

UNIVERSIDAD NACIONAL AUTÓNOMA DE NICARAGUA
UNAN-LEÓN

FACULTAD DE CIENCIAS Y TECNOLOGÍA
DEPARTAMENTO DE COMPUTACIÓN.



CARRERA: Ingeniería en Sistemas de Información.

GRUPO: N° 2.

COMPONENTE: Programación Orientada a la web II.

TEMA: Documento del proyecto.

ELABORADO POR: Br. Ever Josué Ochoa Coronado.

Tutor: Juan Carlos Leyton Briones.

FECHA: martes, 26 de septiembre de 2023.

“A la libertad por la universidad”

Objetivos del Proyecto

El proyecto se propone alcanzar los siguientes objetivos:

Desarrollo de la Aplicación Web SPA de Tienda de Productos Online: El objetivo central es crear una aplicación web altamente funcional que permita a los usuarios explorar, buscar y adquirir productos en línea de manera intuitiva y eficaz.

Uso de React.js, MySQL Server y JavaScript: Como tecnologías principales, React.js se empleará para el front-end, MySQL Server gestionará la persistencia de datos y JavaScript será el lenguaje de programación que conectará ambas partes de la aplicación.

Implementación de Funcionalidades Básicas de una Tienda en Línea: Esto incluye la gestión de productos, el carrito de compras y un proceso de compra seguro y eficiente.

Desarrollo de Funcionalidades Adicionales para una Mejor Experiencia del Usuario: Se incluirán características adicionales como filtros de búsqueda, listas de deseos, comentarios de usuarios y una función de red social para compartir productos y conectar con otros usuarios.

Requisitos de la Aplicación

Requisitos Funcionales:

Navegación por el Catálogo de Productos: Los usuarios podrán explorar el catálogo de productos, presentado en una lista o en un formato de cuadrícula para facilitar la búsqueda y selección.

Añadir Productos al Carrito de Compra: Los usuarios tendrán la capacidad de agregar productos a su carrito de compras, especificando la cantidad deseada de cada artículo.

Proceso de Compra Completo: Se permitirá a los usuarios llevar a cabo todo el proceso de compra, incluyendo la selección de productos, el pago y la confirmación del pedido.

Requisitos No Funcionales:

Responsividad: La aplicación será completamente responsiva, garantizando que los usuarios puedan disfrutar de la experiencia de compra en línea en dispositivos móviles y de escritorio.

Seguridad: Se implementarán medidas de seguridad robustas para proteger la información sensible de los usuarios, como datos de pago y detalles personales.

Introducción

El comercio electrónico ha emergido como uno de los sectores de más rápido crecimiento en la economía global en los últimos años. Las tiendas en línea se han convertido en una parte integral de la vida cotidiana, brindando a los consumidores la comodidad de comprar una amplia gama de productos desde cualquier lugar y en cualquier momento. Esta transformación en la forma en que compramos y vendemos ha sido impulsada en gran medida por avances tecnológicos y cambios en el comportamiento del consumidor.

La creación de una aplicación web de una sola página (SPA) dedicada a una tienda de productos en línea es una respuesta directa a esta revolución en el comercio. La creciente demanda de experiencias de compra en línea más rápidas, intuitivas y atractivas ha llevado a la evolución de las aplicaciones web hacia las SPAs. Estas aplicaciones brindan a los usuarios la capacidad de explorar productos, agregarlos al carrito de compras y completar sus pedidos de manera fluida y sin las interrupciones típicas de las páginas web tradicionales. Este enfoque de una sola página minimiza la necesidad de recargar páginas completas, lo que se traduce en una experiencia más ágil y eficiente.

El proyecto que se describe en este documento busca aprovechar esta creciente tendencia en el comercio electrónico y satisfacer las expectativas cambiantes de los usuarios. Al adoptar tecnologías clave como React.js, MySQL Server y JavaScript, estoy equipado para desarrollar una aplicación web SPA que no solo cumple con los requisitos básicos de una tienda en línea, sino que también brinda un nivel superior de interacción y facilidad de uso.

Desarrollo de la Aplicación

La aplicación se divide en dos componentes esenciales:

Backend:

El backend se desarrollará utilizando Javascript y se encargará de administrar los datos y la comunicación con el frontend. MySQL Server servirá como la base de datos para almacenar y recuperar información relacionada con productos, pedidos y clientes.

Frontend:

El frontend se construirá utilizando React.js y se centrará en presentar la información de manera atractiva y funcional a los usuarios. React.js permitirá la creación de componentes reutilizables para garantizar una experiencia de usuario dinámica y eficaz.

Implementación del Proyecto

La implementación del proyecto seguirá estas fases clave:

Creación de la Base de Datos: Se establecerá una base de datos MySQL que almacenará y gestionará todos los datos relacionados con la aplicación, como detalles de productos, historiales de pedidos y perfiles de usuarios.

Desarrollo del Backend: Se programará el backend de la aplicación para manejar las operaciones de gestión de datos y la comunicación con el frontend.

Desarrollo del Frontend: Se creará el frontend utilizando React.js para proporcionar a los usuarios una interfaz amigable y receptiva.

Integración del Backend y Frontend: Se llevará a cabo la integración de ambas partes de la aplicación para garantizar su funcionamiento eficaz.

Pruebas

Se realizarán pruebas exhaustivas durante el proceso de desarrollo para asegurar la calidad y funcionalidad de la aplicación:

Pruebas Unitarias: Se llevarán a cabo pruebas unitarias para verificar que cada componente de la aplicación funcione correctamente de manera individual.

Pruebas de Integración: Se realizarán pruebas de integración para confirmar que los diversos componentes de la aplicación interactúan de manera efectiva.

Pruebas de Rendimiento: Se ejecutarán pruebas de rendimiento para garantizar que la aplicación funcione sin problemas bajo cargas de usuarios.

Despliegue

La aplicación final se desplegará en un servidor web para que esté disponible para los usuarios. Se implementarán las medidas de seguridad necesarias para proteger tanto los datos de la aplicación como la experiencia del usuario.

Conclusiones

En la conclusión de este proyecto, es fundamental resaltar la importancia y el potencial de la Aplicación Web SPA de Tienda en Línea desarrollada utilizando React.js, MySQL Server y JavaScript como tecnologías principales. Este proyecto no solo busca satisfacer las necesidades cambiantes de los consumidores en el ámbito del comercio electrónico, sino que también demuestra el poder de las herramientas y enfoques tecnológicos modernos en la creación de experiencias de usuario excepcionales.

Uno de los aspectos más destacados de esta aplicación es su capacidad para brindar a los usuarios una experiencia de compra en línea fluida y eficiente. La adopción de una arquitectura de una sola página (SPA) permite a los clientes explorar productos, agregarlos al carrito de compras y completar sus pedidos de manera rápida y sin interrupciones. Esto no solo mejora la satisfacción del usuario, sino que también puede aumentar la conversión y las ventas, lo que es esencial en el competitivo mundo del comercio electrónico.

Además de las funcionalidades básicas de una tienda en línea, como la navegación por el catálogo de productos y el proceso de compra, esta aplicación ofrece características adicionales que enriquecen la experiencia del usuario. Los filtros de búsqueda, las listas de deseos y la capacidad de comentar productos promueven una interacción más significativa con los usuarios y fomentan la lealtad del cliente. La función de red social para compartir productos y conectar con otros usuarios agrega un componente social valioso a la experiencia de compra, lo que puede ayudar a construir una comunidad en torno a la marca.

Desde una perspectiva técnica, este proyecto ha demostrado cómo combinar y aprovechar React.js, MySQL Server y JavaScript para crear una aplicación web sofisticada y eficiente. La modularidad y la reutilización de componentes en React.js han facilitado el desarrollo y el mantenimiento de la interfaz de usuario. MySQL Server ha demostrado ser una elección sólida para la gestión de datos, permitiendo un almacenamiento seguro y eficiente de información crítica. Por último, JavaScript ha actuado como el pegamento que conecta el front-end y el back-end, garantizando una comunicación fluida y una experiencia de usuario cohesiva.

En última instancia, este proyecto demuestra cómo la tecnología puede potenciar y transformar el comercio electrónico, brindando a los usuarios una forma más efectiva y agradable de comprar en línea. La combinación de tecnologías modernas con una comprensión sólida de las necesidades del usuario ha resultado en una aplicación que no solo cumple con los estándares actuales, sino que también está preparada para adaptarse a las demandas futuras del mercado en constante evolución. En un mundo donde la experiencia del cliente es clave, esta Aplicación Web SPA de Tienda en Línea es un ejemplo destacado de cómo la tecnología puede impulsar el éxito en el mundo del comercio electrónico.

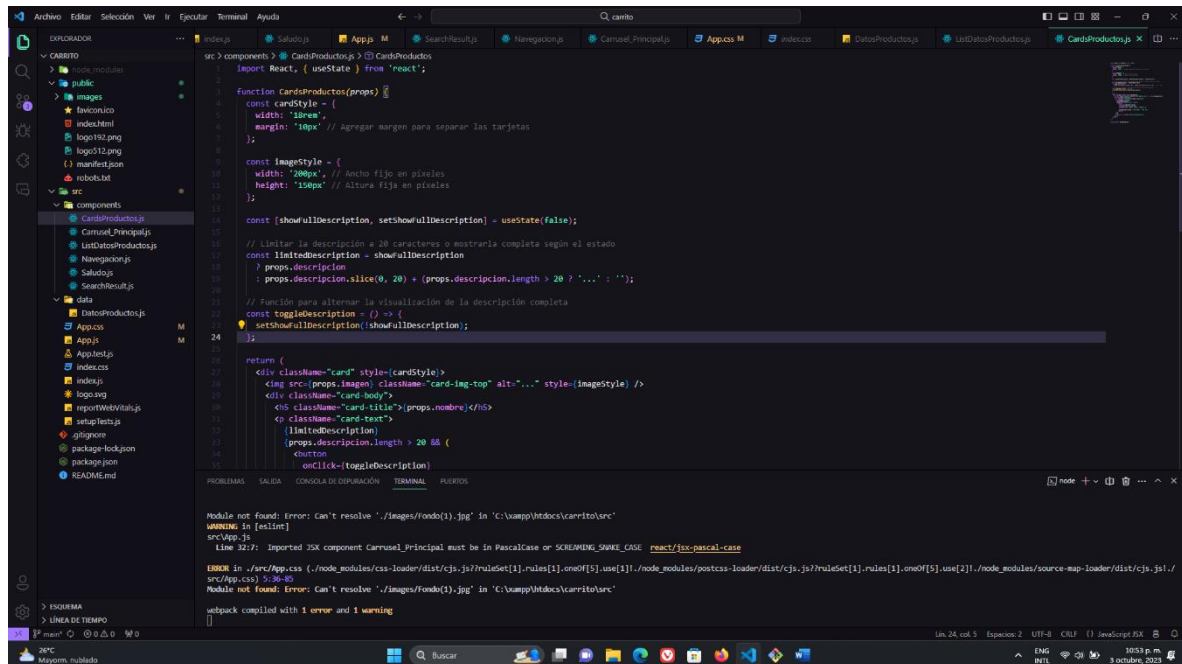
Referencias bibliográficas

React.js. (s.f.). Documentación oficial de React. <https://react.dev/blog/2023/03/16/introducing-react-dev>

MySQL. (s.f.). Documentación oficial de MySQL. <https://dev.mysql.com/doc/>

Node.js. (s.f.). Documentación oficial de Node.js. <https://nodejs.org/en/docs/>

Nuevas implementaciones: Componentes -> componente CardsProductos



```
src > components > CardsProductos.js < CardsProductos
import React, { useState } from 'react';

function CardsProductos(props) {
  const cardStyle = {
    width: '180px',
    margin: '10px' // Agregar margen para separar las tarjetas
  };

  const imageStyle = {
    width: '100px', // Ancho fijo en píxeles
    height: '150px' // Altura fija en píxeles
  };

  const [showFullDescription, setShowFullDescription] = useState(false);

  // Limitar la descripción a 20 caracteres o mostrarla completa según el estado
  const limitedDescription = showFullDescription
    ? props.description
    : props.description.slice(0, 20) + (props.description.length > 20 ? '...' : '');

  // Función para alternar la visualización de la descripción completa
  const toggleDescription = () => {
    setShowFullDescription(!showFullDescription);
  };

  return (
    <div className="card" style={cardStyle}>
      <img src={props.image} className="card-img-top" alt="..." style={imageStyle} />
      <div className="card-body">
        <h5 className="card-title">{props.nombre}</h5>
        <p className="card-text">
          {limitedDescription}
          {props.description.length > 20 && (
            <button
              onClick={toggleDescription}
            >

```

Module not found: Error: Can't resolve './images/fondo(1).jpg' in 'C:\xampp\htdocs\carrito\src'

WARNING in [eslint]

src\Views.js

Line 32:7: Imported JSX component Carrusel_Principal must be in PascalCase or SCREAMING_SNAKE_CASE react/jsx-pascal-case

ERROR in ./src/App.css (.node_modules/css-loader/dist/cjs.js?ruleset[1].rules[1].oneOf[5].use[2])/node_modules/source-map-loader/dist/cjs.js!./src/App.css 5:30-40

Module not found: Error: Can't resolve './images/fondo(1).jpg' in 'C:\xampp\htdocs\carrito\src'

webpack compiled with 1 error and 1 warning

El componente CardsProductos utiliza el estado local de React para mostrar la descripción completa o limitada de un producto. La descripción completa se muestra por defecto, pero se puede mostrar una versión limitada haciendo clic en el botón "Ver más".

El componente utiliza el estado local de React para controlar si se muestra la descripción completa o limitada del producto.

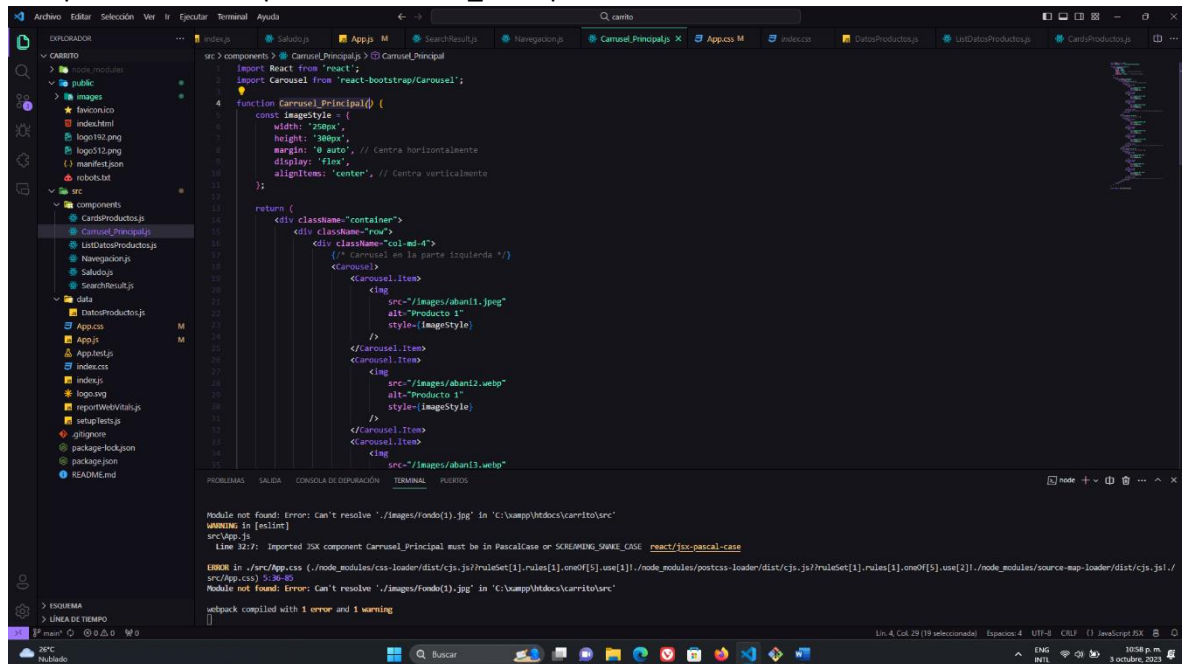
El componente define dos objetos de estilo CSS para aplicar a los elementos de la tarjeta de producto.

El componente utiliza una variable para almacenar la descripción limitada del producto.

El componente utiliza una función para alternar entre mostrar la descripción completa y limitada del producto.

El componente renderiza una tarjeta de Bootstrap con la estructura HTML típica de una tarjeta.

Componentes -> componente Carrusel_Principal



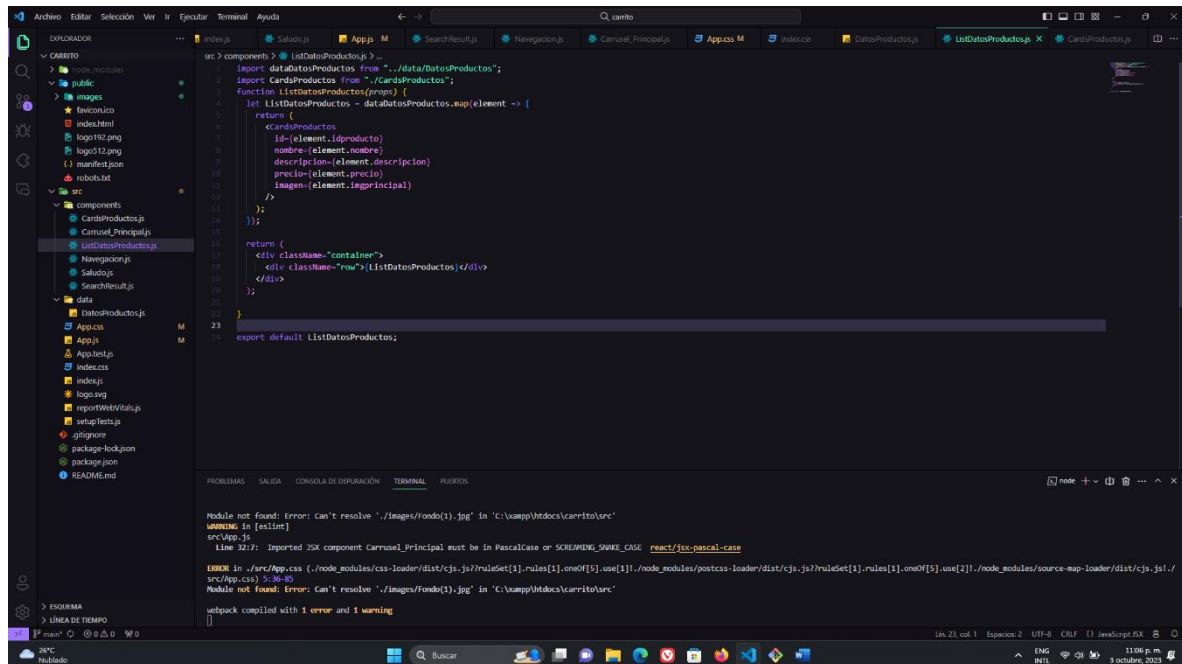
El componente Carrusel_Principal utiliza la librería Carousel de react-bootstrap para crear un carrusel de imágenes dividido en tres secciones. Cada sección contiene un conjunto de imágenes que se pueden desplazar horizontalmente.

El componente Carrusel_Principal importa la librería Carousel de react-bootstrap, que proporciona las funciones necesarias para crear un carrusel de imágenes. También define un objeto de estilo imageStyle que se aplica a las imágenes dentro del carrusel. Este estilo fija el ancho y la altura de las imágenes, los centra horizontal y verticalmente, y las alinea verticalmente en el centro.

El componente utiliza una estructura HTML para organizar el carrusel. Se divide en tres columnas (col-md-4) dentro de una fila (row) de una cuadrícula Bootstrap (container). Cada columna contiene su propio carrusel de imágenes creado con la etiqueta <Carousel>. Dentro de cada carrusel, se definen múltiples elementos <Carousel.Item>, donde cada uno representa una diapositiva del carrusel. Dentro de cada elemento <Carousel.Item>, se coloca una imagen con la etiqueta . Cada imagen tiene una fuente (src), un texto alternativo (alt) y se aplica el estilo imageStyle que se definió anteriormente.

Para personalizar el carrusel, se puede agregar más elementos al carrusel simplemente copiando y pegando más bloques de código <Carousel.Item> e imágenes dentro de cada carrusel individual. Esto te permite mostrar varias imágenes en cada sección del carrusel y personalizarlo mas.

Componentes -> componente ListDatosProductos



```
src > components > ListDatosProductos.js > ...
1 import datosProductos from "../data/DatosProductos";
2 import CardsProductos from "../CardsProductos";
3 function ListDatosProductos(props) {
4   let ListDatosProductos = datosProductos.map(element => {
5     return (
6       <CardsProductos
7         id={element.idproducto}
8         nombre={element.nombre}
9         descripcion={element.descripcion}
10        precio={element.precio}
11        imagen={element.imagenprincipal}
12      />
13    );
14  });
15  return (
16    <div className="container">
17      <div className="row">{ListDatosProductos}</div>
18    </div>
19  );
20 }
21
22 export default ListDatosProductos;
```

Module not found: Error: Can't resolve './images/fondo(1).jpg' in 'C:\xampp\htdocs\carrito\src'

WARNING in [eslint]

Line 32:7: Imported JSX component Carrusel_Principal must be in PascalCase or SCREAMING_SNAKE_CASE [react/jsx-pascal-case](#)

ERROR in ./src/App.css (.node_modules/css-loader/dist/cjs.js?ruleset[1].rules[1].oneOf[5].use[1]!./node_modules/postcss-loader/dist/cjs.js?ruleset[1].rules[1].oneOf[5].use[2]!./node_modules/source-map-loader/dist/cjs.js!./src/App.css) 5:30-40

Module not found: Error: Can't resolve './images/fondo(1).jpg' in 'C:\xampp\htdocs\carrito\src'

webpack compiled with 1 error and 1 warning

El componente ListDatosProductos muestra una lista de productos utilizando el componente CardsProductos. El componente importa un archivo de datos que contiene información sobre los productos, y luego crea un componente CardsProductos para cada producto en los datos. Estos componentes CardsProductos se organizan en filas de acuerdo con la cuadrícula de Bootstrap.

El componente importa un archivo de datos que contiene información sobre los productos.

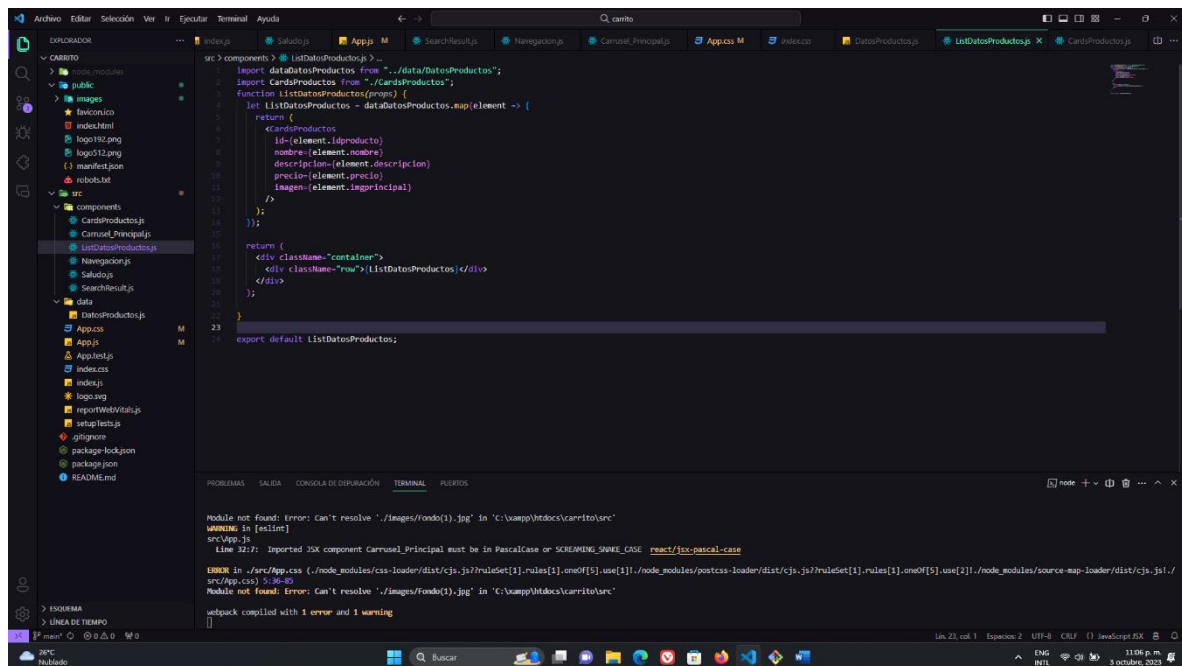
El componente utiliza el método map para crear un componente CardsProductos para cada producto en los datos.

Cada componente CardsProductos se crea con propiedades como id, nombre, descripcion, precio, y imagen que se extraen de los datos de productos correspondientes en el mapeo.

El componente ListDatosProductos devuelve una estructura HTML que utiliza Bootstrap para organizar los productos en una cuadrícula.

Los componentes CardsProductos se colocan en cada fila, uno por uno, de acuerdo con el mapeo de datos realizado anteriormente.

Componentes -> componente Navegación



El componente Navegacion es una barra de navegación Bootstrap con un campo de búsqueda. El componente utiliza el estado de React para manejar el término de búsqueda ingresado por el usuario. Cuando el usuario envía el formulario de búsqueda, los resultados coincidentes se muestran en el componente de resultados de búsqueda correspondiente.

El componente utiliza el estado de React para manejar dos valores: searchTerm y searchResults.

El evento onChange del campo de búsqueda está conectado a la función handleSearchChange, que actualiza el estado searchTerm con el valor actual del campo de búsqueda.

Cuando el usuario envía el formulario de búsqueda, se activa la función handleSearchSubmit, que realiza la lógica de búsqueda utilizando el término almacenado en searchTerm.

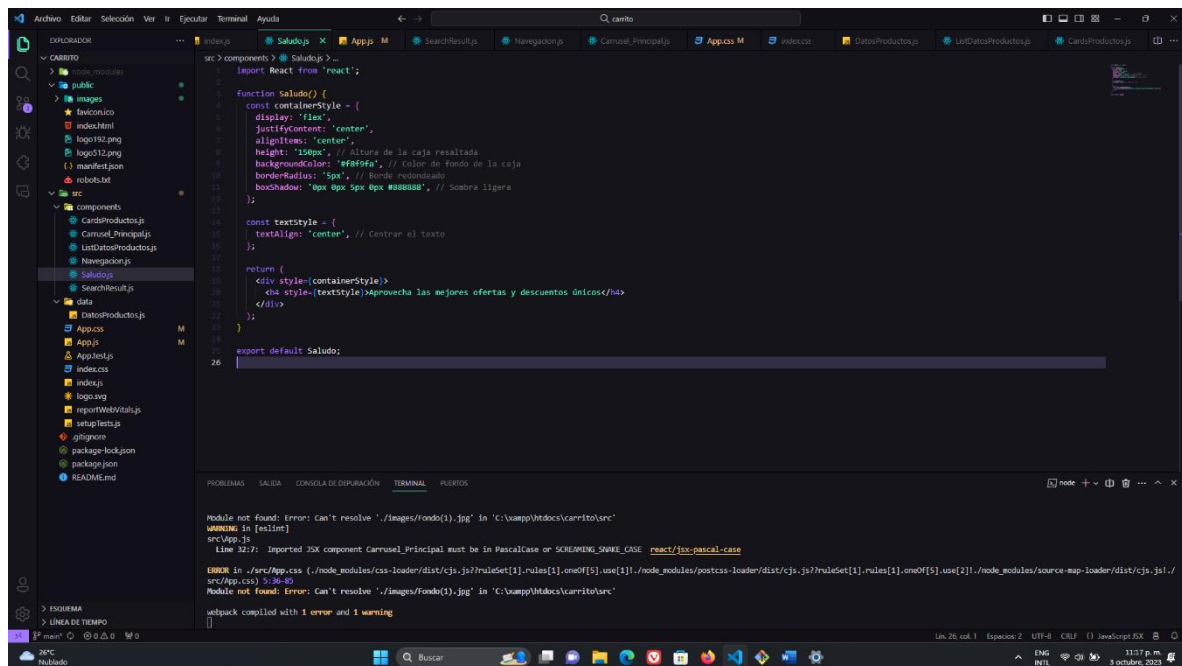
Los resultados coincidentes se almacenan en searchResults.

La barra de navegación se renderiza utilizando Bootstrap y contiene elementos como el logotipo, elementos de menú, un campo de búsqueda y un botón "Buscar".

El valor de searchTerm se establece como el valor del campo de búsqueda.

Cuando se envía el formulario de búsqueda, los resultados coincidentes se mostrarán en el componente de resultados de búsqueda correspondiente.

Componentes -> componente Saludo



```
src > components > Saludo.js > ...
1  import React from 'react';
2
3  function Saludo() {
4    const containerStyle = {
5      display: 'flex',
6      justify-content: 'center',
7      align-items: 'center',
8      height: '150px', // Altura de la caja resaltada
9      backgroundColor: 'ffff99', // Color de fondo de la caja
10     borderRadius: '5px', // Bordes redondeados
11     boxShadow: '0px 0px 5px 0px #000000', // Sombra ligera
12   };
13
14   const textStyle = {
15     textAlign: 'center', // Centrar el texto
16   };
17
18   return (
19     <div style={containerStyle}>
20       <h4 style={textStyle}>Aprovecha las mejores ofertas y descuentos únicos</h4>
21     </div>
22   );
23 }
24
25 export default Saludo;
26
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS

Module not found: Error: Can't resolve './images/fondo(1).jpg' in 'C:\xampp\htdocs\carrito\src'

WARNING in [eslint]

Line 32:7: Imported JSX component Carrusel_Principal must be in PascalCase or SCREAMING_SNAKE_CASE react/jsx-pascal-case

ERROR in ./src/App.css (.node_modules/css-loader/dist/cjs.js?ruleset[1].rules[1].oneOf[5].use[1]!./node_modules/postcss-loader/dist/cjs.js?ruleset[1].rules[1].oneOf[5].use[2]!./node_modules/source-map-loader/dist/cjs.js!./src/App.css) 5:30-40

Module not found: Error: Can't resolve './images/fondo(1).jpg' in 'C:\xampp\htdocs\carrito\src'

webpack compiled with 1 error and 1 warning

El componente Saludo muestra un mensaje de bienvenida en una caja resaltada. El componente utiliza estilos CSS en objetos JavaScript para dar estilo a la caja y al texto del mensaje de saludo. El mensaje de saludo predeterminado es "Aprovecha las mejores ofertas y descuentos únicos".

El componente define dos objetos JavaScript (containerStyle y textStyle) que contienen estilos CSS en línea para aplicar a elementos HTML dentro del componente.

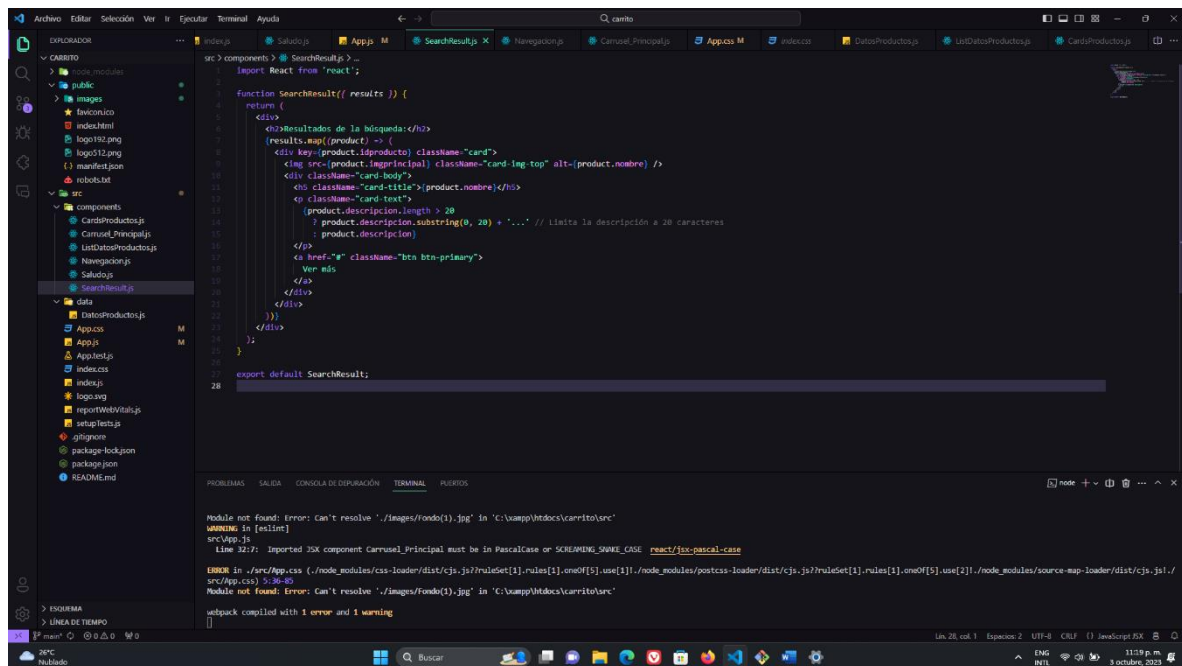
El componente renderiza un <div> que contiene el mensaje de saludo.

El estilo de este <div> se establece utilizando el objeto containerStyle para lograr un diseño centrado y resaltado.

Dentro del <div>, hay un <h4> que contiene el mensaje de saludo. El estilo de este <h4> se establece utilizando el objeto textStyle para centrar el texto horizontalmente.

El mensaje de saludo se encuentra dentro del <h4> y dice: "Aprovecha las mejores ofertas y descuentos únicos".

Componentes -> componente SearchResult



El componente SearchResult muestra los resultados de una búsqueda en una lista de tarjetas. El componente recibe un arreglo de objetos que representan los resultados de la búsqueda. Cada tarjeta de producto contiene una imagen, un título, una descripción y un botón "Ver más".

El componente recibe un prop llamado results, que es un arreglo de objetos que representan los resultados de la búsqueda.

El componente renderiza un encabezado <h2> que dice "Resultados de la búsqueda:" para indicar que se están mostrando resultados de búsqueda.

El componente itera a través del arreglo de resultados (results) utilizando el método map() para generar una tarjeta de producto para cada elemento del arreglo.

Cada tarjeta de producto se representa como un conjunto de elementos HTML, incluyendo una imagen, título, descripción y un botón.

La imagen de la tarjeta de producto se toma del atributo imgPrincipal del objeto de producto.

El título de la tarjeta de producto se toma del atributo nombre del objeto de producto.

La descripción de la tarjeta de producto se toma del atributo descripcion del objeto de producto.