<Usuarios />} />
      </Routes>
    </BrowserRouter>
  );
}

export default App;
```

Usuarios.jsx

```
import { useEffect, useState } from "react";
import axios from "axios";

function Usuarios() {
  const [usuarios, setUsuarios] = useState([]);
```

```
  useEffect(() => {
    axios.get("http://localhost:3000/api/users")
      .then(res => setUsuarios(res.data))
      .catch(err => console.log(err));
  }, []);

  return (
    <div>
      <h2>Lista de usuarios</h2>
      <ul>
        {usuarios.map(u => <li key={u.id}>{u.name}</li>)}
      </ul>
    </div>
  );
}

export default Usuarios;
```

4. Bibliografía

- Martínez Martínez, A. (2021). Proyecto feedback backend y frontend web [Trabajo de grado, Universitat Jaume I].
- Pérez Ibarra, S. G., Quispe, J. R., Mullicundo, F. F., & Lamas, D. A. (2021). Herramientas y tecnologías para el desarrollo web desde el frontend y backend. XXIII Workshop de Investigadores en Ciencias de la Computación.
- Regla, P. D. (2014). Diseño, contenidos y desarrollo del front-end del sitio web del proyecto auralizarte [Trabajo de grado, Universidad Pública de Navarra].
- Valdivia Caballero, J. J. (2021). Modelo de procesos para el desarrollo del front-end de aplicaciones web (pp. 5–40) [Tesis de maestría, Universidad Nacional Mayor de San Marcos].