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

David Stiven Castillo Muñoz

Noviembre 2025

Corporación universitaria iberoamericana

Facultad de ingeniería

Electiva disciplinar II - Desarrollo de aplicaciones web (JOAQUIN SANCHEZ 27102025_C2_202534)

TABLA DE CONTENIDO.

TABLA DE ILUSTRACIONES.

1. INTRODUCCIÓN.

En el desarrollo de aplicativos webs es común usar o combinar herramientas que nos permiten construir soluciones más complejas, organizadas y fáciles de entender y mantener, es importante tener todos nuestros proyectos documentados para esto normalmente usamos Swagger para mostrar de forma estandarizada los datos y funcionalidades, así mismo en un mundo cada vez más dinámico y ágil se hace necesario crear frontends más llamativos que inviten a la interacción ahí donde entra ReactJS que se ha convertido en una solución gracias a su enfoque por componentes y su facilidad de manejo, es importante que nuestra frontend se comunique con nuestro backend, para este fin podemos usar herramientas como Axios que simplifica las peticiones y hace más fluido el intercambio de datos, todas estas herramientas nos ayudan en la creación de proyectos web que nos permiten desarrollar soluciones modernas, eficientes y escalables.

1. REST con Swagger.

Una aplicación web o cualquier aplicación esta compuesta por frontend y backend y para lograr conectar estas dos partes aparece REST que es un conjunto de reglas que facilita la comunicación, es ordenado y predecible, REST propone una estructura en URL y los métodos clásicos de HTTP como son los GET, para obtener información, POST para enviar información, PUT que se usa para actualizar, y DELETE que es para eliminar, por esto las API cumplen con la función de ser una puerta hacia un recurso, ejemplo si estamos en una pagina y queremos ver el catalogo al productos disponible nuestra url podría ser “tiendax/productos” así podríamos obtener los productos disponibles en la tienda.

Sabiendo que es una API, debemos saber cómo podemos documentarla, ahí es donde aparece Swagger que es una herramienta, que convierte la API en un panel interactivo, donde cada ruta aparece documentada con claridad donde nos muestra, qué recibe, qué devuelve y cómo usarla, eso lo convierte en algo interactivo y no tan estático, al tener una estructura bien definida permite que sea de fácil entendimiento y evitando malentendidos entre equipos.

Ilustración 1 API en Swagger

2. ReactJS.

Cuando estas navegando en una pagina web te has fijado en que cuando interactúas con algún elemento este es el que cambia y no es la pagina completa la que se vuelve a cargar para cambiar el elemento ahí es donde interviene ReactJS el cual propone dividir la pagina en componentes el cual cada componente se comporta de manera independiente esto hace que las aplicaciones se sientan más fluidas esto hace que se usen menos recursos y permite armar las paginas por componentes, como si fueran un rompecabezas.

La ventaja de trabajar con ReactJS es que puedes reutilizar, combinar y adaptar los componentes de la misma manera permite crear componentes mezclando HTML con JavaScript esto hace que el desarrollo sea más intuitivo, fácil de mantener.

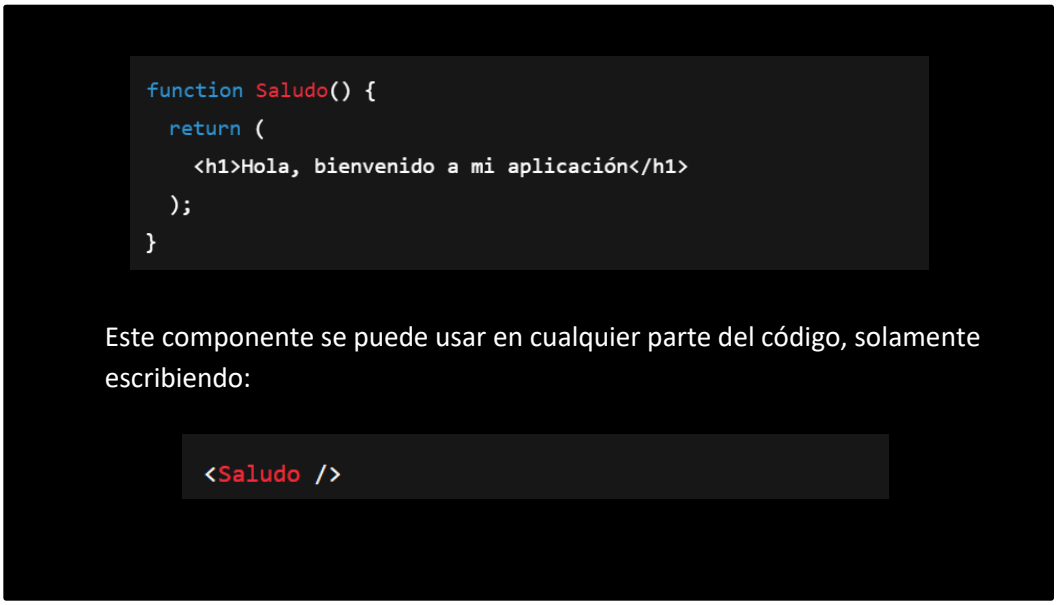

Ilustración 2 Ejemplo creación de componente

3. Hooks (useState, useContext, useEffect, useReducer).

Los Hooks son como un conjunto de herramientas que tiene React las cuales sirven para darle vida a los componentes, antes de los Hooks habían que usar clases, métodos especiales y mucha estructura adicional, ahora con los Hooks es mucho más simple solo basta con llamar una función, cada Hook tiene un propósito unas sirven para recordar datos, otras para reaccionar a eventos del sistema, y otras para compartir información.

- UseState: Es el más básico y sirve para guardar valores que pueden cambiar con el tiempo, es como tener una pequeña libreta donde puedes anotar datos y tenerlos incluso si la pantalla vuelve a cargarse lo podemos ver cuando un usuario escribe en un formulario, si un botón está activado o no, o cuántos productos hay en un carrito.

- UseEffect: Nos permite ejecutar acciones cuando algo ocurre en el componente, es muy ideal para cargar datos de una API, iniciar contadores, escuchar eventos o actualizar el título de la página, un ejemplo de ello es cuando abres una pagina de recetas UseEffect dice “Apenas abras esta pantalla, ve a buscar las recetas”.

- UseContext: Podemos compartir información con varios componentes es como si existiera un comunicado entre todos los componentes y tuvieran la misma información, un ejemplo de ello es cuando iniciamos sesión en una página todos los componentes que requieran la autenticación pueden acceder sin pedirle al usuario que vuelva a iniciar la sesión.

- UseReducer: es como un centro de control dentro del componente en lugar de cambiar una variable, definimos acciones (como “agregar”, “editar” o “eliminar”) y una función que describe cómo cambia el estado cuando se ejecuta cada acción.

4. Context API.

Cuando algunos datos necesitan ser compartidos con componentes muy lejanos es conveniente usar una solución que sea más limpia y fácil de entender que se convine con Hooks y sea nativo de React así como evitar usar props innecesariamente, un ejemplo de ello es el idioma de la interfaz, el diseño claro u oscuro, si se intentara pasar esta información manualmente de componente en componente, el código se volvería pesado y desordenado, por esto Contex API crea un espacio compartido al que los componentes puedan acceder directamente, sin intermediarios.

Vamos a ver un ejemplo de Context API, para aplicar una aplicación que tiene 2 temas.

```
import { createContext, useState } from "react";

export const ThemeContext = createContext();

export function ThemeProvider({ children }) {
  const [theme, setTheme] = useState("light");

  const toggleTheme = () => {
    setTheme(prev => (prev === "light" ? "dark" : "light"));
  };

  return (
    <ThemeContext.Provider value={{ theme, toggleTheme }}>
      {children}
    </ThemeContext.Provider>
  );
}
```

Ilustración 3 Ejemplo Context API creación de contexto

```jsx
import { ThemeProvider } from "./ThemeContext";
import Home from "./Home";

function App() {
  return (
    <ThemeProvider>
      <Home />
    </ThemeProvider>
  );
}

export default App;
```

Ilustración 4 Ejemplo Context API Envolver la aplicación

```jsx
import { useContext } from "react";
import { ThemeContext } from "./ThemeContext";

function Home() {
  const { theme, toggleTheme } = useContext(ThemeContext);

  return (
    <div style={{
      background: theme === "light" ? "#fff" : "#333",
      color: theme === "light" ? "#000" : "#fff",
      padding: "20px"
    }}>
      <h1>Tema actual: {theme}</h1>
      <button onClick={toggleTheme}>Cambiar Tema</button>
    </div>
  );
}

export default Home;
```

Ilustración 5 Ejemplo Context API aplicación del contexto

5. Peticiones HTTP con Axios.

En las aplicaciones modernas podemos ver que no funcionan como islas ya que la mayoría están interconectadas o dependientes de otras, esta comunicación ocurre mediante peticiones HTTP, aquí es donde aparece Axios, una librería muy popular en JavaScript que hace que estas conversaciones sean mucho más fáciles, en lugar de escribir instrucciones largas y complicadas, nos permite usar una sintaxis muy natural, es como decirle, ve a esta URL y tráeme los datos, por favor", es rápido, funciona en el navegador y en Node.js, y además maneja respuestas, errores y configuraciones avanzadas, ejemplo: Mostrar una lista de usuarios de usuarios desde una API pública.

```
import axios from "axios";

export const api = axios.create({
  baseURL: "https://jsonplaceholder.typicode.com"
});
```

Ilustración 6 Ejemplo de Axios1

```
import { useEffect, useState } from "react";
import { api } from "./api";

function Users() {
  const [users, setUsers] = useState([]);

  useEffect(() => {
    api.get("/users")
      .then(response => setUsers(response.data))
      .catch(() => console.log("Error al obtener usuarios"));
  }, []);

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

export default Users;
```

Ilustración 7 Ejemplos de Axios2

6. Rutas y navegación.

En las aplicaciones webs necesitamos movernos entre las distintas pantallas es por esta razón que se hace necesario que el navegador mantenga la URL, pero React actualiza solo la parte de la interfaz que debe cambiar. Esto permite crear apps más rápidas, ordenadas y fáciles de navegar.

Para lograr eso, React utiliza librerías como React Router, que actúan como un mapa interno de la aplicación. Con este mapa podemos indicarle a la pagina que de acuerdo a cada ruta nos muestre la información que le corresponde, ahora vamos a ver un ejemplo de rutas.

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

function App() {
  return (
    <BrowserRouter>
      <nav>
        <Link to="/">Inicio</Link> | <Link to="/perfil">Perfil</Link>
      </nav>
```

Ilustración 8 Ejemplo Creación de rutas

```
function Home() {
  return <h1>Pantalla de Inicio</h1>;
}

export default Home;
```

Ilustración 9 Ejemplo Creación de rutas_2

7. Despliegue.

En todo proceso de desarrollo de software tenemos la fase de despliegue e implementación en pocas palabras es llevar nuestro aplicativo web a publicación, para que cualquiera pueda usarlo y verlo en el caso de aplicaciones con React, el proceso suele ser muy sencillo, tomamos el proyecto, lo convertimos a archivos optimizados (HTML, CSS y JavaScript) y lo enviamos a un servicio en la nube, el despliegue no es solo subir archivos, también es generar una versión lista para producción, donde el código está organizado y preparado para funcionar de manera correcta, así aseguramos que la experiencia del usuario sea fluida, incluso desde dispositivos móviles o conexiones lentas, el despliegue es el puente entre el desarrollo y el uso real de la aplicación.

8. Código y GitHub.

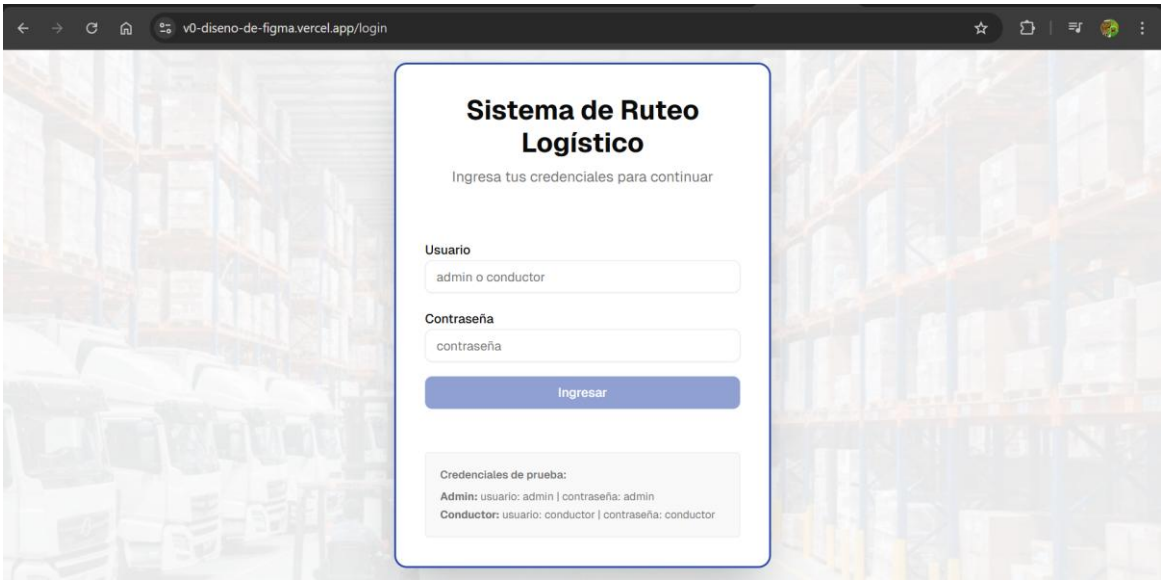

Ilustración 10 Sistema RutaLog

Para ingresar al sistema de clic aquí.