



Planeta Formación y Universidades

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

Estudiante: William Javier Amaya Castaño

Facultad Ingeniería de Software,

Corporación Universitaria Iberoamericana

Electiva disciplinar II - Desarrollo de aplicaciones web

Joaquin Sanchez

Noviembre de 2025

## Introducción

En el marco del proyecto HappyPaws Web II, esta actividad busca profundizar en los fundamentos del desarrollo web moderno, abordando tecnologías esenciales para la creación de aplicaciones escalables, modulares y usables. A través de la comprensión de conceptos como REST, Swagger, ReactJS, Hooks, Axios y React Router, se construirá la base teórica y práctica para el desarrollo de la interfaz gráfica y su interacción con el backend.

### **Conceptos exigidos en la actividad**

A continuación, se desarrollan los conceptos requeridos por el docente, explicando qué son, para qué sirven y un ejemplo aplicado al proyecto HappyPaws.

## REST con Swagger

### ¿Qué es REST?

REST (Representational State Transfer) es un estilo de arquitectura que define cómo deben comunicarse los sistemas mediante HTTP. Es la forma estándar de construir APIs modernas.

### Características:

- Comunicación cliente–servidor
- Uso de métodos HTTP: GET, POST, PUT, DELETE
- Respuestas en JSON
- Stateless (no guarda estado entre peticiones)

### Swagger

Swagger es una herramienta que documenta automáticamente APIs REST y permite probarlas desde un panel visual.

### Ejemplo REST para HappyPaws

#### Endpoint:

GET /api/mascotas

#### Respuesta JSON:

```
[  
 {  
   "id": 1,  
   "nombre": "Luna",  
   "especie": "Perro",  
   "estado": "En adopción"  
 }  
 ]
```

## ReactJS

ReactJS es una librería de JavaScript creada por Meta para construir interfaces de usuario basadas en componentes reutilizables.

### Ventajas:

- Rápido
- Reutilizable
- Fácil de escalar
- Soporta manejo avanzado de estados

### Ejemplo HappyPaws

Componente que muestra el nombre de una mascota:

```
function Mascota({ nombre }) {  
  return <h2>{nombre}</h2>;  
}
```

## Hooks: useState – useContext – useEffect – useReducer

### useState

Permite manejar estados locales.

Ejemplo:

```
const [count, setCount] = useState(0);
```

### useEffect

Ejecuta código cuando el componente se monta, actualiza o desmonta.

Ejemplo:

```
useEffect(() => {  
  console.log("Componente cargado");  
}, []);
```

### useContext

Comparte estados globales entre componentes.

Ejemplo:

```
const UserContext = createContext();
```

### useReducer

Maneja estados más complejos.

Ejemplo:

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

### Context API

Herramienta nativa de React para manejar variables globales sin necesidad de usar Redux.

Ejemplo creando un contexto global:

```
const AppContext = createContext();
```

```
function Provider({ children }) {
  const [usuario, setUsuario] = useState(null);

  return (
    <AppContext.Provider value={{ usuario, setUsuario }}>
      {children}
    </AppContext.Provider>
  );
}
```

### Peticiones HTTP con Axios

Axios es una librería usada para hacer peticiones al backend.  
Más amigable y poderosa que fetch.

Ejemplo consumiendo mascotas de HappyPaws:

```
import axios from "axios";
```

```
useEffect(() => {
  axios.get("http://localhost:3000/api/mascotas")
```

```
.then(res => setMascotas(res.data))  
}, []);
```

## Rutas y Navegación (React Router)

Permite cambiar entre páginas sin recargar el navegador.

Ejemplo:

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

```
function App() {  
  return (  
    <BrowserRouter>  
      <Routes>  
        <Route path="/" element={<Home />} />  
        <Route path="/mascotas" element={<Mascotas />} />  
      </Routes>  
    </BrowserRouter>  
  );  
}
```

## Despliegue

El despliegue consiste en publicar la aplicación para que cualquier usuario pueda acceder a ella.

Opciones:

- **Netlify** (frontend)
- **Vercel** (frontend)
- **Railway / Render / AWS** (backend)

Ejemplo despliegue Netlify:

1. Crear repositorio en GitHub
2. Subir código
3. Conectar Netlify → “Deploy site from GitHub”

#### 4. Build automático

### 3. Código base del proyecto (HappyPaws)

Aquí incluyo código mínimo para cumplir con la actividad:

#### Backend (Node + Express) – server.js

```
const express = require("express");
const app = express();
app.use(express.json());

const mascotas = [
  { id: 1, nombre: "Luna", estado: "En adopción" },
  { id: 2, nombre: "Max", estado: "Adoptado" }
];

app.get("/api/mascotas", (req, res) => {
  res.json(mascotas);
});

app.listen(3000, () => console.log("Servidor corriendo en puerto 3000"));
```

#### Frontend (React) – Mostrar mascotas

##### Mascotas.jsx

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

function Mascotas() {
  const [mascotas, setMascotas] = useState([]);

  useEffect(() => {
    axios.get("http://localhost:3000/api/mascotas")
      .then(res => setMascotas(res.data));
  }, []);
```

```
}, []);
```

```
return (  
  <div>  
    <h1>Lista de Mascotas</h1>  
    <ul>  
      {mascotas.map(m => (  
        <li key={m.id}>{m.nombre} - {m.estado}</li>  
      ))}  
    </ul>  
  </div>  
);  
}
```

```
export default Mascotas;
```

## [Repositorio GitHub](#)

## Bibliografía

Martínez Martínez, A. (2021). 2. Planificación del proyecto. En Martínez Martínez, A. Proyecto feedback backend y frontend web. [Trabajo de grado, Universitat Jaume]. (pp. 17-33)

Lagos Galindo, J. E. (2017). Desarrollo del Backend de una aplicación para vinculación de clientes en una entidad bancaria. [Trabajo de grado, Universidad Católica de Colombia].