

Documentación del Proyecto Shopi

ALBERT ALARCÓN MARTÍNEZ

Índice del Proyecto Shopi

Componentes y Páginas

- Descripción General
- Estructura del Proyecto
- main.jsx - Punto de entrada de la aplicación
- index.jsx - Componente principal de la aplicación
- Home/index.jsx - Página principal
- Layout/index.jsx - Estructura base de la aplicación
- Products/index.jsx - Página de productos
- Card/index.jsx - Componente de tarjeta de producto
- ProductDetailPage/index.jsx - Página de detalles de un producto
- ProductFilters/index.jsx - Filtros para productos
- Navbar/index.jsx - Barra de navegación
- Hero/index.jsx - Carrusel de productos destacados
- Cart/index.jsx - Carrito de compras
- CartContext.jsx - Contexto global del carrito de compras
- Checkout/index.jsx - Página de finalización de compra
- Modo Oscuro
- SignIn/index.jsx - Página de inicio de sesión
- MyOrders/index.jsx - Página de pedidos del usuario
- MyAccount/index.jsx - Página de perfil del usuario

Descripción General

Shopi es una aplicación de comercio electrónico desarrollada en React que permite a los usuarios navegar productos, gestionar un carrito de compras y realizar pedidos.

Tecnologías Principales

- React
- React Router
- Framer Motion
- Tailwind CSS
- Context API

Estructura del Proyecto

```
src/  
  └── Components/
```

```

  └── Card/
    └── index.jsx      # Tarjetas de productos

  └── Cart/
    └── index.jsx      # Carrito de compras flotante

  └── Footer/
    └── index.jsx      # Pie de página

  └── Hero/
    └── index.jsx      # Banner principal y productos destacados

  └── Layout/
    └── index.jsx      # Estructura base de la aplicación

  └── Navbar/
    └── index.jsx      # Barra de navegación

  └── ProductFilters/
    └── index.jsx      # Filtros de productos (categorías, precio, etc.)

  Context/
  ├── AuthContext.jsx
  ├── CartContext.jsx
  ├── ThemeContext.jsx
  └── index.jsx      # Contexto de autenticación
                      # Contexto del carrito
                      # Contexto del tema (dark/light)
                      # Contexto principal

  Pages/
  ├── App/
    ├── index.jsx      # Componente principal y rutas
    └── App.css         # Estilos globales

  ├── Checkout/
    └── index.jsx      # Página de finalización de compra

  ├── Home/
    └── index.jsx      # Página principal

  ├── MyAccount/
    └── index.jsx      # Página de perfil de usuario

  ├── MyOrder/
    └── index.jsx      # Detalle de una orden

  ├── MyOrders/
    └── index.jsx      # Lista de órdenes

  ├── NotFound/
    └── index.jsx      # Página 404

  ├── ProductDetailPage/
    └── index.jsx      # Detalle de producto

  ├── Products/
    └── index.jsx      # Lista de productos

  └── SignIn/
    └── index.jsx      # Página de inicio de sesión

  main.jsx          # Punto de entrada de la aplicación

```

main.jsx - Punto de entrada de la aplicación

Este archivo es el ****punto de entrada**** donde se monta la aplicación en el DOM.

```

## 📂 Importaciones
```jsx
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'
```

- **React y ReactDOM**: Manejan el renderizado de la aplicación en el navegador.
- **App**: Componente principal que contiene la estructura de la aplicación.
- **ThemeProvider**: Contexto que maneja el modo oscuro/claro en la app.
- **App.css**: Estilos globales de la aplicación.

## ⌚ Renderizado de la aplicación en el DOM

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

- **ReactDOM.createRoot(document.getElementById('root'))**:
  - Selecciona el elemento con `id="root"` en el `index.html` donde se insertará la app.
  - Utiliza **React 18** con `createRoot()` para un mejor manejo de la renderización concurrente.
- **<React.StrictMode>**:
  - Modo estricto de React, que ayuda a identificar problemas en el desarrollo.
- **<ThemeProvider>**:
  - **Encapsula toda la aplicación** (`<App />`) dentro del **contexto de temas**.
  - **Permite que cualquier componente acceda y modifique el tema**.
- **<App />**:
  - Es el **componente principal**, que contiene todas las páginas y funcionalidades de la aplicación.

## ⌚ ¿Dónde se carga la aplicación?

Este código **inyecta la app dentro del elemento root del index.html**, permitiendo que React tome el control de la página.

Ejemplo del `index.html`:

```
<body>
 <div id="root"></div>
</body>
```

- React renderiza toda la aplicación dentro de `<div id="root">`.

```
⚒ `index.jsx` - Componente principal de la aplicación
```

El archivo `index.jsx` actúa como el \*\*punto de entrada principal\*\* de la aplicación. Define la estructura de enrutamiento utilizando `react-router-dom` y envuelve la aplicación en distintos \*\*context providers\*\* para manejar autenticación, carrito de compras, y más.

---

```
📁 **Importaciones**
```jsx  
import { useRoutes, BrowserRouter } from 'react-router-dom';  
  
import { ShoppingCartProvider } from '../../../../../Context';  
import { AuthProvider } from '../../../../../Context/AuthContext';  
import { CartProvider } from '../../../../../Context/CartContext';  
  
import Home from '../Home';  
import Products from '../Products';  
import ProductDetailPage from '../ProductDetailPage';  
import MyAccount from '../MyAccount';  
import MyOrder from '../MyOrder';  
import MyOrders from '../MyOrders';  
import NotFound from '../NotFound';  
import SignIn from '../SignIn';  
import Checkout from '../Checkout';  
import './App.css';
```

◊ Explicación:

- **React Router:** Se importan `BrowserRouter` y `useRoutes` de `react-router-dom` para manejar la navegación entre páginas.
- **Contextos:** Se importan `ShoppingCartProvider`, `AuthProvider` y `CartProvider` para compartir estados globales en la aplicación.
- **Componentes de página:** Se importan distintos componentes que representan vistas de la aplicación (`Home`, `Products`, `Checkout`, etc.).
- **Estilos:** Se importa el archivo `App.css` para los estilos globales del componente.

⚡ Manejo de Rutas (AppRoutes Component)

```
const AppRoutes = () => {  
  let routes = useRoutes([  
    { path: '/', element: <Home /> },
```

```

    { path: '/products', element: <Products /> },
    { path: '/product/:id', element: <ProductDetailPage /> },
    { path: '/my-account', element: <MyAccount /> },
    { path: '/my-order', element: <MyOrder /> },
    { path: '/my-orders', element: <MyOrders /> },
    { path: '/sign-in', element: <SignIn /> },
    { path: '/checkout', element: <Checkout /> },
    { path: '*', element: <NotFound /> }, // Página 404
  ]);

  return routes;
};


```

❖ Explicación:

- **useRoutes()**: Se utiliza para definir las rutas de la aplicación de manera declarativa.
- **Rutas principales:**
 - `/` → Página de inicio (`Home`).
 - `/products` → Página con lista de productos.
 - `/product/:id` → Página de detalles de un producto (el `:id` es un parámetro dinámico).
 - `/my-account` → Página de cuenta del usuario.
 - `/my-order` y `/my-orders` → Páginas para ver órdenes de compra.
 - `/sign-in` → Página de inicio de sesión.
 - `/checkout` → Página de pago.
 - `/*` → Ruta por defecto que muestra una página **404** si la URL no coincide con ninguna ruta definida.

❖ Componente App

```

function App() {
  return (
    <BrowserRouter>
      <AuthProvider>
        <CartProvider>
          <ShoppingCartProvider>
            <AppRoutes />
          </ShoppingCartProvider>
        </CartProvider>
      </AuthProvider>
    </BrowserRouter>
  );
}

```

❖ Explicación:

1. **BrowserRouter** (de `react-router-dom`):

- Habilita el enrutamiento de la aplicación.
- Permite la navegación entre páginas sin recargar la página completa.

2. Context Providers (manejo de estados globales):

- <AuthProvider>: Maneja la autenticación de usuarios.
- <CartProvider>: Administra el estado del carrito de compras.
- <ShoppingCartProvider>: Otro contexto de carrito (posiblemente una versión legacy).
- Los contextos están **anidados** dentro del **BrowserRouter**, lo que permite que cualquier componente dentro de la app acceda a los estados globales.

💡 Ejemplo: Uso de AuthProvider

Si dentro de algún componente de la aplicación se necesita el estado de autenticación, se podría acceder con:

```
import { useContext } from 'react';
import { AuthContext } from '../..../Context/AuthContext';

const Profile = () => {
  const { user, login, logout } = useContext(AuthContext);

  return (
    <div>
      {user ? (
        <>
          <p>Bienvenido, {user.name}</p>
          <button onClick={logout}>Cerrar sesión</button>
        </>
      ) : (
        <button onClick={() => login('usuarioEjemplo')}>Iniciar sesión</button>
      )}
    </div>
  );
};
```

◊ Explicación:

- Se usa **useContext(AuthContext)** para acceder al estado de autenticación.
- Se puede llamar a **login()** para iniciar sesión y a **logout()** para cerrar sesión.
- Se muestra el nombre del usuario si está autenticado.

💡 Ejemplo: Uso de CartProvider

Para obtener y modificar el estado del carrito de compras:

```
import { useContext } from 'react';
import { CartContext } from '../..../Context/CartContext';
```

```
const CartButton = ({ product }) => {
  const { cart, addToCart, removeFromCart } = useContext(CartContext);

  const isInCart = cart.some(item => item.id === product.id);

  return (
    <button onClick={() => (isInCart ? removeFromCart(product.id) :
    addToCart(product))}>
      {isInCart ? 'Eliminar del carrito' : 'Añadir al carrito'}
    </button>
  );
};
```

❖ Explicación:

- Se usa `useContext(CartContext)` para acceder al estado del carrito.
 - Se verifica si el producto ya está en el carrito con `some()`.
 - Se usa `addToCart()` y `removeFromCart()` para gestionar los productos en el carrito.
-

```
# Home/index.jsx - Página principal

## 📁 Importación de dependencias
```jsx
import { useEffect, useState } from 'react'
import { motion } from 'framer-motion'
import Layout from '../../../../../Components/Layout'
import Hero from '../../../../../Components/Hero'
import Card from '../../../../../Components/Card'
```

- `useEffect`, `useState`: Manejo de estados y efectos en React.
- `motion` de `framer-motion`: Animaciones en los elementos de la UI.
- Componentes reutilizables: `Layout`, `Hero`, `Card`.

## 🛒 Estados

```
const [items, setItems] = useState(null) // Productos normales
const [featuredProducts, setFeaturedProducts] = useState([]) // Productos
destacados
const [error, setError] = useState(null) // Estado de error
```

- `items`: Productos generales mostrados en la grid.
- `featuredProducts`: Productos destacados en la sección `Hero`.
- `error`: Manejo de errores en la carga de datos.

## 🔗 Carga de productos desde API

```

useEffect(() => {
 fetch('https://fakestoreapi.com/products')
 .then(response => {
 if (!response.ok) throw new Error('Network response was not ok')
 return response.json()
 })
 .then(data => {
 const shuffled = [...data].sort(() => 0.5 - Math.random())
 setFeaturedProducts(shuffled.slice(0, 3)) // 3 productos destacados
 setItems(shuffled.slice(3, 9)) // 6 productos en la grid
 })
 .catch(error => {
 console.error('Error fetching data:', error)
 setError(error.message)
 })
}, [])

```

- **Petición a la API** <https://fakestoreapi.com/products>.
- **Validación de respuesta**: Si falla, se lanza un error.
- **Aleatorización de productos**: Se mezclan para mostrar diferentes en cada carga.
- **Gestión de errores**: Captura y almacenamiento en `error`.

## ⑦ Manejo de error en la UI

```
if (error) return <div>Error: {error}</div>
```

- Si la API falla, se muestra un mensaje en la interfaz.

## ☒ Renderizado de la página

```

return (
 <Layout>
 {/* Sección Hero con productos destacados */}
 <Hero featuredProducts={featuredProducts} />

 {/* Sección de productos principales */}
 <section className="container px-4 py-16 mx-auto">
 {/* Título con animación de entrada */}
 <motion.h2
 initial={{ opacity: 0, y: 20 }}
 whileInView={{ opacity: 1, y: 0 }}
 viewport={{ once: true }}
 className="mb-12 text-3xl font-bold text-center">
 >
 Productos Destacados
 </motion.h2>

```

```

 {/* Grid de productos */}
 <div className='grid gap-6 grid-cols-1 sm:grid-cols-2 lg:grid-cols-3 w-full
max-w-[1400px]>
 {items?.map((item, index) => (
 <Card
 key={item.id}
 data={{
 ...item,
 images: item.image || '',
 category: {
 name: item.category
 }
 }}
 index={index}
 />
)))
 </div>
 </section>
</Layout>
)

```

- **Hero:** Muestra los productos destacados.
- **Título con animación** (`motion.h2` de `framer-motion`).
- **Grid de productos:**
  - Mapea los productos y los pasa a `Card`.
  - Formato responsive (`grid-cols-1`, `sm:grid-cols-2`, `lg:grid-cols-3`).

## ← END Exportación del componente

```
export default Home
```

- Se exporta para ser utilizado en el enrutador principal.

```

```markdown
# Layout/index.jsx - Estructura base de la aplicación

## 📂 Importaciones
```jsx
import { useContext } from 'react'
import { ShoppingCartContext } from '../../Context'
import Navbar from '../Navbar'
import Footer from '../Footer'
import Cart from '../Cart'

```

- **useContext:** Hook de React para acceder al contexto global.
- **ShoppingCartContext:** Contexto del carrito de compras.

- **Componentes reutilizables:**

- **Navbar**: Barra de navegación fija.
- **Footer**: Pie de página.
- **Cart**: Componente del carrito de compras.

## 🔧 Componente Layout

```
function Layout({ children }) {
 const context = useContext(ShoppingCartContext)
```

- Recibe **children**, que representa el contenido dinámico de cada página.
- Usa **useContext(ShoppingCartContext)** para obtener el estado del carrito.

## 🌐 Estructura y estilos

```
return (
 <div className="flex flex-col min-h-screen bg-white dark:bg-gray-900 transition-
 colors duration-200">
 <Navbar />

 <div className="flex-1 pt-[68px]">
 {children}
 </div>

 <Cart />
 <Footer />
 </div>
)
```

- **flex flex-col min-h-screen**:
  - Estructura en columna (**flex-col**).
  - Mínimo alto de **100vh (min-h-screen)**.
- **Tema claro/oscuro**:
  - **bg-white** para modo claro.
  - **dark:bg-gray-900** para modo oscuro.
  - **transition-colors duration-200** para cambio suave.
- **flex-1 pt-[68px]**:
  - **flex-1** permite que el contenido ocupe el espacio restante.
  - **pt-[68px]** evita que la **Navbar** fija lo cubra.
- **Cart y Footer**:
  - **Cart** se muestra/oculta según el contexto.
  - **Footer** permanece visible al final.

## ⬅ END Exportación del componente

```
export default Layout
```

- Se exporta para envolver todas las páginas dentro de la aplicación.

```
```markdown
# Products/index.jsx - Página de productos

## 📁 Importaciones
```jsx
import { useState, useEffect, useMemo } from 'react'
import Layout from '../../Components/Layout'
import Card from '../../Components/Card'
import ProductFilters from '../../Components/ProductFilters'
import { motion } from 'framer-motion'
```

- **useState**: Manejo de estados locales.
- **useEffect**: Carga de productos desde la API.
- **useMemo**: Optimización de filtrado y ordenación.
- **Componentes reutilizables**:
  - **Layout**: Estructura base de la página.
  - **Card**: Tarjeta individual para mostrar cada producto.
  - **ProductFilters**: Componente de filtros.
- **motion de framer-motion**: Animaciones para botones de paginación.

## 🛒 Estados del componente

```
const [items, setItems] = useState(null)
const [error, setError] = useState(null)
const [currentPage, setCurrentPage] = useState(1)
const [filters, setFilters] = useState({
 category: '',
 maxPrice: 1000,
 sort: 'default'
})
```

- **items**: Productos obtenidos de la API.
- **error**: Manejo de errores en la carga.
- **currentPage**: Página actual para la paginación.
- **filters**: Objeto con criterios de filtrado.

## 🔗 Carga de productos desde API

```
useEffect(() => {
 fetch('https://fakestoreapi.com/products')
 .then(response => {
 if (!response.ok) throw new Error('Network response was not ok')
 return response.json()
 })
 .then(data => setItems(data))
 .catch(error => {
 console.error('Error fetching data:', error)
 setError(error.message)
 })
}, [])
```

- Obtiene productos desde <https://fakestoreapi.com/products>.
- Maneja errores si la petición falla.

## 🔍 Filtrado y ordenación con useMemo

```
const filteredProducts = useMemo(() => {
 if (!items) return []

 return items
 .filter(item => {
 const matchesCategory = !filters.category || item.category ===
filters.category
 const matchesPrice = item.price <= filters.maxPrice
 return matchesCategory && matchesPrice
 })
 .sort((a, b) => {
 switch (filters.sort) {
 case 'price-asc': return a.price - b.price
 case 'price-desc': return b.price - a.price
 case 'name-asc': return a.title.localeCompare(b.title)
 case 'name-desc': return b.title.localeCompare(a.title)
 default: return 0
 }
 })
}, [items, filters])
```

- Filtra productos por **categoría** y **precio máximo**.
- Ordena por **precio** o **nombre** según **filters.sort**.
- Usa **useMemo** para mejorar el rendimiento.

## 12 34 Paginación

```
const productsPerPage = 8
const indexOfLastProduct = currentPage * productsPerPage
```

```
const indexOfFirstProduct = indexOfLastProduct - productsPerPage
const currentProducts = filteredProducts.slice(indexOfFirstProduct,
indexOfLastProduct)
const totalPages = Math.ceil(filteredProducts.length / productsPerPage)
```

- Define `productsPerPage = 8`.
- Calcula índices de productos a mostrar en la página actual.

```
const paginate = (pageNumber) => {
 setCurrentPage(pageNumber)
 window.scrollTo({ top: 0, behavior: 'smooth' })
}
```

- Cambia de página y desplaza la vista hacia arriba.

## ⑦ Renderizado de error

```
if (error) {
 return (
 <Layout>
 <div className="flex items-center justify-center">
 <p className="text-red-500">Error: {error}</p>
 </div>
 </Layout>
)
}
```

- Muestra mensaje de error si la carga falla.

## 🖼 Renderizado principal

```
return (
 <Layout>
 <div className="flex flex-col min-h-screen pb-24">
 <ProductFilters
 filters={filters}
 setFilters={setFilters}
 categories={[...new Set(items?.map(item => item.category) || [])]}
 priceRange={{
 min: 0,
 max: Math.ceil(Math.max(...(items?.map(item => item.price) || [0])))
 }}
 sortOptions={[
 { value: 'default', label: 'Por defecto' },
 { value: 'price-asc', label: 'Precio: Menor a Mayor' },
 { value: 'price-desc', label: 'Precio: Mayor a Menor' },
 { value: 'name-asc', label: 'Nombre: A-Z' },
]}
 </ProductFilters>
 </div>
 </Layout>
)
```

```

 { value: 'name-desc', label: 'Nombre: Z-A' }
]}
/>

<div className="grid gap-4 grid-cols-1 sm:grid-cols-2 lg:grid-cols-4 p-4">
 {currentProducts.map((item) => (
 <Card
 key={item.id}
 data={{
 ...item,
 images: item.image,
 category: {
 name: item.category
 }
 }}
 />
)))
</div>

```

- **ProductFilters**: Controla filtros de categoría, precio y ordenación.
- **Grid de productos**:
  - **Responsive** (grid-cols-1 sm:grid-cols-2 lg:grid-cols-4).
  - **Iteración sobre currentProducts**.

## Paginación con animaciones

```

{totalPages > 1 && (
 <div className="flex justify-center gap-2 mt-8 mb-12">
 <motion.button
 whileHover={{ scale: 1.05 }}
 whileTap={{ scale: 0.95 }}
 onClick={() => currentPage > 1 && paginate(currentPage - 1)}
 className={`${px-4 py-2 rounded-lg ${currentPage === 1
 ? 'bg-gray-300 cursor-not-allowed'
 : 'bg-primary text-white hover:bg-secondary'}`}
 >
 Anterior
 </motion.button>

 <div className="flex gap-2">
 {[...Array(totalPages)].map(_, index) => (
 <motion.button
 key={index + 1}
 whileHover={{ scale: 1.1 }}
 whileTap={{ scale: 0.95 }}
 onClick={() => paginate(index + 1)}
 className={`${w-10 h-10 rounded-lg ${currentPage === index + 1
 ? 'bg-gray-300 cursor-not-allowed'
 : 'bg-primary text-white hover:bg-secondary'}`}
 >
 ${index + 1}
 </motion.button>
)}
 </div>
 </div>
)

```

```

 ? 'bg-primary text-white'
 : 'bg-gray-300 hover:bg-gray-300'
 }`}
 >
 {index + 1}
 </motion.button>
)
)
</div>

<motion.button
 whileHover={{ scale: 1.05 }}
 whileTap={{ scale: 0.95 }}
 onClick={() => currentPage < totalPages && paginate(currentPage + 1)}
 className={`px-4 py-2 rounded-lg ${currentPage === totalPages
 ? 'bg-gray-200 cursor-not-allowed'
 : 'bg-primary text-white hover:bg-secondary'}`}
>
 Siguiente
</motion.button>
</div>
)
}

```

- **Botón "Anterior"**: Se desactiva en la primera página.
- **Botones de número de página**:
  - Destaca la página actual (`bg-primary text-white`).
  - Animaciones con `motion.button`.
- **Botón "Siguiente"**: Se desactiva en la última página.

## Exportación

```
export default Products
```

- Permite importar `Products` en el enrutador.

```

```markdown
# Card/index.jsx - Componente de tarjeta de producto

## 📂 Importaciones
```jsx
import { useNavigate } from 'react-router-dom'
import { motion } from 'framer-motion'
import { PlusIcon } from '@heroicons/react/24/outline'
import { useCart } from '../../../../../Context/CartContext'

```

- **useNavigate**: Navegación entre páginas (`react-router-dom`).
- **motion de framer-motion**: Animaciones en la tarjeta y botones.
- **PlusIcon de Heroicons**: Icono para el botón "Añadir al carrito".
- **useCart**: Hook del contexto del carrito.

## Props del componente

```
function Card({ data: product, index }) {
```

- **product**: Datos del producto a mostrar.
- **index**: Índice en la lista (usado en animaciones).

## Navegación y carrito

```
const navigate = useNavigate()
const { addToCart, toggleCart } = useCart()
```

- **navigate()**: Redirige a la página de detalles del producto.
- **addToCart()**: Agrega un producto al carrito.
- **toggleCart()**: Muestra/oculta el carrito.

## Manejo de eventos

```
const handleNavigate = () => {
 navigate(`/product/${product.id}`)
}

const handleAddToCart = (event) => {
 event.stopPropagation()

 addToCart({
 id: product.id,
 title: product.title,
 price: product.price,
 image: product.images || product.image,
 category: typeof product.category === 'string' ? product.category :
 product.category.name,
 description: product.description,
 quantity: 1
 })

 toggleCart()
}
```

- **handleNavigate()**: Redirige al usuario al detalle del producto.

- **handleAddToCart(event):**
  - **Detiene la propagación** (`stopPropagation()`) para evitar activar `handleNavigate()`.
  - **Agrega el producto al carrito** con `addToCart()`.
  - **Abre el carrito** con `toggleCart()`.

## 🖼️ Renderizado de la tarjeta

```
return (
 <motion.div
 initial={{ opacity: 0, y: 20 }}
 animate={{ opacity: 1, y: 0 }}
 whileHover={{ y: -5 }}
 transition={{ delay: index * 0.1 }}
 className="bg-white dark:bg-gray-800 rounded-2xl shadow-lg overflow-hidden"
 >
```

- **Animaciones:**
  - Aparece con `opacity: 0 → 1` y `y: 20 → 0`.
  - Efecto flotante en `hover` (`y: -5`).
  - Retraso dinámico basado en `index`.

## ❖ Imagen del producto

```
<div
 onClick={handleNavigate}
 className="relative cursor-pointer"
>
 <img
 src={product.images || product.image}
 alt={product.title}
 className="w-full h-48 object-contain bg-white dark:bg-gray-700 p-4"
 />
</div>
```

- **Click en la imagen** → Redirige a detalles del producto.
- **object-contain**: Evita deformaciones en la imagen.

## ❖ Información del producto

```
<div className="flex flex-col flex-1 p-4">
 <div className="flex flex-col flex-1">
 <h3 className="text-lg font-semibold text-gray-800 dark:text-white line-
 clamp-2 mb-2">
 {product.title}
 </h3>
 <span className="px-2 py-1 text-sm rounded-lg bg-primary/10 text-primary
 dark:bg-primary/20 self-start">
```

```

 {typeof product.category === 'string' ? product.category :
product.category.name}

</div>

```

- **h3**: Muestra el nombre del producto con un máximo de 2 líneas (`line-clamp-2`).

- **Categoría:**

- `bg-primary/10 text-primary`: Estilo para fondo y texto.
- `self-start`: Ajuste automático en el layout.

## ❖ Precio y botón "Añadir al carrito"

```

<div className="flex items-center justify-between mt-4">

 ${product.price}

 <motion.button
 whileHover={{ scale: 1.1 }}
 whileTap={{ scale: 0.95 }}
 onClick={handleAddToCart}
 className="flex items-center gap-2 p-2 text-white transition-colors
duration-300 rounded-xl bg-primary hover:bg-secondary"
 >
 <PlusIcon className="h-6 w-6" />
 Añadir
 </motion.button>
</div>
</div>

```

- **Precio** (`text-2xl font-bold text-primary`).

- **Botón "Añadir":**

- **Animaciones `motion.button`:**
  - `whileHover={{ scale: 1.1 }}` → Efecto al pasar el mouse.
  - `whileTap={{ scale: 0.95 }}` → Efecto al hacer clic.
- **Estilos:**
  - `bg-primary hover:bg-secondary`: Color dinámico.
  - `rounded-xl p-2 flex items-center gap-2`: Diseño adaptado.

## ◀ END Exportación del componente

```
export default Card
```

- Permite importar `Card` en otros módulos.

```
```markdown
# ProductDetailPage/index.jsx - Página de detalles de un producto

## 📂 Importaciones
```jsx
import { useState, useEffect } from 'react'
import { useParams, useNavigate } from 'react-router-dom'
import { motion } from 'framer-motion'
import { useCart } from '../../Context/CartContext'
import { PlusIcon, MinusIcon, ArrowLeftIcon, ShoppingCartIcon } from
'@heroicons/react/24/solid'
import Layout from '../../Components/Layout'
```

- **useParams**: Obtiene el ID del producto desde la URL.
- **useNavigate**: Permite regresar a la página anterior.
- **motion de framer-motion**: Animaciones para botones y elementos de la UI.
- **useCart**: Hook del contexto del carrito (**CartContext**).
- **Iconos de Heroicons**: **PlusIcon**, **MinusIcon**, **ArrowLeftIcon**, **ShoppingCartIcon**.
- **Layout**: Estructura base de la página.

---

## STATES Estados del componente

```
const { id } = useParams()
const navigate = useNavigate()
const { addToCart } = useCart()

const [product, setProduct] = useState(null)
const [quantity, setQuantity] = useState(1)
const [loading, setLoading] = useState(true)
const [error, setError] = useState(null)
```

- **id**: Identificador del producto.
- **navigate**: Función para volver a la página anterior.
- **addToCart**: Agrega el producto al carrito.
- **Estados**:
  - **product**: Datos del producto.
  - **quantity**: Cantidad seleccionada.
  - **loading**: Indica si la API aún está cargando.
  - **error**: Almacena mensajes de error.

---

## UPLOAD Carga del producto

```
useEffect(() => {
 fetch(`https://fakestoreapi.com/products/${id}`)
 .then(response => {
 if (!response.ok) throw new Error('Producto no encontrado')
 return response.json()
 })
 .then(data => {
 setProduct(data)
 setLoading(false)
 })
 .catch(error => {
 setError(error.message)
 setLoading(false)
 })
}, [id])
```

- **Petición a la API** con `id` dinámico.
- **Manejo de errores** en caso de fallo en la carga.

## ⌚ Manejo de cantidad

```
const handleQuantityChange = (value) => {
 const newQuantity = Math.max(1, quantity + value)
 setQuantity(newQuantity)
}
```

- No permite valores menores a `1`.

## 🛒 Añadir al carrito

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

- Agrega el producto con la cantidad seleccionada.

## 🖼 Renderizado durante carga

```
if (loading) {
 return (
```

```

<Layout>
 <div className="min-h-screen bg-gray-100 dark:bg-gray-900 py-12 px-4 sm:px-6
lg:px-8">
 <div className="max-w-7xl mx-auto flex justify-center items-center">
 <div className="animate-spin rounded-full h-32 w-32 border-b-2 border-
primary"></div>
 </div>
 </div>
</Layout>
)
}

```

- **Indicador de carga (animate-spin).**

## ✖ Renderizado en caso de error

```

if (error) {
 return (
 <Layout>
 <div className="min-h-screen bg-gray-100 dark:bg-gray-900 py-12 px-4 sm:px-6
lg:px-8">
 <div className="max-w-7xl mx-auto text-center text-red-500">
 Error: {error}
 </div>
 </div>
 </Layout>
)
}

```

- **Muestra mensaje de error** si el producto no se encuentra.

## 📋 Renderizado principal

```

return (
 <Layout>
 <div className="min-h-screen bg-gray-100 dark:bg-gray-900 py-12 px-4 sm:px-6
lg:px-8">
 <div className="max-w-7xl mx-auto flex flex-col md:flex-row gap-8">

```

- **Disposición flex (md:flex-row).**

## ⬅ Botón "Volver atrás"

```

<motion.button
 whileHover={{ scale: 1.05 }}>
```

```
whileTap={{ scale: 0.95 }}
onClick={() => navigate(-1)}
className="absolute top-24 left-8 p-2 text-gray-600 dark:text-gray-400
hover:text-gray-900 dark:hover:text-white"
>
 <ArrowLeftIcon className="w-6 h-6" />
</motion.button>
```

- **Regresa a la página anterior.**
- **Efectos whileHover y whileTap.**

---

## 🔗 Información del producto

```
<h1 className="text-3xl font-bold text-gray-900 dark:text-white mb-4">
 {product.title}
</h1>
<span className="px-3 py-1 text-sm rounded-full bg-primary/10 text-primary
dark:bg-primary/20">
 {product.category}

<p className="mt-6 text-gray-600 dark:text-gray-300">
 {product.description}
</p>
<p className="mt-6 text-3xl font-bold text-primary dark:text-secondary">
 ${product.price}
</p>
```

- **Nombre (**h1**).**
- **Categoría (**span**).**
- **Descripción y precio.**

---

## 🔢 Controles de cantidad

```
<div className="flex items-center justify-center gap-8">
 <motion.button
 whileTap={{ scale: 0.95 }}
 onClick={() => handleQuantityChange(-1)}
 className="p-3 border-2 border-gray-300 dark:border-gray-600 rounded-full
 hover:bg-gray-100 dark:hover:bg-gray-700"
 >
 <MinusIcon className="w-6 h-6 text-gray-600 dark:text-gray-300" />
 </motion.button>

 <span className="text-3xl font-bold text-gray-900 dark:text-white w-12 text-
 center">
 {quantity}

```

```

<motion.button
 whileTap={{ scale: 0.95 }}
 onClick={() => handleQuantityChange(1)}
 className="p-3 border-2 border-gray-300 dark:border-gray-600 rounded-full
hover:bg-gray-100 dark:hover:bg-gray-700"
>
 <PlusIcon className="w-6 h-6 text-gray-600 dark:text-gray-300" />
</motion.button>
</div>

```

- **Botones - y +:**
    - `whileTap={{ scale: 0.95 }}` → efecto clic.
    - **No permite valores menores a 1.**
- 

## Botón "Añadir al carrito"

```

<motion.button
 whileHover={{ scale: 1.02 }}
 whileTap={{ scale: 0.98 }}
 onClick={handleAddToCart}
 className="w-full flex items-center justify-center gap-2 px-4 py-4 border
border-transparent text-lg font-medium rounded-xl text-white bg-primary hover:bg-
secondary mb-4"
>
 <ShoppingCartIcon className="w-6 h-6" />
 Añadir al carrito - ${product.price * quantity).toFixed(2)}
</motion.button>

```

- **Efecto `whileHover` y `whileTap`.**
  - **Muestra el precio total dinámico.**
- 

## Exportación del componente

```
export default ProductDetailPage
```

- Permite importar `ProductDetailPage` en otros módulos.

```

```markdown
# ProductFilters/index.jsx - Filtros para productos

## 📁 Importaciones

```

```
```jsx
import PropTypes from 'prop-types'
```

- **PropTypes**: Validación de tipos de propiedades para mayor seguridad.

## Props del componente

```
function ProductFilters({
 filters,
 setFilters,
 categories,
 priceRange,
 sortOptions
})
```

- **filters**: Estado actual de los filtros.
- **setFilters**: Función para actualizar filtros.
- **categories**: Lista de categorías disponibles.
- **priceRange**: Rango mínimo y máximo de precios.
- **sortOptions**: Opciones de ordenación.

## Contenedor principal

```
return (
 <div className="mb-8 mt-8 p-4 bg-white dark:bg-gray-800 rounded-xl shadow-md">
 <div className="grid grid-cols-1 md:grid-cols-3 gap-4">
```

- **Espaciado (mb-8 mt-8)**.
- **Fondo claro (bg-white) y oscuro (dark:bg-gray-800)**.
- **grid-cols-1 md:grid-cols-3**: Layout responsive.

## Filtro de categoría

```
<div>
 <label className="block text-sm font-medium text-gray-700 dark:text-gray-300 mb-1">
 Categoría
 </label>
 <select
 value={filters.category}
 onChange={(e) => setFilters({ ...filters, category: e.target.value })}
 className="w-full px-4 py-2 border dark:border-gray-600 rounded-lg bg-white dark:bg-gray-700 text-gray-900 dark:text-white"
```

```

>
<option value="">Todas</option>
{categories.map((category) => (
 <option key={category} value={category}>{category}</option>
))
}>
</select>
</div>

```

- **select:**
    - Actualiza `filters.category` al cambiar.
    - `value=""` → Muestra "Todas" por defecto.
  - **Opciones dinámicas** (`categories.map()`).
- 

## 🔧 Filtro de precio

```

<div>
 <label className="block text-sm font-medium text-gray-700 dark:text-gray-300 mb-1">
 Precio máximo: ${filters.maxPrice}
 </label>
 <input
 type="range"
 min={priceRange.min}
 max={priceRange.max}
 value={filters.maxPrice}
 onChange={(e) => setFilters({ ...filters, maxPrice: Number(e.target.value) })}
 className="w-full"
 />
</div>

```

- **range input:**
    - Control deslizante para seleccionar `maxPrice`.
    - `filters.maxPrice` se actualiza en `setFilters`.
- 

## 🔧 Ordenar productos

```

<div>
 <label className="block text-sm font-medium text-gray-700 dark:text-gray-300 mb-1">
 Ordenar por
 </label>
 <select
 value={filters.sort}
 onChange={(e) => setFilters({ ...filters, sort: e.target.value })}>
 className="w-full px-4 py-2 border dark:border-gray-600 rounded-lg bg-white
 dark:bg-gray-700 text-gray-900 dark:text-white"
 </select>
</div>

```

```
>
 {sortOptions.map((option) => (
 <option key={option.value} value={option.value}>
 {option.label}
 </option>
)))
</select>
</div>
```

- **select** para ordenación:
  - **filters.sort** define el criterio de orden.
  - **Opciones dinámicas** (**sortOptions.map()**).

## 🔍 Validación con **PropTypes**

```
ProductFilters.propTypes = {
 filters: PropTypes.shape({
 category: PropTypes.string,
 maxPrice: PropTypes.number,
 sort: PropTypes.string
 }).isRequired,
 setFilters: PropTypes.func.isRequired,
 categories: PropTypes.arrayOf(PropTypes.string).isRequired,
 priceRange: PropTypes.shape({
 min: PropTypes.number,
 max: PropTypes.number
 }).isRequired,
 sortOptions: PropTypes.arrayOf(PropTypes.shape({
 value: PropTypes.string,
 label: PropTypes.string
 })).isRequired
}
```

- **Valida tipos de datos** para evitar errores en el uso del componente.

## ⬅ Exportación del componente

```
export default ProductFilters
```

- Permite importar **ProductFilters** en otros módulos.

```
```markdown
# Navbar/index.jsx - Barra de navegación
```

```
## 📁 Importaciones
```jsx
import { NavLink, useLocation } from 'react-router-dom'
import { ShoppingBagIcon, Bars3Icon, XMarkIcon, SunIcon, MoonIcon,
ComputerDesktopIcon } from '@heroicons/react/24/solid'
import { motion, AnimatePresence } from 'framer-motion'
import { useCart } from '../Context/CartContext'
import { useAuth } from '../Context/AuthContext'
import { useTheme } from '../Context/ThemeContext'
import { useState } from 'react'
```

- **react-router-dom:**
  - **NavLink:** Navegación entre páginas con estilos dinámicos.
  - **useLocation():** Detecta la ruta actual.
- **Heroicons:** Iconos para el menú, carrito y temas.
- **framer-motion:**
  - **motion:** Animaciones en botones y elementos.
  - **AnimatePresence:** Manejo de animaciones al montar/desmontar componentes.
- **Contextos:**
  - **useCart():** Estado del carrito (**CartContext**).
  - **useAuth():** Información del usuario (**AuthContext**).
  - **useTheme():** Tema visual (**ThemeContext**).
- **useState():**
  - **isMenuOpen:** Estado del menú en móviles.
  - **isThemeMenuOpen:** Control del menú de temas.

## STATES Y VARIABLES

```
const location = useLocation()
const { toggleCart, cart } = useCart()
const { user, logout } = useAuth()
const { theme, toggleTheme } = useTheme()
const [isMenuOpen, setIsMenuOpen] = useState(false)
const [isThemeMenuOpen, setIsThemeMenuOpen] = useState(false)
```

- **useLocation():** Detecta la página actual.
- **toggleCart():** Abre/cierra el carrito.
- **cart:** Lista de productos en el carrito.
- **user y logout()\*\*:** Estado del usuario y cierre de sesión.
- **theme y toggleTheme():** Modo oscuro/claro/sistema.

## CONTADOR DE PRODUCTOS EN EL CARRITO

```
const cartItemsCount = cart.reduce((total, item) => total + item.quantity, 0)
```

- Suma la cantidad de productos en el carrito.

## ⌚ Configuración de temas

```
const themeOptions = [
 { value: 'light', icon: SunIcon, label: 'Claro' },
 { value: 'dark', icon: MoonIcon, label: 'Oscuro' },
 { value: 'system', icon: ComputerDesktopIcon, label: 'Sistema' }
]

const getCurrentThemeIcon = () => {
 const currentTheme = themeOptions.find(option => option.value === theme)
 return currentTheme ? currentTheme.icon : themeOptions[0].icon
}

const ThemeIcon = getCurrentThemeIcon()
```

- **themeOptions**: Lista de temas disponibles.
- **getCurrentThemeIcon()**: Obtiene el ícono según el tema actual.

## 📷 Renderizado principal

```
return (
 <motion.nav
 initial={{ y: -100 }}
 animate={{ y: 0 }}
 className="flex justify-between items-center fixed z-10 top-0 w-full py-5 px-8
text-sm font-light bg-white dark:bg-gray-800 dark:text-white shadow-md transition-
colors"
 >
```

- **Animación**: La **Navbar** aparece desde arriba (**y: -100 → 0**).
- **Fija en la parte superior** (**fixed top-0**).
- **Modo oscuro**:
  - **bg-white dark:bg-gray-800 dark:text-white**.
- **Sombra** (**shadow-md**).

## 🔗 Logo y menú en móviles

```
<div className='flex items-center gap-3'>
 <motion.div
 whileHover={{ scale: 1.05 }}
 className='font-semibold text-lg'>
 <NavLink to='/'>Shopi</NavLink>
```

```
</motion.div>

<button className="md:hidden" onClick={() => setIsMenuOpen(!isMenuOpen)}>
 {isMenuOpen ? <XMarkIcon className="h-6 w-6" /> : <Bars3Icon className="h-6 w-6" />}
</button>
</div>
```

- **Logo con efecto hover** (`scale: 1.05`).
  - **Menú hamburguesa** (`Bars3Icon` / `XMarkIcon`):
    - Solo visible en móviles (`md:hidden`).
    - Alterna `isMenuOpen`.
- 

## 🔗 Menú en desktop

```
<ul className='hidden md:flex items-center gap-3'>
 <motion.li whileHover={{ y: -2 }}>
 <NavLink to='/' className={({ isActive }) => isActive ? activeStyle : undefined}>Home</NavLink>
 </motion.li>
 <motion.li whileHover={{ y: -2 }}>
 <NavLink to='/products' className={({ isActive }) => isActive ? activeStyle : undefined}>Products</NavLink>
 </motion.li>
 {user && (
 <>
 <motion.li whileHover={{ y: -2 }}>
 <NavLink to='/my-orders' className={({ isActive }) => isActive ? activeStyle : undefined}>My Orders</NavLink>
 </motion.li>
 <motion.li whileHover={{ y: -2 }}>
 <NavLink to='/my-account' className={({ isActive }) => isActive ? activeStyle : undefined}>My Account</NavLink>
 </motion.li>
 </>
)}

```

- **Links con efecto hover** (`y: -2`).
  - **Links dinámicos** (`user` → muestra opciones adicionales).
- 

## 🔗 Menú móvil (`AnimatePresence`)

```
<AnimatePresence>
 {isMenuOpen && (
 <motion.div
```

```

 initial={{ opacity: 0, y: -20 }}
 animate={{ opacity: 1, y: 0 }}
 exit={{ opacity: 0, y: -20 }}
 className="absolute top-full left-0 w-full bg-white dark:bg-gray-800 shadow-lg md:hidden"
 >
 <ul className='flex flex-col py-4 px-8 gap-4 dark:text-gray-300'>
 {['/', '/products'].map((path) => (
 <motion.li key={path} whileHover={{ x: 5 }} onClick={() =>
setIsMenuOpen(false)}>
 <NavLink to={path} className={({ isActive }) => isActive ? activeStyle : undefined}>
 {path === '/' ? 'Home' : 'Products'}
 </NavLink>
 </motion.li>
)))

</motion.div>
)
</AnimatePresence>

```

- **AnimatePresence:**
    - Maneja la animación de entrada/salida.
  - **Botones** `whileHover={{ x: 5 }}`.
- 

## Selector de temas

```

<div className="relative">
 <motion.button
 whileHover={{ scale: 1.1 }}
 whileTap={{ scale: 0.95 }}
 onClick={() => setIsThemeMenuOpen(!isThemeMenuOpen)}
 className="p-2 rounded-lg"
 >
 <ThemeIcon className="h-5 w-5" />
 </motion.button>

 <AnimatePresence>
 {isThemeMenuOpen && (
 <>
 <motion.div className="fixed inset-0 z-40" onClick={() =>
setIsThemeMenuOpen(false)} />

 <motion.div className="absolute right-0 mt-2 py-2 w-48 bg-white dark:bg-gray-800 rounded-lg shadow-xl z-50">
 {themeOptions.map(({ value, icon: Icon, label }) => (
 <motion.button
 key={value}
 whileHover={{ x: 5 }}
 onClick={() => { toggleTheme(value); setIsThemeMenuOpen(false) }}

```

```

 className={`w-full px-4 py-2 flex items-center gap-2`}
 >
 <Icon className="h-5 w-5" />
 {label}
</motion.button>
))})
</motion.div>
</>
)
)
</AnimatePresence>
</div>

```

- **Botón de tema dinámico** (`ThemeIcon`).
- **Menú desplegable con `AnimatePresence`.**

## ◀ END Exportación del componente

```
export default Navbar
```

- Permite importar `Navbar` en otros módulos.

```

```markdown
# Hero/index.jsx - Carrusel de productos destacados

## 📁 Importaciones
```jsx
import { Swiper, SwiperSlide } from 'swiper/react'
import { Autoplay, Pagination, Navigation } from 'swiper/modules'
import { motion } from 'framer-motion'
import { Link } from 'react-router-dom'
import 'swiper/css'
import 'swiper/css/pagination'
import 'swiper/css/navigation'
import 'swiper/css/effect-fade'
```

```

- **Swiper y SwiperSlide:** Componente de carrusel (`Swiper`).
- **Módulos de Swiper:**
 - `Autoplay`: Reproducción automática.
 - `Pagination`: Indicadores de paginación.
 - `Navigation`: Botones de navegación.
- **`motion` de framer-motion:** Animaciones en elementos.
- **`Link` de react-router-dom:** Navegación a páginas de productos.
- **Estilos de Swiper:**
 - `swiper/css`: Estilos base.

- [swiper/css/pagination](#), [swiper/css/navigation](#), [swiper/css/effect-fade](#).
-

Propiedades del componente

```
function Hero({ featuredProducts }) {  
  if (!featuredProducts?.length) return null
```

- **featuredProducts**: Lista de productos destacados.
 - **Si no hay productos**, el componente no se renderiza.
-

Configuración de [Swiper](#)

```
<Swiper  
  spaceBetween={0}  
  centeredSlides={true}  
  effect="fade"  
  autoplay={{  
    delay: 5000,  
    disableOnInteraction: false,  
  }}  
  pagination={{  
    clickable: true,  
    dynamicBullets: true,  
  }}  
  navigation={true}  
  modules={[Autoplay, Pagination, Navigation]}  
  className="w-full h-full"  
>
```

- **spaceBetween={0}**: Sin espacios entre slides.
 - **centeredSlides={true}**: Centra los slides.
 - **effect="fade"**: Transición de fundido.
 - **autoplay**:
 - Retraso de **5000ms** (5 segundos).
 - No se desactiva al interactuar (**disableOnInteraction: false**).
 - **pagination**:
 - Clickeable.
 - Usa bullets dinámicos.
 - **navigation={true}**: Activa botones de navegación.
-

Renderizado de slides

```
{featuredProducts.map((product) => (
  <SwiperSlide key={product.id} className="w-full">
```

- **Iteración sobre `featuredProducts`.**
- **Cada producto es un `SwiperSlide`.**

🔗 Contenedor del slide

```
<div className="relative flex items-center justify-center w-full h-full px-4 md:px-8">
  <div className="container relative z-10 flex flex-col md:flex-row items-center justify-between gap-8">
```

- **Flexbox** para disposición del contenido (`items-center justify-center`).
- `md:flex-row`: Layout adaptable en pantallas grandes.

📝 Sección de texto

```
<motion.div
  initial={{ opacity: 0, x: -50 }}
  animate={{ opacity: 1, x: 0 }}
  transition={{ duration: 0.5 }}
  className="max-w-xl"
>
  <h2 className="mb-4 text-3xl font-bold text-gray-800 dark:text-white md:text-5xl lg:text-6xl">
    {product.title}
  </h2>
  <p className="mb-8 text-lg text-gray-600 dark:text-gray-300 line-clamp-3">
    {product.description}
  </p>
  <div className="flex items-center gap-4">
    <Link
      to={`/product/${product.id}`}
      className="px-8 py-3 text-white transition-colors bg-primary hover:bg-secondary rounded-xl">
      >
      Ver Producto
    </Link>
    <span className="text-3xl font-bold text-gray-800 dark:text-white">
      ${product.price}
    </span>
  </div>
</motion.div>
```

- **Animación (`motion.div`):**

- **initial**: Opacidad 0, desplazado x: -50.
 - **animate**: Opacidad 1, x: 0.
 - **transition**: 0.5s.
 - **line-clamp-3**: Limita la descripción a 3 líneas.
 - **Botón "Ver Producto" (<Link>)**:
 - **bg-primary hover:bg-secondary**: Cambia de color al pasar el ratón.
-

Sección de imagen

```
<motion.div
  initial={{ opacity: 0, scale: 0.8 }}
  animate={{ opacity: 1, scale: 1 }}
  transition={{ duration: 0.5 }}
  className="w-full md:w-1/2 h-[40vh] md:h-[60vh]"
>
  <img
    src={product.image}
    alt={product.title}
    className="object-contain w-full h-full"
  />
</motion.div>
```

- **Animación (motion.div)**:
 - **initial**: Opacidad 0, scale: 0.8.
 - **animate**: Opacidad 1, scale: 1.
 - **object-contain**: Evita distorsión en la imagen.
-

Exportación del componente

```
export default Hero
```

- Permite importar **Hero** en otros módulos.

```
```markdown
Cart/index.jsx - Carrito de compras

Importaciones
```jsx
import { motion, AnimatePresence } from 'framer-motion'
import { FaPlus, FaMinus, FaTrash } from 'react-icons/fa'
import { useCart } from '../../Context/CartContext'
import { useNavigate } from 'react-router-dom'
import { useState } from 'react'
```

- **motion y AnimatePresence de framer-motion**: Animaciones en la UI.
- **Iconos de react-icons/fa**: FaPlus, FaMinus, FaTrash.
- **useCart**: Contexto del carrito de compras.
- **useNavigate**: Redirección a la página de pago.
- **useState**: Estado local para mensaje de login.

STATES Y VARIABLES

```
const navigate = useNavigate()
const { cart, isCartOpen, toggleCart, updateQuantity, removeFromCart, cartTotal } = useCart()
const [showLoginMessage, setShowLoginMessage] = useState(false)
```

- **cart**: Lista de productos en el carrito.
- **isCartOpen**: Estado del carrito (abierto/cerrado).
- **toggleCart()**: Función para abrir/cerrar el carrito.
- **updateQuantity()**: Modifica la cantidad de un producto.
- **removeFromCart()**: Elimina un producto del carrito.
- **cartTotal**: Total de la compra.
- **showLoginMessage**: Estado para mostrar mensaje si el usuario no está autenticado.

FUNCIÓN PARA FINALIZAR COMpra

```
const handleCheckout = () => {
  const user = localStorage.getItem('user')
  if (!user) {
    setShowLoginMessage(true)
    setTimeout(() => setShowLoginMessage(false), 3000)
    return
  }
  toggleCart()
  navigate('/checkout')
}
```

- **Verifica si el usuario está autenticado** (`localStorage.getItem('user')`).
- **Si no está autenticado**, muestra un mensaje durante 3 segundos.

FUNCIÓN PARA ELIMINAR PRODUCTOS

```
const handleRemoveProduct = (productId) => {
  removeFromCart(productId)
  if (cart.length === 1) {
```

```

    setTimeout(() => {
      toggleCart()
    }, 300)
  }
}

```

- **Elimina el producto y cierra el carrito** si queda vacío.

💻 Renderizado del carrito

```

return (
  <AnimatePresence>
    {isCartOpen && (
      <>
        <motion.div
          initial={{ opacity: 0 }}
          animate={{ opacity: 0.5 }}
          exit={{ opacity: 0 }}
          onClick={toggleCart}
          className="fixed inset-0 z-40 bg-black backdrop-blur-sm"
        />

```

- **AnimatePresence** gestiona la animación al abrir/cerrar el carrito.
- **Fondo oscuro con desenfoque (backdrop-blur-sm)**.

🔧 Panel lateral del carrito

```

<motion.div
  initial={{ x: '100%' }}
  animate={{ x: 0 }}
  exit={{ x: '100%' }}
  transition={{ type: 'tween' }}
  className="fixed top-0 right-0 z-50 w-full h-full max-w-md p-6 bg-white dark:bg-gray-800 shadow-lg"
>

```

- **Panel lateral (right-0)** que se desliza (x: '100%' → 0).
- **max-w-md** limita el ancho.

🔧 Cabecera del carrito

```

<div className="flex items-center justify-between mb-6">
  <h2 className="text-2xl font-semibold text-gray-900 dark:text-
```

```
white">Carrito</h2>
  <button
    onClick={toggleCart}
    className="p-2 rounded-full hover:bg-gray-100 dark:hover:bg-gray-700 text-gray-500 dark:text-gray-400"
  >
  ×
  </button>
</div>
```

- **Título.**
 - **Botón para cerrar el carrito.**
-

⌚ Mensaje de login

```
<AnimatePresence>
  {showLoginMessage && (
    <motion.div
      initial={{ opacity: 0, y: -20 }}
      animate={{ opacity: 1, y: 0 }}
      exit={{ opacity: 0, y: -20 }}
      className="p-4 mb-4 text-center text-white bg-red-500 rounded-lg"
    >
      Debes iniciar sesión para realizar la compra
    </motion.div>
  )}
</AnimatePresence>
```

- **Aparece con animación (y: -20 → 0)** si el usuario no está autenticado.
-

🛒 Lista de productos

```
{cart.length === 0 ? (
  <div className="flex-1 flex items-center justify-center">
    <p className="text-gray-500 dark:text-gray-400">Tu carrito está vacío</p>
  </div>
) : (
<>
  <div className="flex-1 overflow-y-auto">
    <AnimatePresence mode="popLayout">
      {cart.map((item) => (
        <motion.div
          key={item.id}
          layout
          initial={{ opacity: 0, y: 20 }}
          animate={{ opacity: 1, y: 0 }}
          exit={{ opacity: 0, y: -20 }}>
```

```
    className="flex gap-4 p-4 border-b dark:border-gray-700"
  >
```

- **Si el carrito está vacío**, muestra un mensaje.
- **Si hay productos**, los muestra con animaciones.

⌚ Producto en el carrito

```
<img
  src={item.image}
  alt={item.title}
  className="w-20 h-20 object-contain rounded-lg bg-white dark:bg-gray-700"
/>

<div className="flex-1">
  <h3 className="font-medium dark:text-white">{item.title}</h3>
  <p className="text-primary dark:text-secondary font-bold">
    ${(item.price * item.quantity).toFixed(2)}
  </p>
```

- **Imagen y nombre** del producto.
- **Precio total** (**precio * cantidad**).

🔢 Controles de cantidad

```
<div className="flex items-center gap-2 mt-2">
  <motion.button
    whileTap={{ scale: 0.95 }}
    onClick={() => updateQuantity(item.id, item.quantity - 1)}
    className="p-1 rounded-full hover:bg-gray-100 dark:hover:bg-gray-700
dark:text-gray-300"
  >
    <FaMinus className="w-4 h-4" />
  </motion.button>

  <span className="w-8 text-center dark:text-gray-300">{item.quantity}</span>

  <motion.button
    whileTap={{ scale: 0.95 }}
    onClick={() => updateQuantity(item.id, item.quantity + 1)}
    className="p-1 rounded-full hover:bg-gray-100 dark:hover:bg-gray-700
dark:text-gray-300"
  >
    <FaPlus className="w-4 h-4" />
  </motion.button>

  <motion.button
```

```

    whileTap={{ scale: 0.95 }}
    onClick={() => handleRemoveProduct(item.id)}
    className="p-1 text-red-500 rounded-full hover:bg-red-50 dark:hover:bg-red-900
    ml-auto"
  >
  <FaTrash className="w-4 h-4" />
</motion.button>
</div>

```

- **Botones + y -** para modificar la cantidad.
 - **Botón 🗑 para eliminar el producto.**
-

🛒 Total y botón de checkout

```

<div className="mt-6 pt-6 border-t dark:border-gray-700">
  <div className="flex justify-between items-center mb-4">
    <span className="text-lg dark:text-white">Total:</span>
    <span className="text-2xl font-bold text-primary dark:text-secondary">
      ${cartTotal.toFixed(2)}
    </span>
  </div>

  <motion.button
    whileHover={{ scale: 1.02 }}
    whileTap={{ scale: 0.98 }}
    onClick={handleCheckout}
    className="w-full py-3 text-white transition-colors bg-primary hover:bg-
    secondary rounded-xl"
  >
    Finalizar Compra
  </motion.button>
</div>

```

- **Total dinámico.**
 - **Botón "Finalizar Compra".**
-

⬅ END Exportación del componente

```
export default Cart
```

- Permite importar **Cart** en otros módulos.

```
```markdown
```

```
CartContext.jsx - Contexto global del carrito de compras

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

- **createContext y useContext:** Creación y acceso al contexto del carrito.
- **useState:** Manejo del estado del carrito.
- **useEffect:** Persistencia del carrito en `localStorage`.

⌚ Creación del contexto

```
const CartContext = createContext()
```

- **Crea un nuevo contexto** para gestionar el estado global del carrito.

⟳ Componente `CartProvider`

```
export function CartProvider({ children }) {
```

- **Proveedor del contexto (`CartProvider`).**
- **Permite que cualquier componente acceda al estado del carrito.**

❖ Estados del carrito

```
const [isCartOpen, setIsCartOpen] = useState(false)
const [cart, setCart] = useState(() => {
  const savedCart = localStorage.getItem('cart')
  return savedCart ? JSON.parse(savedCart) : []
})
```

- **isCartOpen:** Controla si el carrito está abierto o cerrado.
- **cart:**
 - Se inicializa con los datos guardados en `localStorage`.
 - Si no hay datos previos, se establece como un array vacío.

⌚ Persistencia con `localStorage`

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

- Guarda automáticamente el carrito en `localStorage` cada vez que cambia.

⌚ Cálculo del total del carrito

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

- Suma el precio de cada producto multiplicado por su cantidad.

🛒 Funciones del carrito

☒ Mostrar/ocultar carrito

```
const toggleCart = () => setIsCartOpen(!isCartOpen)
```

- Alterna entre abrir y cerrar el carrito.

✚ Añadir productos al carrito

```
const addToCart = (product) => {
  setCart(currentCart => {
    const existingItem = currentCart.find(item => item.id === product.id)
    if (existingItem) {
      return currentCart.map(item =>
        item.id === product.id
          ? { ...item, quantity: item.quantity + product.quantity }
          : item
      )
    }
    return [...currentCart, product]
  })
}
```

- Si el producto ya existe, incrementa su cantidad.
- Si no existe, lo añade al carrito.

⤓ Actualizar cantidad de un producto

```
const updateQuantity = (productId, newQuantity) => {
  if (newQuantity < 1) {
    removeFromCart(productId)
    return
  }
  setCart(currentCart =>
    currentCart.map(item =>
      item.id === productId
        ? { ...item, quantity: newQuantity }
        : item
    )
  )
}
```

- Si la cantidad es menor a 1, elimina el producto.
- Si la cantidad es válida, actualiza el producto.

Eliminar productos del carrito

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

- Filtra el carrito eliminando el producto con el id indicado.

Vaciar el carrito

```
const clearCart = () => {
  setCart([])
  setIsCartOpen(false)
}
```

- Limpia todos los productos y cierra el carrito.

Proveedor del contexto

```
return (
  <CartContext.Provider value={{
    cart,
    isCartOpen,
    toggleCart,
    addToCart,
    updateQuantity,
    removeFromCart,
    cartTotal,
```

```
    clearCart
  }})
  {children}
</CartContext.Provider>
)
```

- Proporciona acceso global a los valores y funciones del carrito.

⌚ Hook personalizado useCart

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

- Facilita el acceso al contexto sin necesidad de importar useContext manualmente.

⬅ END Exportación del componente

```
export { CartProvider, useCart }
```

- Permite importar CartProvider y useCart en otros módulos.

```
```markdown
Checkout/index.jsx - Página de finalización de compra

📂 Importaciones
```jsx
import { useState } from 'react'
import { useNavigate } from 'react-router-dom'
import { motion, AnimatePresence } from 'framer-motion'
import { useCart } from '../../Context/CartContext'
import { useAuth } from '../../Context/AuthContext'
import Layout from '../../Components/Layout'
```

- useNavigate: Redirección tras la compra.
- motion y AnimatePresence de framer-motion: Animaciones en el formulario y mensajes.
- useCart: Accede a los productos y total del carrito (CartContext).
- useAuth: Información del usuario (AuthContext).
- Layout: Estructura base de la página.

🕒 Estados del componente

```
const navigate = useNavigate()
const { cart, cartTotal, clearCart } = useCart()
const { user } = useAuth()
const [showSuccess, setShowSuccess] = useState(false)
const [formData, setFormData] = useState({
  fullName: '',
  address: '',
  city: '',
  cardNumber: '',
  expiryDate: '',
  cvv: ''
})
```

- **cart**: Lista de productos en el carrito.
- **cartTotal**: Total de la compra.
- **clearCart()**: Vacía el carrito tras la compra.
- **user**: Información del usuario autenticado.
- **showSuccess**: Controla la visibilidad del mensaje de compra exitosa.
- **formData**: Contiene la información del comprador.

📝 Manejo del formulario

📌 Guardar cambios en el formulario

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

- **Actualiza el estado formData** cuando el usuario ingresa datos.

❖ Procesar la compra

```
const handleSubmit = async (e) => {
  e.preventDefault()

  const order = {
    id: `ORD-${Date.now()}`,
    date: new Date().toISOString(),
    products: cart,
    total: cartTotal,
    shipping: formData,
    status: 'Completado',
  }
```

```

        user: user.email
    }

const existingOrders = JSON.parse(localStorage.getItem('orders') || '[]')

localStorage.setItem('orders', JSON.stringify([...existingOrders, order]))

setShowSuccess(true)

setTimeout(() => {
    clearCart()
    navigate('/my-orders')
}, 2000)
}

```

- **Crea una orden con ID único.**
 - **Guarda la orden en `localStorage`.**
 - **Muestra el mensaje de éxito y vacía el carrito.**
 - **Redirige a la página "Mis Órdenes" en 2 segundos.**
-

Renderizado principal

```

return (
<Layout>
  <div className="min-h-screen bg-gray-50 dark:bg-gray-900 py-12 px-4 sm:px-6 lg:px-8">
    <div className="max-w-3xl mx-auto">
      <motion.div
        initial={{ opacity: 0, y: 20 }}
        animate={{ opacity: 1, y: 0 }}
        className="bg-white dark:bg-gray-800 p-8 rounded-xl shadow-lg">
    >

```

- **Layout base (`Layout`).**
 - **Tarjeta (`div`) con efecto de entrada (`y: 20 → 0`).**
-

Mensaje de éxito

```

<AnimatePresence>
  {showSuccess && (
    <motion.div
      initial={{ opacity: 0, scale: 0.8 }}
      animate={{ opacity: 1, scale: 1 }}
      exit={{ opacity: 0, scale: 0.8 }}
      className="absolute inset-0 flex items-center justify-center bg-white bg-opacity-90 z-50 rounded-xl"

```

```

    >
    <div className="text-center p-8">
      <motion.div
        initial={{ scale: 0 }}
        animate={{ scale: 1 }}
        className="w-16 h-16 bg-green-500 rounded-full mx-auto mb-4 flex items-center justify-center">
        >
          <svg className="w-8 h-8 text-white" fill="none" stroke="currentColor"
viewBox="0 0 24 24">
            <path strokeLinecap="round" strokeLinejoin="round" strokeWidth="2"
d="M5 13l4 4L19 7" />
          </svg>
        </motion.div>
      <h3 className="text-2xl font-bold text-gray-900 mb-2">
        ¡Compra realizada con éxito!
      </h3>
      <p className="text-gray-600">Redirigiendo a tus pedidos...</p>
    </div>
  </motion.div>
)
</AnimatePresence>

```

- **Aparece cuando showSuccess === true.**
- **Animación de escala (scale: 0 → 1).**

Resumen del pedido

```

<div className="mb-8">
  <h3 className="text-xl font-semibold text-gray-900 dark:text-white mb-4">
    Resumen del Pedido
  </h3>
  <div className="space-y-4">
    {cart.map((item) => (
      <div key={item.id} className="flex items-center gap-4 p-4 bg-gray-50
dark:bg-gray-700 rounded-lg">
        <img src={item.image} alt={item.title} className="w-20 h-20 object-contain
bg-white dark:bg-gray-600 rounded-lg" />
        <div className="flex-1">
          <h4 className="font-medium text-gray-900 dark:text-white">{item.title}</h4>
          <p className="text-primary dark:text-secondary">${(item.price *
item.quantity).toFixed(2)}</p>
        </div>
      </div>
    )))
  </div>
  <div className="mt-4 flex justify-between items-center text-lg font-semibold">
    <span className="text-gray-900 dark:text-white">Total:</span>
    <span className="text-primary dark:text-secondary">${cartTotal.toFixed(2)}</span>
  </div>
</div>

```

```
</span>
  </div>
</div>
```

-
- **Muestra cada producto con imagen, título y precio.**
 - **Calcula el total.**

🔗 Formulario de checkout

```
<form onSubmit={handleSubmit} className="space-y-6">
  <div className="grid grid-cols-1 gap-6 md:grid-cols-2">
```

-
- **Formulario dividido en dos columnas (md:grid-cols-2).**

📦 Datos de envío

```
<div>
  <h3 className="text-xl font-semibold mb-4">Datos de Envío</h3>
  <div className="space-y-4">
    <div>
      <label className="block text-sm font-medium text-gray-700 dark:text-gray-300">Nombre completo</label>
      <input
        type="text"
        name="fullName"
        required
        value={formData.fullName}
        onChange={handleChange}
        className="mt-1 block w-full px-3 py-2 border border-gray-300 dark:border-gray-600 rounded-lg bg-white dark:bg-gray-700 text-gray-900 dark:text-white"
      />
    </div>
```

-
- **Campos para el nombre, dirección y ciudad.**

💳 Datos de pago

```
<div>
  <h3 className="text-xl font-semibold mb-4">Datos de Pago</h3>
  <div className="space-y-4">
    <div>
      <label className="block text-sm font-medium text-gray-700 dark:text-gray-300">Número de tarjeta</label>
      <input type="text" name="cardNumber" required maxLength="16" value=
```

```
{formData.cardNumber} onChange={handleChange} />
</div>
```

- Incluye tarjeta, fecha de expiración y CVV.

☒ Botón de compra

```
<motion.button
  whileHover={{ scale: 1.02 }}
  whileTap={{ scale: 0.98 }}
  type="submit"
  className="w-full py-3 text-white transition-colors bg-primary hover:bg-
secondary rounded-xl mt-8"
>
  Finalizar Compra
</motion.button>
```

- Efecto hover y tap con `motion.button`.

⬅ END Exportación del componente

```
export default Checkout
```

- Permite importar `Checkout` en otros módulos.

```
# Modo Oscuro

```markdown
ThemeContext.jsx - Contexto global para el manejo de temas

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

- `createContext` y `useContext`: Creación y uso del contexto global de temas.
- `useEffect`: Sincronización del tema con `localStorage` y el sistema.
- `useState`: Manejo del estado del tema.

🌐 Creación del contexto

```
const ThemeContext = createContext()
```

- **Crea un nuevo contexto** para gestionar el tema de la aplicación.

⌚ Componente ThemeProvider

```
export function ThemeProvider({ children }) {
```

- **Proveedor del contexto** (**ThemeProvider**).
- **Permite que cualquier componente acceda y modifique el tema.**

🕒 Estado del tema

```
const [theme, setTheme] = useState(() => {
  const savedTheme = localStorage.getItem('theme')
  if (savedTheme) return savedTheme

  if (window.matchMedia('(prefers-color-scheme: dark)').matches) return 'dark'

  return 'light'
})
```

- **Inicializa con el tema guardado en **localStorage**.**
- **Si no hay tema guardado, detecta la preferencia del sistema.**
- **Si no hay preferencia, usa **light** como predeterminado.**

⌚ Sincronización con **document.documentElement**

```
useEffect(() => {
  const root = window.document.documentElement

  root.classList.remove('light', 'dark', 'system')

  if (theme === 'system') {
    if (window.matchMedia('(prefers-color-scheme: dark)').matches) {
      root.classList.add('dark')
    } else {
      root.classList.add('light')
    }
  } else {
    root.classList.add(theme)
  }
})
```

```
localStorage.setItem('theme', theme)
}, [theme])
```

- **Elimina clases previas (light, dark, system).**
- **Si el tema es system, ajusta según la configuración del usuario.**
- **Guarda la preferencia en localStorage.**

⌚ Detección de cambios en la preferencia del sistema

```
useEffect(() => {
  if (theme === 'system') {
    const mediaQuery = window.matchMedia('(prefers-color-scheme: dark)')
    const handleChange = (e) => {
      const root = window.document.documentElement
      root.classList.remove('light', 'dark')
      root.classList.add(e.matches ? 'dark' : 'light')
    }

    mediaQuery.addEventListener('change', handleChange)
    return () => mediaQuery.removeEventListener('change', handleChange)
  }
}, [theme])
```

- **Si el tema es system, detecta cambios en prefers-color-scheme.**
- **Cambia el tema dinámicamente cuando el usuario lo modifica en su sistema.**

👤 Función para cambiar el tema

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

- **Permite cambiar el tema (light, dark o system).**

⌚ Proveedor del contexto

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

- Proporciona **theme** y **toggleTheme** a los componentes hijos.
-

⌚ Hook personalizado **useTheme**

```
export function useTheme() {
  const context = useContext(ThemeContext)
  if (context === undefined) {
    throw new Error('useTheme must be used within a ThemeProvider')
  }
  return context
}
```

- **Facilita el acceso al contexto** sin necesidad de importar **useContext** manualmente.
 - **Lanza un error si se usa fuera de **ThemeProvider**.**
-

⬅ END Exportación del componente

```
export { ThemeProvider, useTheme }
```

- Permite importar **ThemeProvider** y **useTheme** en otros módulos.

```
```markdown
SignIn/index.jsx - Página de inicio de sesión

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

- **useState**: Manejo del estado del formulario y errores.
 - **useNavigate**: Redirección tras el inicio de sesión.
 - **motion de framer-motion**: Animaciones en el formulario.
 - **useAuth**: Acceso a la función de autenticación (**AuthContext**).
 - **Layout**: Estructura base de la página.
-

🕒 Estados del componente

```
const navigate = useNavigate()
const { login } = useAuth()

const [formData, setFormData] = useState({
  email: '',
  password: ''
})
const [error, setError] = useState('')
```

- **formData**: Almacena los datos ingresados en el formulario.
- **error**: Guarda mensajes de error si el formulario es inválido.

📝 Manejo del formulario

❖ Validación y autenticación

```
const handleSubmit = (e) => {
  e.preventDefault()
  setError('')

  if (formData.email && formData.password) {
    login({
      email: formData.email,
      name: formData.email.split('@')[0]
    })
    navigate('/')
  } else {
    setError('Por favor, completa todos los campos')
  }
}
```

- **Verifica que el email y la contraseña no estén vacíos.**
- **Llama a `login()` con los datos ingresados.**
- **Extrae el nombre del email (`nombre@correo.com` → `nombre`).**
- **Si el login es exitoso, redirige a la página principal (`navigate('/')`).**

❖ Manejo de cambios en los inputs

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

- Actualiza **formData** dinámicamente conforme el usuario escribe.
-

🖼 Renderizado principal

```
return (
  <Layout>
    <div className="min-h-screen bg-gray-50 dark:bg-gray-900 py-12 px-4 sm:px-6 lg:px-8">
      <div className="max-w-md mx-auto">
        <motion.div
          initial={{ opacity: 0, y: 20 }}
          animate={{ opacity: 1, y: 0 }}
          className="bg-white dark:bg-gray-800 p-8 rounded-xl shadow-lg">
```

- Usa **Layout** para la estructura base.
 - Crea un contenedor centrado (**max-w-md mx-auto**).
 - Aplica una animación de entrada (**y: 20 → 0**).
-

✖ Mensaje de error condicional

```
{error && (
  <div className="mb-4 p-4 text-red-700 bg-red-100 rounded-lg">
    {error}
  </div>
)}
```

- Se muestra solo si **error** contiene un mensaje.
 - Diseñado con fondo rojo claro (**bg-red-100**) y texto rojo (**text-red-700**).
-

🔑 Formulario de inicio de sesión

```
<form onSubmit={handleSubmit} className="space-y-6">
  <div>
    <label htmlFor="email" className="block text-sm font-medium text-gray-700 dark:text-gray-300">
      Email
    </label>
    <input
      id="email"
      name="email"
      type="email"
      required
      value={formData.email}>
```

```

    onChange={handleChange}
    className="mt-1 block w-full px-3 py-2 border border-gray-300 dark:border-
gray-600 rounded-lg bg-white dark:bg-gray-700 text-gray-900 dark:text-white
focus:ring-primary focus:border-primary"
  />
</div>

<div>
  <label htmlFor="password" className="block text-sm font-medium text-gray-700
dark:text-gray-300">
    Contraseña
  </label>
  <input
    id="password"
    name="password"
    type="password"
    required
    value={formData.password}
    onChange={handleChange}
    className="mt-1 block w-full px-3 py-2 border border-gray-300 dark:border-
gray-600 rounded-lg bg-white dark:bg-gray-700 text-gray-900 dark:text-white
focus:ring-primary focus:border-primary"
  />
</div>

```

- **Campos de email y password:**

- **required:** Evita envío sin completar campos.
 - **Se actualizan en formData mediante handleChange().**
 - **Diseño responsive y accesible.**
-

☒ Botón de envío

```

<motion.button
  whileHover={{ scale: 1.02 }}
  whileTap={{ scale: 0.98 }}
  type="submit"
  className="w-full flex justify-center py-3 px-4 border border-transparent
rounded-xl shadow-sm text-sm font-medium text-white bg-primary hover:bg-secondary
focus:outline-none focus:ring-2 focus:ring-offset-2 focus:ring-primary"
>
  Iniciar Sesión
</motion.button>

```

-
- **Animaciones con motion.button (scale: 1.02 en hover).**
 - **Diseño accesible (focus:ring).**
 - **Colores dinámicos (bg-primary hover:bg-secondary).**
-

◀ END Exportación del componente

```
export default SignIn
```

- Permite importar **SignIn** en otros módulos.

```
```markdown
MyOrders/index.jsx - Página de pedidos del usuario

📁 Importaciones
```jsx
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**: Manejo del estado de las órdenes.
- **Link de react-router-dom**: Permite redireccionar a la página de productos.
- **motion de framer-motion**: Aplica animaciones en los pedidos.
- **useAuth**: Accede a la información del usuario (**AuthContext**).
- **Layout**: Proporciona la estructura base de la página.

🕒 Estado del componente

```
const { user } = useAuth()

const [orders, setOrders] = useState(() => {
  const savedOrders = localStorage.getItem('orders')
  return savedOrders ? JSON.parse(savedOrders) : []
})
```

- **user**: Contiene la información del usuario autenticado.
- **orders**:
 - Se inicializa con los pedidos almacenados en **localStorage**.
 - Si no hay pedidos guardados, comienza 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 en formato ISO en una fecha legible en español.**
 - **Ejemplo:** "2024-01-15T14:30:00Z" → "15 de enero de 2024".
-

Renderizado principal

```

return (
  <Layout>
    <div className="min-h-screen bg-gray-50 dark:bg-gray-900 py-12 px-4 sm:px-6 lg:px-8">
      <div className="max-w-3xl mx-auto">
        <motion.div
          initial={{ opacity: 0, y: 20 }}
          animate={{ opacity: 1, y: 0 }}
          className="bg-white dark:bg-gray-800 p-8 rounded-xl shadow-lg"
        >

```

- **Usa `Layout` para la estructura base.**
 - **Aplica una animación de entrada (`y: 20 → 0`).**
-

Si no hay pedidos

```

{orders.length === 0 ? (
  <div className="text-center py-12">
    <p className="text-gray-500 dark:text-gray-400">No tienes pedidos
realizados</p>
    <Link
      to="/products"
      className="mt-4 inline-block text-primary hover:text-secondary"
    >
      Ir a comprar
    </Link>
  </div>
) : (

```

- **Si el usuario no tiene pedidos,** muestra un mensaje con un enlace a la tienda.
-

Listado de pedidos

```
<div className="space-y-6">
  {orders.map((order) => (
    <motion.div
      key={order.id}
      initial={{ opacity: 0 }}
      animate={{ opacity: 1 }}
      className="border dark:border-gray-700 rounded-lg p-6 hover:shadow-md
transition-shadow dark:bg-gray-700"
    >
```

- Recorre el array **orders** y genera una tarjeta por cada pedido.
- Animación de aparición (**opacity: 0 → 1**).

🔗 Información del pedido

```
<div className="flex justify-between items-start mb-4">
  <div>
    <h3 className="text-lg font-semibold dark:text-white">
      Pedido #{order.id}
    </h3>
    <p className="text-gray-500 dark:text-gray-400">
      {formatDate(order.date)}
    </p>
  </div>
  <span className="px-3 py-1 bg-green-100 text-green-800 rounded-full text-sm">
    {order.status}
  </span>
</div>
```

- Muestra el número de pedido y la fecha formateada.
- El estado (Completado) se muestra en una etiqueta verde (**bg-green-100**).

📋 Productos en la orden

```
<div className="space-y-2">
  {order.products.map((product) => (
    <div
      key={`${order.id}-${product.id}`}
      className="flex justify-between items-center"
    >
      <div className="flex items-center gap-4">
        <img
          src={product.image}
          alt={product.title}
          className="w-12 h-12 object-contain rounded"
        />
```

```

<div>
  <p className="font-medium">{product.title}</p>
  <p className="text-sm text-gray-500">
    Cantidad: {product.quantity}
  </p>
  </div>
</div>
<p className="font-medium">
  ${(product.price * product.quantity).toFixed(2)}
</p>
</div>
))}
```

- **Muestra cada producto del pedido** con su imagen, nombre y cantidad.
 - **Calcula el precio total del producto** (`precio * cantidad`).
-

🚚 Información de envío y total

```

<div className="mt-4 pt-4 border-t flex justify-between items-center">
  <div>
    <p className="text-gray-500">Enviado a:</p>
    <p className="font-medium">{order.shipping.fullName}</p>
    <p className="text-sm text-gray-500">
      {order.shipping.address}, {order.shipping.city}
    </p>
  </div>
  <div className="text-right">
    <p className="text-gray-500">Total:</p>
    <p className="text-2xl font-bold text-primary">
      ${order.total.toFixed(2)}
    </p>
  </div>
</div>
```

- **Muestra el nombre y dirección de envío.**
 - **Calcula y muestra el total del pedido.**
-

⬅ Exportación del componente

```
export default MyOrders
```

- **Permite importar `MyOrders` en otros módulos.**

```
```markdown
MyAccount/index.jsx - Página de perfil del usuario

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

- **useState**: Manejo del estado de edición y formulario.
 - **motion de framer-motion**: Animaciones en los botones y el formulario.
 - **useAuth**: Accede a la información del usuario ([AuthContext](#)).
 - **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 || ''
})
```

- **user**: Contiene la información del usuario autenticado.
 - **isEditing**: Controla si el usuario está editando sus datos (**true** o **false**).
 - **formData**:
 - Inicializa con los datos del usuario autenticado.
 - Si algún campo no está definido, se inicializa como una cadena vacía ('').
-

📝 Manejo del formulario

📌 Actualizar información del usuario

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

- **Actualiza los datos del usuario al hacer clic en "Guardar Cambios".**
 - **Llama a `login()` con los nuevos datos** (simulación de actualización).
 - **Desactiva el modo edición (`setIsEditing(false)`).**
-

Renderizado principal

```
return (
  <Layout>
    <div className="min-h-screen bg-gray-50 dark:bg-gray-900 py-12 px-4 sm:px-6 lg:px-8">
      <div className="max-w-3xl mx-auto">
        <motion.div
          initial={{ opacity: 0, y: 20 }}
          animate={{ opacity: 1, y: 0 }}
          className="bg-white dark:bg-gray-800 p-8 rounded-xl shadow-lg"
        >
```

- **Usa `Layout` para la estructura base.**
 - **Aplica una animación de entrada (`y: 20 → 0`).**
-

Botón de edición

```
<div className="flex justify-between items-center mb-8">
  <h2 className="text-3xl font-bold text-gray-900 dark:text-white">Mi Cuenta</h2>
  <motion.button
    whileHover={{ scale: 1.05 }}
    whileTap={{ scale: 0.95 }}
    onClick={() => setIsEditing(!isEditing)}
    className="px-4 py-2 text-white bg-primary hover:bg-secondary rounded-xl"
  >
    {isEditing ? 'Cancelar' : 'Editar'}
  </motion.button>
</div>
```

- **Cambia entre modo edición y vista normal.**
 - **Texto dinámico:**
 - Si `isEditing === true`, muestra "Cancelar".
 - Si `isEditing === false`, muestra "Editar".
 - **Animación al hacer clic (`scale: 1.05`).**
-

Formulario editable

```
<form onSubmit={handleSubmit} className="space-y-6">
```

- **Cuando se envía (onSubmit)**, actualiza los datos del usuario.

✉ Campo Nombre

```
<div>
  <label className="block text-sm font-medium text-gray-700 dark:text-gray-300">
    Nombre
  </label>
  {isEditing ? (
    <input
      type="text"
      value={formData.name}
      onChange={(e) => setFormData({...formData, name: e.target.value})}
      className="mt-1 block w-full px-3 py-2 border border-gray-300 dark:border-gray-600 rounded-lg bg-white dark:bg-gray-700 text-gray-900 dark:text-white"
    />
  ) : (
    <p className="mt-1 text-gray-900 dark:text-white">{formData.name}</p>
  )}
</div>
```

- Si `isEditing === true`, se muestra un `input` para modificar el nombre.
- Si `isEditing === false`, se muestra el nombre como texto (`<p>`)**.

✉ Campo Email

```
<div>
  <label className="block text-sm font-medium text-gray-700 dark:text-gray-300">
    Email
  </label>
  {isEditing ? (
    <input
      type="email"
      value={formData.email}
      onChange={(e) => setFormData({...formData, email: e.target.value})}
      className="mt-1 block w-full px-3 py-2 border border-gray-300 dark:border-gray-600 rounded-lg bg-white dark:bg-gray-700 text-gray-900 dark:text-white"
    />
  ) : (
    <p className="mt-1 text-gray-900 dark:text-white">{formData.email}</p>
  )}
</div>
```

- Permite modificar el email solo en modo edición.

🏠 Campo Dirección

```
<div>
  <label className="block text-sm font-medium text-gray-700 dark:text-gray-300">
    Dirección
  </label>
  {isEditing ? (
    <input
      type="text"
      value={formData.address}
      onChange={(e) => setFormData({...formData, address: e.target.value})}
      className="mt-1 block w-full px-3 py-2 border border-gray-300 dark:border-gray-600 rounded-lg bg-white dark:bg-gray-700 text-gray-900 dark:text-white"
    />
  ) : (
    <p className="mt-1 text-gray-900 dark:text-white">{formData.address || 'No especificada'}</p>
  )}
</div>
```

- Si la dirección no está definida, muestra "No especificada".

📞 Campo Teléfono

```
<div>
  <label className="block text-sm font-medium text-gray-700 dark:text-gray-300">
    Teléfono
  </label>
  {isEditing ? (
    <input
      type="tel"
      value={formData.phone}
      onChange={(e) => setFormData({...formData, phone: e.target.value})}
      className="mt-1 block w-full px-3 py-2 border border-gray-300 dark:border-gray-600 rounded-lg bg-white dark:bg-gray-700 text-gray-900 dark:text-white"
    />
  ) : (
    <p className="mt-1 text-gray-900 dark:text-white">{formData.phone || 'No especificado'}</p>
  )}
</div>
```

- Si el teléfono no está definido, muestra "No especificado".

💾 Botón "Guardar Cambios"

```
{isEditing && (
  <motion.button
    whileHover={{ scale: 1.02 }}
    whileTap={{ scale: 0.98 }}
    type="submit"
    className="w-full py-3 text-white transition-colors bg-primary hover:bg-secondary rounded-xl"
  >
    Guardar Cambios
  </motion.button>
)}
```

- Solo aparece si `isEditing === true`.
- Guarda los cambios al enviarse el formulario.

⬅ END Exportación del componente

```
export default MyAccount
```

- Permite importar `MyAccount` en otros módulos.