

ALBERT ALARCÓN MARTÍNEZ



# Documentación de React en Shopi



## Índice

1. Estructura del Proyecto
2. main.jsx - Punto de Entrada
3. index.jsx - Enrutamiento y Contextos
4. Home - Página Principal
5. Products - Página de Productos
6. Card - Componente de Tarjeta de Producto
7. ProductDetailPage - Página de Detalles
8. ProductFilters - Componente de Filtros
9. Navbar - Barra de Navegación
10. Hero - Carrusel de Productos Destacados
11. Cart - Carrito de Compras
12. Checkout - Página de Compra
13. SignIn - Página de Inicio de Sesión
14. MyOrders - Pedidos del Usuario
15. MyAccount - Perfil del Usuario
16. Contextos Globales: Carrito, Autenticación y Tema



## Estructura del Proyecto

El proyecto sigue una estructura modular con **separación de responsabilidades**.

```
src/
├── Components/      # Componentes reutilizables
│   ├── Card/        # Tarjeta de producto
│   ├── Cart/        # Carrito de compras
│   ├── Hero/        # Carrusel de productos
│   ├── Layout/      # Estructura base de la aplicación
│   ├── Navbar/      # Barra de navegación
│   └── ProductFilters/ # Filtros de productos
├── Context/         # Contextos globales (autenticación, carrito, tema)
├── Pages/           # Páginas principales
│   ├── Home/        # Página principal
│   ├── Products/    # Página de productos
│   ├── ProductDetailPage/ # Página de detalles de un producto
│   ├── Checkout/    # Página de finalización de compra
│   ├── SignIn/      # Página de inicio de sesión
│   ├── MyOrders/    # Página de pedidos del usuario
│   ├── MyAccount/   # Página de perfil del usuario
│   └── NotFound/    # Página 404
└── main.jsx         # Punto de entrada de la aplicación
```

## main.jsx - Punto de Entrada

Archivo donde se monta la aplicación en el DOM.

```
import React from 'react'
import ReactDOM from 'react-dom/client'
import App from './Pages/App'
import { ThemeProvider } from './Context/ThemeContext'
import './Pages/App/App.css'

ReactDOM.createRoot(document.getElementById('root')).render(
  <React.StrictMode>
    <ThemeProvider>
      <App />
    </ThemeProvider>
  </React.StrictMode>
)
```

- Encapsula la aplicación en **ThemeProvider** para manejar el modo oscuro.
- Se renderiza en **#root** definido en **index.html**.

## index.jsx - Enrutamiento y Contextos

Gestiona la navegación y el acceso a los contextos globales.

```
import { useRoutes, BrowserRouter } from 'react-router-dom'
import { CartProvider } from '../../../Context/CartContext'
import Home from '../Home'
import Products from '../Products'
import ProductDetailPage from '../ProductDetailPage'
import Checkout from '../Checkout'
import MyOrders from '../MyOrders'
import MyAccount from '../MyAccount'
import SignIn from '../SignIn'

const AppRoutes = () => {
  let routes = useRoutes([
    { path: '/', element: <Home /> },
    { path: '/products', element: <Products /> },
    { path: '/product/:id', element: <ProductDetailPage /> },
    { path: '/checkout', element: <Checkout /> },
    { path: '/my-orders', element: <MyOrders /> },
    { path: '/my-account', element: <MyAccount /> },
    { path: '/sign-in', element: <SignIn /> },
    { path: '/*', element: <NotFound /> }, // Página 404
  ])
  return routes
}
```

```
}

function App() {
  return (
    <BrowserRouter>
      <CartProvider>
        <AppRoutes />
      </CartProvider>
    </BrowserRouter>
  )
}
export default App
```

- **BrowserRouter**: Maneja la navegación entre páginas.
- **useRoutes()**: Define las rutas de la aplicación.
- **CartProvider**: Proporciona acceso global al carrito.

---

## Home - Página Principal

Carga los productos desde una API y muestra los destacados.

```
useEffect(() => {
  fetch('https://fakestoreapi.com/products')
    .then(res => res.json())
    .then(data => {
      setFeaturedProducts(data.slice(0, 3))
      setItems(data.slice(3, 9))
    })
}, [])
```

- **Obtiene los productos desde una API.**
- **Muestra 3 productos destacados** en un carrusel.

---

## Products - Página de Productos

Filtra y ordena productos.

```
const filteredProducts = useMemo(() => {
  return items.filter(item => item.price <= filters.maxPrice)
}, [items, filters])
```

- **Filtra productos por precio.**
- **Se usa useMemo()** para optimizar rendimiento.

## Card - Componente de Tarjeta

```
const handleAddToCart = () => {  
  addToCart({ ...product, quantity: 1 })  
}
```

- Añade productos al carrito con `addToCart()`.

---

## ProductDetailPage - Página de Detalles

```
const { id } = useParams()  
useEffect(() => {  
  fetch(`https://fakestoreapi.com/products/${id}`)  
    .then(res => res.json())  
    .then(setProduct)  
}, [id])
```

- Obtiene detalles de un producto según el `id` de la URL.

---

## ProductFilters - Componente de Filtros

```
const handleChange = (e) => {  
  setFilters({ ...filters, [e.target.name]: e.target.value })  
}
```

- Actualiza filtros en tiempo real.

---

## Cart - Carrito de Compras

```
const { cart, removeFromCart } = useCart()  
const cartTotal = cart.reduce((total, item) => total + item.price * item.quantity,  
0)
```

- Maneja el carrito con `useCart()`.
- Calcula el total de la compra.

---

## Checkout - Página de Compra

```
const handleSubmit = (e) => {  
  e.preventDefault()  
  clearCart()  
  navigate('/my-orders')  
}
```

- Vacía el carrito y redirige a "Mis Pedidos".

---

## SignIn - Página de Inicio de Sesión

```
const handleSubmit = (e) => {  
  e.preventDefault()  
  login({ email: formData.email })  
  navigate('/')  
}
```

- Autentica al usuario y redirige a la página principal.

---

## MyOrders - Pedidos del Usuario

Muestra los pedidos que ha realizado el usuario, obteniéndolos desde **localStorage**.

### Importaciones

```
import { useState } from 'react'  
import { Link } from 'react-router-dom'  
import { motion } from 'framer-motion'  
import Layout from '../Components/Layout'  
import { useAuth } from '../Context/AuthContext'
```

- **useState**: Almacena los pedidos.
- **Link**: Permite redireccionar a la página de productos.
- **motion**: Aplica animaciones en los pedidos.
- **useAuth**: Obtiene la información del usuario autenticado.
- **Layout**: Proporciona la estructura de la página.

---

### Estado del Componente

```
const { user } = useAuth()  
  
const [orders, setOrders] = useState(() => {
```

```
const savedOrders = localStorage.getItem('orders')
return savedOrders ? JSON.parse(savedOrders) : []
})
```

- Obtiene los pedidos almacenados en `localStorage`.
- Si no hay pedidos guardados, inicializa con un array vacío.

## Función para Formatear Fechas

```
const formatDate = (dateString) => {
  return new Date(dateString).toLocaleDateString('es-ES', {
    year: 'numeric',
    month: 'long',
    day: 'numeric'
  })
}
```

- Convierte una fecha ISO en una fecha más legible en español.
- Ejemplo: "2024-01-15T14:30:00Z" → "15 de enero de 2024".

## Renderizado de Pedidos

```
{orders.length === 0 ? (
  <div className="text-center py-12">
    <p>No tienes pedidos realizados</p>
    <Link to="/products">Ir a comprar</Link>
  </div>
) : (
  <div className="space-y-6">
    {orders.map((order) => (
      <motion.div key={order.id} className="border p-6">
        <h3>Pedido #{order.id}</h3>
        <p>{formatDate(order.date)}</p>
        <p>Total: ${order.total.toFixed(2)}</p>
        <p>Estado: {order.status}</p>
      </motion.div>
    ))}
  </div>
)}
```

- Si no hay pedidos, muestra un mensaje y un enlace a la tienda.
- Si hay pedidos, los lista con su ID, fecha, total y estado.

## MyAccount - Perfil del Usuario

Permite al usuario ver y editar sus datos personales.

## Importaciones

```
import { useState } from 'react'
import { motion } from 'framer-motion'
import { useAuth } from '../Context/AuthContext'
import Layout from '../Components/Layout'
```

- **useState**: Maneja el estado del formulario.
- **motion**: Aplica animaciones en los botones.
- **useAuth**: Obtiene y actualiza la información del usuario.
- **Layout**: Proporciona la estructura base de la página.

---

## Estados del Componente

```
const { user, login } = useAuth()

const [isEditing, setIsEditing] = useState(false)
const [formData, setFormData] = useState({
  name: user?.name || '',
  email: user?.email || '',
  address: user?.address || '',
  phone: user?.phone || ''
})
```

- **isEditing**: Controla si el usuario está editando su información.
- **formData**: Contiene los datos del usuario, inicializados desde `useAuth()`.

---

## Guardar Cambios

```
const handleSubmit = (e) => {
  e.preventDefault()
  login({ ...user, ...formData })
  setIsEditing(false)
}
```

- **Simula la actualización del perfil** llamando a `login()` con los nuevos datos.
- **Cierra el modo edición** (`isEditing = false`).

---

## Botón de Edición

```
<motion.button onClick={() => setIsEditing(!isEditing)}>
  {isEditing ? 'Cancelar' : 'Editar'}
</motion.button>
```

- Cambia entre modo edición y vista normal.

## Formulario Editable

```
<form onSubmit={handleSubmit}>
  <label>Nombre</label>
  {isEditing ? (
    <input type="text" value={formData.name} onChange={(e) =>
setFormData({...formData, name: e.target.value})} />
  ) : (
    <p>{formData.name}</p>
  )}
  <motion.button type="submit">Guardar Cambios</motion.button>
</form>
```

- Si **isEditing === true**, muestra inputs editables.
- Si **isEditing === false**, muestra los datos como texto.

## Contextos Globales: Carrito, Autenticación y Tema

### 1 CartContext - Carrito de Compras

```
import { createContext, useContext, useState, useEffect } from 'react'

const CartContext = createContext()

export function CartProvider({ children }) {
  const [cart, setCart] = useState(() => {
    const savedCart = localStorage.getItem('cart')
    return savedCart ? JSON.parse(savedCart) : []
  })

  useEffect(() => {
    localStorage.setItem('cart', JSON.stringify(cart))
  }, [cart])

  const addToCart = (product) => {
    setCart([...cart, product])
  }

  const removeFromCart = (productId) => {
    setCart(cart.filter(item => item.id !== productId))
  }
}
```



```

    }

    return (
      <CartContext.Provider value={{ cart, addToCart, removeFromCart }}>
        {children}
      </CartContext.Provider>
    )
  }
}

export const useCart = () => useContext(CartContext)

```

- **Almacena el carrito en `localStorage`.**
- **Funciones:** `addToCart()`, `removeFromCart()`, `useCart()`.

## 2 AuthContext - Autenticación

```

import { createContext, useContext, useState } from 'react'

const AuthContext = createContext()

export function AuthProvider({ children }) {
  const [user, setUser] = useState(() => {
    return JSON.parse(localStorage.getItem('user')) || null
  })

  const login = (userData) => {
    setUser(userData)
    localStorage.setItem('user', JSON.stringify(userData))
  }

  const logout = () => {
    setUser(null)
    localStorage.removeItem('user')
  }

  return (
    <AuthContext.Provider value={{ user, login, logout }}>
      {children}
    </AuthContext.Provider>
  )
}

export const useAuth = () => useContext(AuthContext)

```

- **Maneja el estado del usuario y lo almacena en `localStorage`.**
- **Funciones:** `login()`, `logout()`, `useAuth()`.

## 3 ThemeContext - Tema (Claro/Oscuro)

```
import { createContext, useContext, useEffect, useState } from 'react'

const ThemeContext = createContext()

export function ThemeProvider({ children }) {
  const [theme, setTheme] = useState(() => {
    return localStorage.getItem('theme') || 'light'
  })

  useEffect(() => {
    document.documentElement.classList.remove('light', 'dark')
    document.documentElement.classList.add(theme)
    localStorage.setItem('theme', theme)
  }, [theme])

  const toggleTheme = (newTheme) => {
    setTheme(newTheme)
  }




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

export const useTheme = () => useContext(ThemeContext)
```

- **Cambia entre temas claro y oscuro.**
- **Funciones:** `toggleTheme()`, `useTheme()`.

---

## Resumen

-  **CartContext:** Maneja el **carrito de compras**.
  -  **AuthContext:** Gestiona la **autenticación de usuarios**.
  -  **ThemeContext:** Controla el **modo oscuro/claro**.
-