

Tecnología front end en la construcción de una aplicación web

Lineer Javir Rolong Ebratt

Código: 100138239

Corporación Universitaria Iberoamericana

Facultad de Ingeniería de Software

Electiva disciplinar II - Desarrollo de aplicaciones
web

JOAQUIN SANCHEZ

Barranquilla, Colombia

30 de noviembre 2025

INTRODUCCIÓN

En el ámbito del desarrollo de aplicaciones web modernas, resulta fundamental comprender los conceptos, herramientas y metodologías que permiten construir sistemas escalables, eficientes y fáciles de mantener. La creación de interfaces dinámicas, la correcta organización de componentes, la comunicación entre el frontend y el backend, así como el despliegue en entornos de producción, son aspectos esenciales en el ciclo de vida del software.

El presente documento aborda conceptos clave utilizados ampliamente en proyectos actuales: servicios REST documentados con Swagger, desarrollo frontend con ReactJS, uso de Hooks, Context API para manejo global del estado, peticiones HTTP mediante Axios, estrategias de rutas y navegación, y consideraciones para el despliegue de aplicaciones web. Cada concepto es explicado y acompañado de ejemplos que facilitan su comprensión y aplicación práctica.

REST CON SWAGGER

¿Qué es REST?

REST (Representational State Transfer) es un estilo arquitectónico para la construcción de servicios web que se comunican mediante HTTP. Se caracteriza por el uso de métodos como GET, POST, PUT y DELETE, así como por el intercambio de información en formatos como JSON.

¿Qué es Swagger?

Swagger es una herramienta para documentar APIs REST. Permite describir los endpoints, parámetros, respuestas y modelos de datos. Facilita la lectura, prueba y comprensión de la API por parte de desarrolladores.

Ejemplo sencillo:

Ruta REST (Express + Swagger):

```
app.get('/api/pacientes', (req, res) => {  
  res.json([  
    { id: 1, nombre: "Juan Pérez" }  
  ]));  
});
```

Documento Swagger:

```
paths:  
  /api/pacientes:  
    get:  
      summary: Obtiene la lista de pacientes  
      responses:  
        200:  
          description: Lista de pacientes
```

REACTJS

ReactJS es una biblioteca de JavaScript creada por Meta para construir interfaces de usuario basadas en componentes. Se caracteriza por el uso del DOM virtual y por manejar la vista de forma declarativa.

Funcionalidad principal

- Crear componentes reutilizables
- Manejar estados y eventos
- Renderizar vistas dinámicamente

Ejemplo:

```
function Saludo() {  
  return <h1>Hola, bienvenido al sistema</h1>;  
}
```

HOOKS (USESTATE, USECONTEXT, USEEFFECT, USEREDUCER)

Los Hooks permiten manejar el estado, efectos y lógica interna de los componentes funcionales de React.

useState

Permite manejar valores que cambian.

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

useEffect

Ejecuta efectos secundarios como llamadas API.

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

useContext

Permite acceder a valores globales.

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

useReducer

Útil para estados complejos.

```
const [estado, dispatch] = useReducer(reducer, estadoInicial);
```

CONTEXT API

Context API permite compartir información entre componentes sin necesidad de pasar props manualmente.

Ejemplo:

```
export const UserContext = createContext();

function App() {
  return (
    <UserContext.Provider value={{ nombre: "Lineer" }}>
      <Dashboard />
    </UserContext.Provider>
  );
}
```

PETICIONES HTTP CON AXIOS

Axios es una librería para realizar peticiones HTTP de manera sencilla.

Ejemplo:

```
import axios from "axios";

useEffect(() => {
  axios.get("http://localhost:5000/api/pacientes")
    .then(res => setPacientes(res.data));
}, []);
```

RUTAS Y NAVEGACIÓN

React utiliza bibliotecas como **React Router** para navegar entre vistas sin recargar la página.

Ejemplo:

```
<Routes>
  <Route path="/" element={<Inicio />} />
  <Route path="/login" element={<Login />} />
</Routes>
```

DESPLIEGUE

El despliegue consiste en publicar la aplicación en un servidor para que pueda ser accedida por usuarios. Incluye:

- Construcción del frontend (npm run build)
- Configuración del backend
- Uso de servicios como Netlify, Vercel, Render o VPS propios
- Variables de entorno
- Conexión a bases de datos remotas

Repositorio de github

<https://github.com/LineerJRE/Desarrollo-de-aplicacion>

CONCLUSIONES

El desarrollo moderno de aplicaciones web requiere la comprensión integral de herramientas que permiten estructurar, construir y desplegar sistemas eficientes. ReactJS y sus Hooks facilitan la creación de interfaces dinámicas; Context API resuelve el manejo global del estado; Axios permite comunicaciones fluidas con APIs REST, mientras que Swagger facilita la documentación y mantenimiento del backend.

Asimismo, la correcta gestión de rutas y el despliegue aseguran que la aplicación sea accesible en entornos reales. En conjunto, estos conceptos representan la base tecnológica de la mayoría de plataformas web actuales, siendo esenciales para el desarrollo profesional en ingeniería de software.

BIBLIOGRAFÍA

- OpenAPI Initiative. (2021). *Swagger: API documentation tools*. <https://swagger.io>
- Meta. (2023). *React documentation*. <https://react.dev>
- Axios. (2023). *Axios documentation*. <https://axios-http.com>
- React Router. (2023). *React Router documentation*. <https://reactrouter.com>