Electiva disciplinar II

Desarrollo de aplicaciones web

Unidad de aprendizaje II

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

Hernán Ricardo Loaiza Doncel

Facultad de ingeniería

Docente tutor. Joaquín Sánchez
Curso: Electiva disciplinar
Ingeniería de software

Corporación Universitaria Iberoamericana
Bogotá – Colombia
Noviembre 30 de 2025

Desarrollo de la actividad

Introducción

El desarrollo de la aplicación web exige la integración de tecnologías que permitan construir interfaces interactivas y eficientes, ya que se conecta con el sistema backend mediante unos protocolos estandarizados. En el contexto de la actividad anterior que se desarrolla una propuesta, aplicación web para la gestión de pedidos e inventarios en un negocio de hamburguesería, ya que es necesario entender que las herramientas como ReactJS, hooks, la ContextAPI,la documentación con Swagger y las peticiones de HTTP con Axios, las rutas de navegación y el despliegue.

Esta actividad se desarrolla con estos conceptos desde una perspectiva práctica, acompañados de ejemplos aplicados al proyecto de la hamburguesería y se finaliza con un fragmento del código que integra frontend y backend.

Contenido

REST con Swagger

REST es un estilo arquitectónico que nos permite lacomunicoon entre cliente y servidor mediante operaciones HTTP como GET, POST, PUT y DELETE.

En el backend del proyecto de hamburguesería, REST permite gestionar pedidos, productos e inventarios a través de rutas bien definidas.

Swagger, ahora conocidos como el OpenAPI, se usa para documentar visualmente las APIs y nos permite su prueba directa desde una interfaz web.

Ejemplo aplicado al proyecto
En tu proyecto de hamburguesería podrías tener rutas como:

GET /pedidos: obtener la lista de pedidos
POST /pedidos: registrar un nuevo pedido
GET /inventario: obtener el inventario
POST /inventario: agregar insumos

Swagger te permitiría visualizar estas rutas así:

/pedidos:

get: retorna la lista de pedidos

post: crea un nuevo pedido

Con Swagger,el equipo puede visualizar y probar los endpoints, si la necesidad de herramientas externas.

ReactJS

Es una biblioteca de JavaScript se usa para construir interfaces de usuario mediante componentes reutilizables.

React se usa para construir pantallas como.

- Registro de los pedidos
- Lista de inventarios
- Reportes

Ejemplo aplicado
En el proyecto de hamburguesería se crearon varios componentes:

PedidoCard : muestra un pedido con su información

Home: página principal

Inventario: formulario y lista de insumos

Pedidos: formulario para registrar hamburguesas, bebidas y extras

App: contiene las rutas de toda la app

React te permite hacer cosas como:

<h2>Registro de Pedidos 🍔</h2>

Hooks

useState

Nos permite manejar estados internos del componente.

Ejemplo aplicado a tu proyecto:

En el módulo de pedidos usaste useState para guardar la hamburguesa seleccionada:

const [hamburguesa, setHamburguesa] = useState("");

useEffect

useEffect ejecuta acciones secundarias (side effects) en un componente. React lo usa para tareas que ocurren fuera del ciclo principal del renderizado

Ejemplo aplicado a tu proyecto:
Si más adelante conectas tu backend con Axios, podrías usarlo así:

```
useEffect(() => {
  axios.get("/inventario").then(res => setInventario(res.data));
}, []);
```

useContext

useContext permite compartir datos entre múltiples componentes sin necesidad de usar “props” por todas partes

Ejemplo aplicado a tu proyecto:

Usaste useContext en Inventario para acceder al contexto global:

const { inventario, agregarItem } = useContext(InventarioContext);

useReducer

ideal para manejar estados complejos, como agregar un adicional.

Ejemplo aplicado

Ejemplo aplicado a tu proyecto:

Si más adelante manejas el inventario como un flujo completo:

```
const reducer = (state, action) => {
  switch(action.type) {
    case "AGREGAR":
      return [...state, action.payload];
    case "BORRAR":
      return state.filter(item => item.id !== action.id);
  }
};
```

Context API

Sirve para manejar un estado global sin necesidad de pasar propiedades por muchos niveles.

usuario autenticado
inventario

carrito de compras
configuraciones globales

Ejemplo aplicado al inventario

```
export function InventarioProvider({ children }) {
  const [inventario, setInventario] = useState([]);
  const agregarItem = (item) => {
    setInventario([...inventario, item]);
  };
  return (
    <InventarioContext.Provider value={{ inventario, agregarItem }}>
      {children}
    </InventarioContext.Provider>
  );
}
```

Peticiones HTTP con Axios

Axios nos permite consumir APIs de forma simple.

Ejemplo aplicado a obtener pedidos

Obtener el inventario desde el backend:

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

Enviar un nuevo pedido al servidor:

```
axios.post("http://localhost:3000/pedidos", {
  hamburguesa,
  bebida,
  extras,
  total,
  turno
});
```

Guardar nuevos insumos en el inventario:

```
axios.post("http://localhost:3000/inventario", {
  nombre: nuevoItem
});
```

Aplicación dentro de tu proyecto

En tu sistema, cuando un usuario registra un pedido:

selecciona hamburguesa

selecciona extras

selecciona bebida

se genera turno

se calcula total

Rutas y navegación

Las rutas representan las diferentes “páginas” o vistas que un usuario puede visitar en una aplicación web. En una aplicación tradicional, navegar entre páginas implica cargar un archivo HTML completamente nuevo.

sin recargar la página
más rápida

más fluida
con transiciones instantáneas

Ejemplo aplicado al proyecto

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

import Home from "./pages/Home";

import Pedidos from "./pages/Pedidos";

import Inventario from "./pages/Inventario";


export default function App() {

  return (

    <BrowserRouter>

      <nav>

        <Link to="/">Inicio</Link>

        <Link to="/pedidos">Pedidos</Link>

        <Link to="/inventario">Inventario</Link>

      </nav>


      <Routes>

        <Route path="/" element={<Home />} />

        <Route path="/pedidos" element={<Pedidos />} />

        <Route path="/inventario" element={<Inventario />} />

      </Routes>

    </BrowserRouter>

  );

}
```

Despliegue

Esta parte consiste en publicar la aplicación para que los usuarios puedan usarla desde internet y para eso tenemos varias opciones comunes como:

Para frontend:

Vercel
Netlify
GitHub Pages

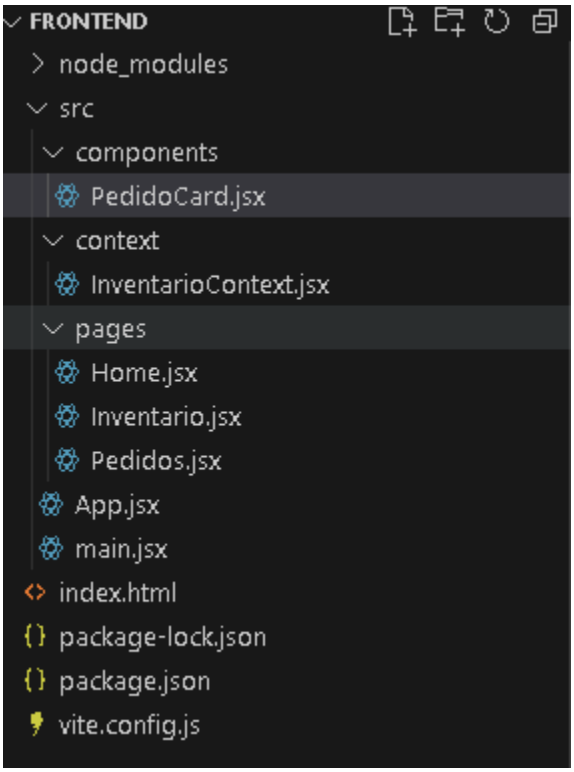
Para backend:

Render
Railway
Heroku
AWS / Azure / Google Cloud

React (el proyecto frontend) se puede desplegar fácilmente, incluso gratis.

Evidencia del proyecto y su desarrollo

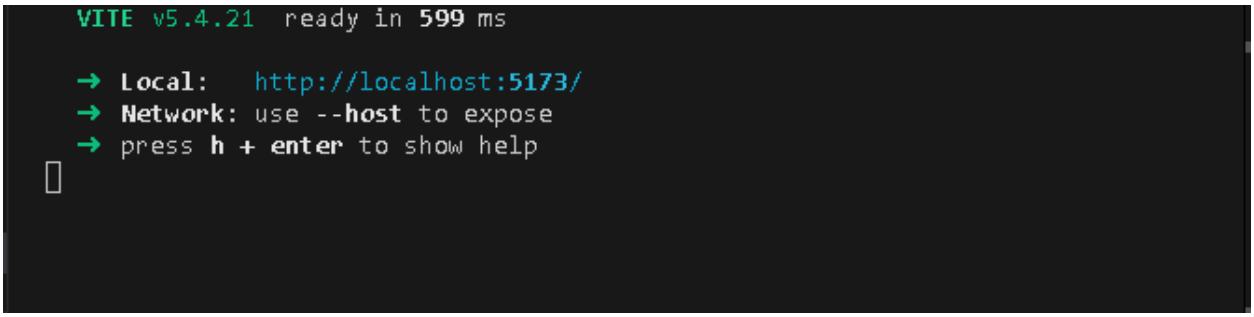
Esta seria toda su documentación del proyecto con respectivo código listo para funciona de manera didáctica aun con sus ciertas mejoras a futuro.



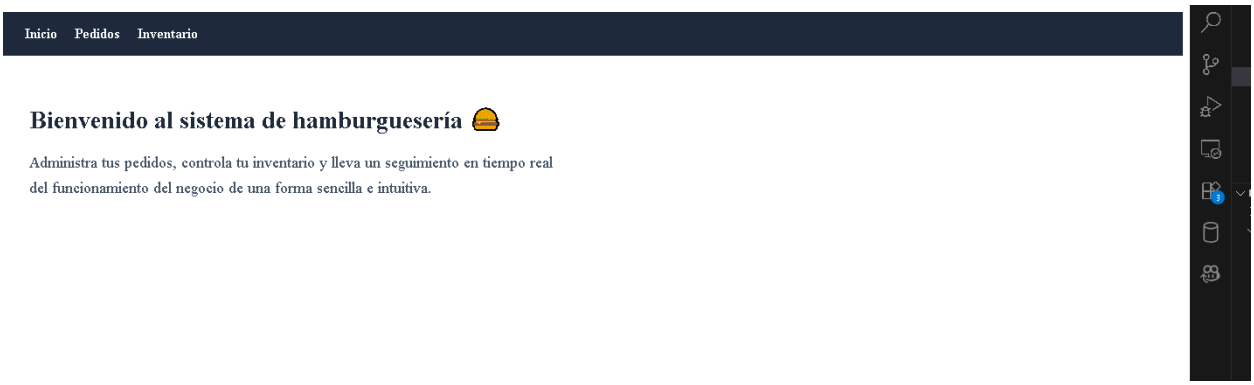
Antes de arrancar la aplicación se debe instalar una liberaría con este comando en la terminal **npm install.** El proyecto necesita ciertas librerías para funcionar.

Por ejemplo:
React(se utilizó en el proyecto)
React Router
Axios
Vite
Express
MongoDB

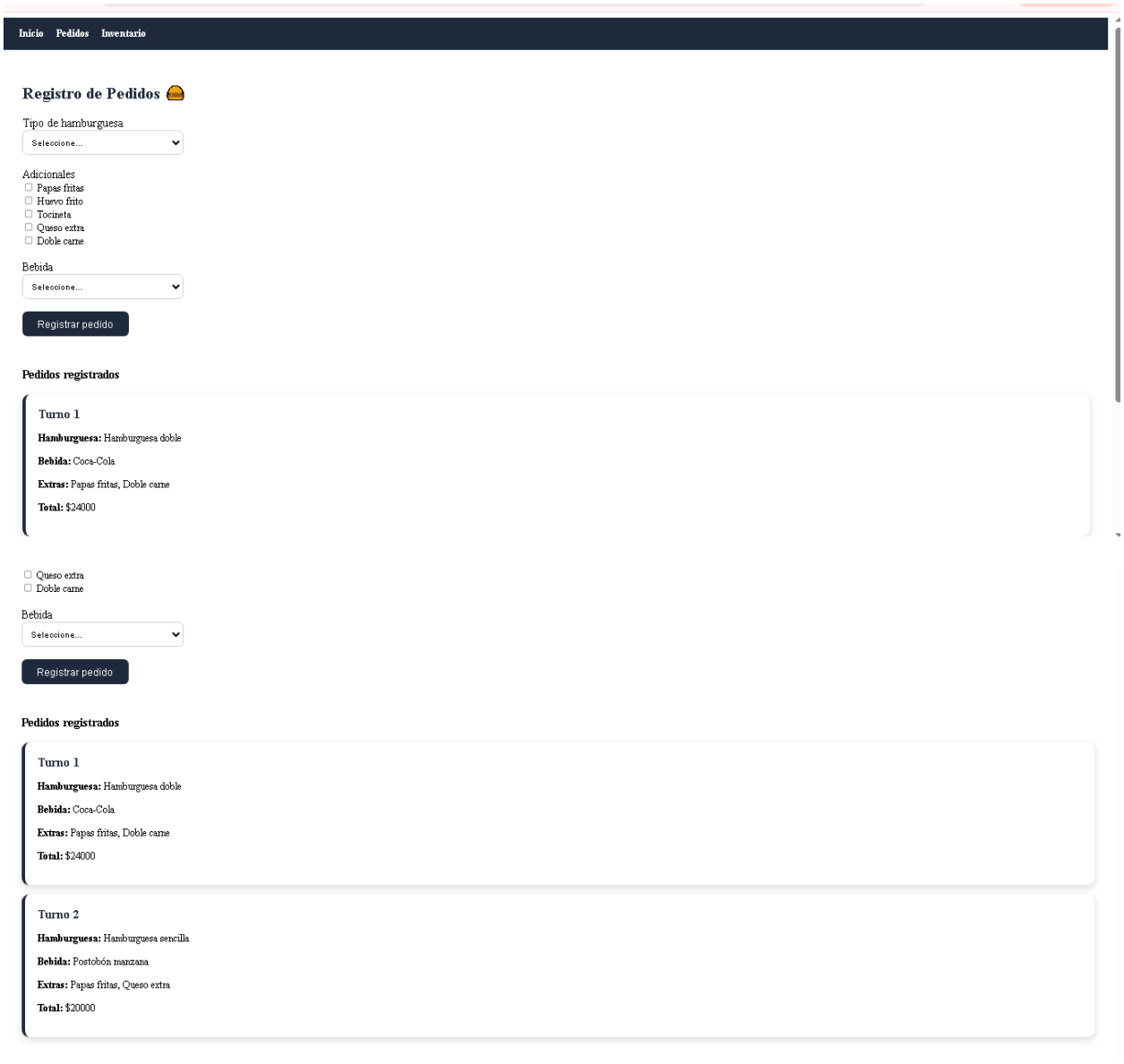
Aquí ya podemos interactuar con el navegador ya que teníamos el locallhost:5173



tenemos qui un inicio de bienvenida al enterar a la aplicación ende a eso tenemos las barras de navegación que son pedidos y invetario.



Aquí tenemos una breve interacción con la aplicación tipo real con turno y su respectivo pedido se realizan dos pedidos.



En esta barra de navegación tenemos el inventario este algo sencillo hay que agregar cosas como más botones el descontar insumo o producto entre otras cosas.

Bibliografia

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. [Tesis de maestría, Universidad Nacional Mayor de San Marcos]. (pp. 5-40).

Martínez Martínez, A. (2021). Proyecto feedback backend y frontend web. [Trabajo de grado, Universitat Jaume].