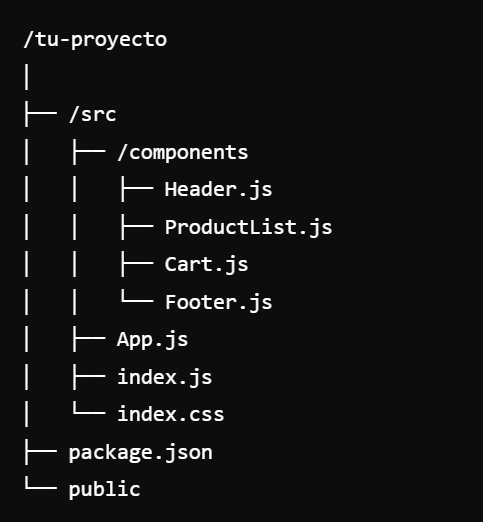
**Ejemplo con REACT**

Para crear una página web de una tienda online dividida en **encabezado (header)** , **lista de productos** , **carrito de compras** y **pie de página (footer)** usando componentes de React,

**Archivos para crear:**

1. **App.js** (componente principal que importa los otros componentes)
2. **componentes/Header.js** (encabezado)
3. **componentes/ProductList.js** (lista de productos)
4. **componentes/Cart.js** (carrito de compras)
5. **componentes/Footer.js** (pie de página)
6. **index.js** (punto de entrada de la aplicación)

**Estructura del proyecto:**

****

### 1. App.js (Componente principal – Maneja el estado del carrito)

Este archivo importa los demás componentes y los organiza en la página.

import React, { useState } from 'react';

import Header from './components/Header';

import ProductList from './components/ProductList';

import Cart from './components/Cart';

import Footer from './components/Footer';

function App() {

  const [cartItems, setCartItems] = useState([]);

  // Función para agregar un producto al carrito

  const addToCart = (product) => {

    const existingProduct = cartItems.find(item => item.id === product.id);

    if (existingProduct) {

      setCartItems(cartItems.map(item =>

        item.id === product.id ? { ...item, quantity: item.quantity + 1 } : item

      ));

    } else {

      setCartItems([...cartItems, { ...product, quantity: 1 }]);

    }

  };

  return (

    <div>

      <Header />

      <main style={styles.main}>

        <ProductList addToCart={addToCart} />

        <Cart cartItems={cartItems} />

      </main>

      <Footer />

    </div>

  );

}

const styles = {

  main: {

    display: 'flex',

    justifyContent: 'space-between',

    padding: '20px',

  }

};

export default App;

### 2. Header.js (Encabezado)

Este componente será el encabezado de la tienda, con el nombre del logo y un buscador.

import React from 'react';

function Header() {

  return (

    <header style={styles.header}>

      <h1>Tienda Online</h1>

      <input type="text" placeholder="Buscar productos..." style={styles.search} />

    </header>

  );

}

const styles = {

  header: {

    backgroundColor: '#333',

    color: 'white',

    padding: '10px',

    display: 'flex',

    justifyContent: 'space-between',

    alignItems: 'center',

  },

  search: {

    padding: '5px',

    borderRadius: '5px',

    border: 'none',

  }

};

export default Header;

### 3. ProductList.js (Botón para añadir al carrito)

Aquí pasamos la función addToCart desde App.js como **prop** para que se ejecute cuando se haga clic en "Añadir al carrito".

import React from 'react';

function ProductList({ addToCart }) {

  const products = [

    { id: 1, name: 'Producto 1', price: 100 },

    { id: 2, name: 'Producto 2', price: 200 },

    { id: 3, name: 'Producto 3', price: 300 }

  ];

  return (

    <section style={styles.productList}>

      {products.map(product => (

        <div key={product.id} style={styles.product}>

          <h2>{product.name}</h2>

          <p>Precio: ${product.price}</p>

          <button onClick={() => addToCart(product)}>Añadir al carrito</button>

        </div>

      ))}

    </section>

  );

}

const styles = {

  productList: {

    display: 'flex',

    justifyContent: 'space-around',

    margin: '20px'

  },

  product: {

    border: '1px solid #ddd',

    padding: '10px',

    borderRadius: '5px',

    textAlign: 'center',

  }

};

export default ProductList;

### 4. Cart.js (Mostrar productos en el carrito)

El componente Cart ahora recibe los productos del carrito como **prop** y los muestra.

import React from 'react';

function Cart({ cartItems }) {

  return (

    <aside style={styles.cart}>

      <h2>Carrito de compras</h2>

      {cartItems.length === 0 ? (

        <p>El carrito está vacío</p>

      ) : (

        <ul>

          {cartItems.map(item => (

            <li key={item.id}>

              {item.name} x {item.quantity} - ${item.price \* item.quantity}

            </li>

          ))}

        </ul>

      )}

    </aside>

  );

}

const styles = {

  cart: {

    border: '1px solid #ddd',

    padding: '10px',

    borderRadius: '5px',

    width: '300px',

    margin: '20px',

  }

};

export default Cart;

### 5. Footer.js (Pie de página)

Este componente es el pie de la página de la tienda online.

import React from 'react';

function Footer() {

  return (

    <footer style={styles.footer}>

      <p>&copy; 2024 Tienda Online. Todos los derechos reservados.</p>

    </footer>

  );

}

const styles = {

  footer: {

    backgroundColor: '#333',

    color: 'white',

    textAlign: 'center',

    padding: '10px',

    position: 'fixed',

    width: '100%',

    bottom: 0,

  }

};

export default Footer;

### 6. index.js (Punto de entrada)

Este archivo es donde React renderiza la aplicación en el DOM. Debes tenerlo en la raíz de tu carpeta src.

import React from 'react';

import ReactDOM from 'react-dom';

import './index.css';

import App from './App';

ReactDOM.render(

  <React.StrictMode>

    <App />

  </React.StrictMode>,

  document.getElementById('root')

);

**Explicación del código:**

1. **Estado del carrito** :
   * useState([])en App.js maneja los productos en el carrito.
   * cartItems almacena el estado actual del carrito, y setCartItems actualiza el carrito.
2. **Agregar productos al carrito** :
   * La función addToCart recibe un producto, y verifica si ya está en el carrito:
     + Si el producto ya está en el carrito, simplemente aumente su cantidad.
     + Si no está en el carrito, lo agrega con una cantidad inicial de 1.
3. **Lista de productos** :
   * El componente ProductList muestra los productos y tiene un botón que llama a addToCart con el producto correspondiente.
4. **Carrito de compras** :
   * El componente Cart muestra los productos que se han agregado al carrito y multiplica el precio por la cantidad de cada producto.
   * Si el carrito está vacío, se muestra un mensaje que dice "El carrito está vacío".

**Cuando se oprime "Añadir al carrito", el carrito se actualiza dinámicamente y muestra los productos seleccionados.**

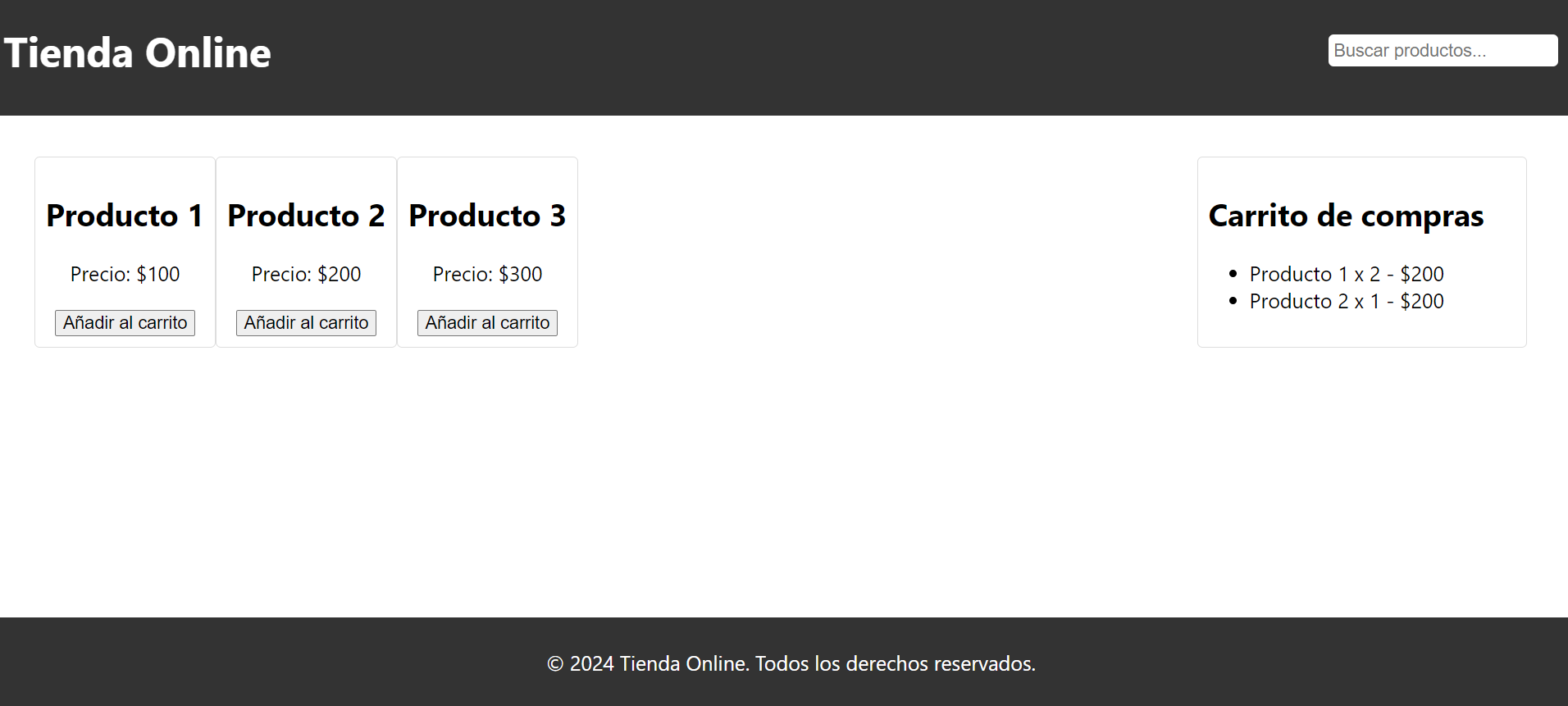
**¿Qué hacer en Visual Studio Code?**

1. **Abrir Visual Studio Code** .
2. **Crea una carpeta del proyecto** (por ejemplo: tienda-online).
3. Dentro de esa carpeta, ejecuta en la terminal:

npx create-react-app tienda-online

1. Dentro de la carpeta src, crea la carpeta components y dentro de esa carpeta crea los archivos:
   * Header.js
   * ProductList.js
   * Cart.js
   * Footer.js
2. Reemplace el contenido de App.js y index.js por el código proporcionado.
3. Inicia el proyecto con:

npm start



**Explicación de App.js**

El archivo **App.js** es el componente principal de la aplicación en React, y su propósito es **gestionar el estado global** (en este caso, el estado del carrito de compras) y distribuirlo entre los componentes hijos: **Header** , **ProductList** , **Cart** , y **Footer** .

**1. Importaciones**

import React, { useState } from 'react';

import Header from './components/Header';

import ProductList from './components/ProductList';

import Cart from './components/Cart';

import Footer from './components/Footer';

* **useState**: Importamos este gancho desde React para gestionar el estado del carrito de compras.
* Los componentes **Header** , **ProductList** , **Cart** y **Footer** se importan porque los vamos a utilizar en la estructura de la página.

**2. Definir el estado del carrito**

function App() {

const [cartItems, setCartItems] = useState([]);

* **cartItems**: Este es el estado que almacenará los productos que el usuario agregue al carrito.
  + Inicialmente, es un **array vacío** porque el carrito empieza sin productos.
* **setCartItems**: Es la función que React nos da para actualizar el valor de cartItems. Cada vez que agreguemos un producto al carrito, llamaremos a esta función para modificar el estado.

**3. Función para añadir productos al carrito**

const addToCart = (product) => {

const existingProduct = cartItems.find(item => item.id === product.id);

if (existingProduct) {

setCartItems(cartItems.map(item =>

item.id === product.id ? { ...item, quantity: item.quantity + 1 } : item

));

} else {

setCartItems([...cartItems, { ...product, quantity: 1 }]);

}

};

* **addToCart(product)**: Esta función se encarga de agregar un producto al carrito.
  + Recibe un parámetro product que contiene los datos del producto que el usuario selecciona.
* **Buscar si el producto ya está en el carrito** :

const existingProduct = cartItems.find(item => item.id === product.id);

* + Aquí usamos el método **find** para buscar si el producto ya existe en el carrito.
  + Si existingProduct no es undefined, significa que el producto ya está en el carrito.
* **Si el producto ya está en el carrito** :

if (existingProduct) {

setCartItems(cartItems.map(item =>

item.id === product.id ? { ...item, quantity: item.quantity + 1 } : item

));

}

* + Si el producto ya está en el carrito, usamos setCartItems para actualizar el carrito.
  + **map()**: Recorremos cada producto en el carrito. Si el id producto coincide con el que estamos agregando, aumentamos su cantidad.
  + **{ ...item, quantity: item.quantity + 1 }**: Esto crea una copia del objeto del producto y aumenta su cantidad en 1.
* **Si el producto no está en el carrito** :

} else {

setCartItems([...cartItems, { ...product, quantity: 1 }]);

}

* + Si el producto no está en el carrito, lo agregamos como un nuevo producto.
  + **setCartItems([...cartItems, ...])**: Esto agrega el nuevo producto al array de cartItems.
  + **{ ...product, quantity: 1 }**: Copiamos los datos del producto y añadimos una propiedad quantity con un valor inicial de 1.

**4. Estructura de la aplicación**

return (

<div>

<Header />

<main style={styles.main}>

<ProductList addToCart={addToCart} />

<Cart cartItems={cartItems} />

</main>

<Footer />

</div>

);

* **<Header />**: Renderiza el componente del encabezado de la página. Este componente no recibe accesorios porque solo muestra un logo y un buscador.
* **<main style={styles.main}>**: Aquí organizamos dos componentes principales:
  + **<ProductList addToCart={addToCart} />**: El componente ProductList muestra la lista de productos disponibles. Le pasamos la función addToCart como una **prop** para que, cuando el usuario haga clic en "Añadir al carrito", se llame esta función.
  + **<Cart cartItems={cartItems} />**: El componente Cart muestra los productos en el carrito. Le pasé cartItems como una **prop** , que es el estado del carrito actualizado.
* **<Footer />**: Renderiza el pie de página. Tampoco necesita accesorios.

**5. Estilos para el contenedor principal**

const styles = {

main: {

display: 'flex',

justifyContent: 'space-between',

padding: '20px',

}

};

* **styles.main**: Aquí usamos estilos CSS para organizar los componentes dentro del <main>.
  + **display: 'flex'**: Coloca ProductListy Cart uno al lado del otro.
  + **justifyContent: 'space-between'**: Separa los elementos de forma que haya espacio entre ellos.
  + **padding: '20px'**: Agregue espacio alrededor de los componentes.

**Resumen de la funcionalidad:**

1. **Estado del carrito** : useState controla los productos en el carrito.
2. **Añadir productos** : La función addToCart añade productos al carrito, verificando si ya existen para aumentar la cantidad.
3. **Renderización** : El componente principal App.js organiza y muestra el **encabezado** , la **lista de productos** , el **carrito de compras** y el **pie de página** . Los componentes hijos reciben las funciones y el estado necesario para funcionar correctamente.

Este patrón de pasar funciones y estado entre componentes usando **props** es común en React, y es lo que permite que la aplicación funcione de forma dinámica y reactiva, actualizando el carrito en tiempo real.