EVIDENCIA: DISEÑO Y DESARROLLO DE SERVICIOS WEB - PROYECTO

GA7-220501096-AA5-EV03

INSTRUCTOR:

YERMAN AUGUSTO HERNANDEZ SAENZ

ESTUDIANTE:

JOHAN GUTIERREZ

EDILSON GRANADOS

FICHA DE MATRICULA:

2627061

ANÁLISIS Y DESARROLLO DE SOFTWARE

SERVICIO NACIONAL DE APRENDIZAJE SENA CENTRO DE SERVICIOS FINANCIEROS

BOGOTÁ DISTRITO CAPITAL

2023

Introducción

En el siguiente documento se visualizará la documentación de los servicios para el proyecto formativo GreenBox el cual podremos observar los servicios, como conexión entre páginas, carrito de compras y registro y login de usuario.

DOCUMENTACION DE SERVICION WEB DEL PROYECTO GREENBOX

GreenBox es una plataforma de compra, venta, transporte de productos agrícolas que utiliza framework React Vite para su desarrollo el diseño es fundamental es permitir la conexión entre usuario y vendedora sobre la venta y compra de productos agrícolas el cual ofrecerá los siguientes servicios.

Conexión entre páginas: Greenbox utiliza React Router para conectar las páginas privadas y públicas del sitio web. Las páginas privadas están restringidas a los usuarios registrados, mientras que las páginas públicas están disponibles para todos los usuarios.

Login-Inicio Sesión: Greenbox permite a los usuarios registrarse y iniciar sesión en el sitio web. El registro se realiza mediante un formulario simple que solicita el nombre, la dirección de correo electrónico y la contraseña del usuario.

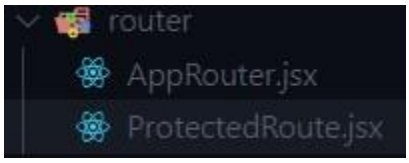
El desarrollo de Greenbox se dividió en las siguientes etapas:

1. Carrito de compras: Greenbox permite a los usuarios agregar productos al carrito de compras y realizar compras. El carrito de compras almacena una lista de los productos que el usuario ha seleccionado y el precio total de la compra.
2. Instalación de dependencias: Se instalaron las dependencias necesarias para el desarrollo de Greenbox, incluyendo React, React Router y Vite.
3. Creación de la estructura del proyecto: Se creó la estructura del proyecto, incluyendo las carpetas para los archivos de código, los estilos y los recursos.
4. Implementación de las características: Se implementaron las características de Greenbox, incluyendo la conexión entre páginas, el login y el carrito de compras.
5. Pruebas: Se realizaron pruebas para garantizar que las características de Greenbox funcionaran correctamente.

Conexión entre paginas

Para conexión de las paginas del proyecto se utilizó la biblioteca React Router la cual nos permite el desarrollo de la navegación entre paginas el cual permite determinar rutas públicas y privadas.

- Se realiza la creación de la siguiente carpeta y archivos.



- Para no presentar fallo en la hora de la conexión de las rutas se instala la biblioteca: **npm install react-router-dom** la cual nos permite realizar la navegación entre páginas.
- En nuestro Archivo ya creado por la biblioteca React Vite archivo: main.jsx realizamos las modificaciones necesarias y el código queda de la siguiente manera.

```jsx
import React from 'react'
import ReactDOM from 'react-dom/client'
import App from './App.jsx'
import './index.css'
import { BrowserRouter } from 'react-router-dom'

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

  </React.StrictMode>,
)
```

Donde **BrowserRouter** nos permite la navegación entre paginas dentro de nuestra web.

- El archivo AppRouter.jsx realizamos la codificación y las importaciones para la navegación entre las diferentes páginas.

```jsx
import { Route, Routes } from "react-router-dom"
import { Navbar } from "../components/Navbar"
import { Dashboard, Home, Login, Lugares, Productos, Registro,Perfil, Ayuda,
Categorias,
Comprar, Frutas, Vegetales, Cereales, Legumbres, Especias, Calendario, Venta
} from "../pages"
import { ProtectedRoute } from "./ProtectedRoute"

export const AppRouter = () => {
  return (
     <>
     <Routes>
     <Route path="/" element = {<Home/>} />
      <Route path="home" element = {<Home/>} />
      <Route path="/" element={<Navbar/>} >
      <Route path="login" element ={<Login/>}/>
      <Route path="registro" element ={<Registro/>}/>
      <Route path="dashboard" element ={
        <ProtectedRoute>
          <Dashboard />
        </ProtectedRoute>
      }/>
      </Route>

      <Route path="productos" element={<Productos />}/>
```

```
      <Route path="lugares" element={<Lugares/>}/>
      <Route path="categorias" element={<Categorias/>}/>
      <Route path="perfil" element={<Perfil/>}/>
      <Route path="ayuda" element={<Ayuda/>}/>
      <Route path="comprar" element={<Comprar/>}/>
      <Route path="frutas" element={<Frutas/>}/>
      <Route path="vegetales" element={<Vegetales/>}/>
      <Route path="cereales" element={<Cereales/>}/>
      <Route path="legumbres" element={<Legumbres/>}/>
      <Route path="especias" element={<Especias/>}/>
      <Route path="calendario" element={<Calendario/>}/>
      <Route path="venta" element={<Venta/>}/>
     </Routes>
     </>
  )
}
```

Se realiza las conexiones necesarias para la navegación de nuestro proyecto el cual con la ayuda de la biblioteca **react-router-dom** nos permite la navegación correcta el cual esa biblioteca nos da los siguientes archivos Routes el cual encierra las rutas donde se van a navegar y Route es el archivo de la biblioteca router-dom que nos permite agregar cada una de las paginas que necesitamos en el proyecto GreenBox.

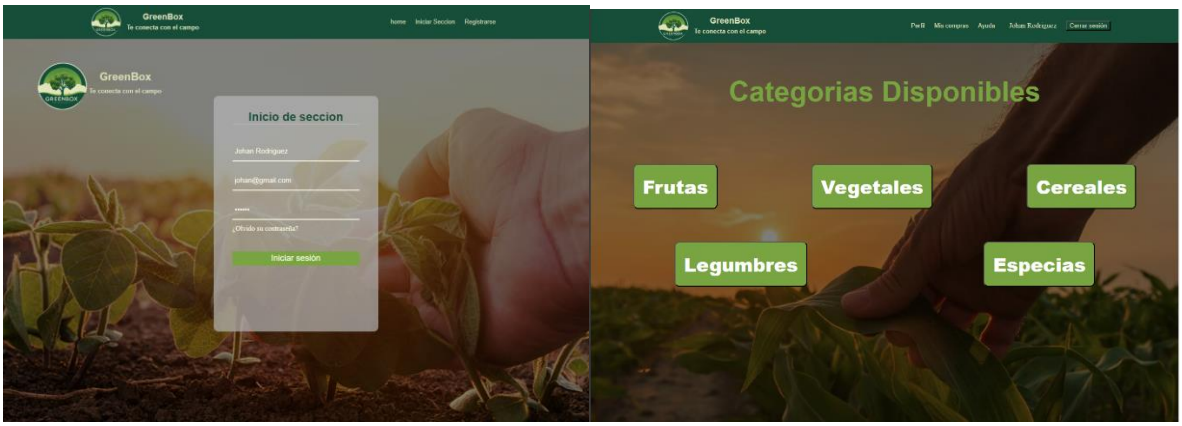
El siguiente archivo:

```
        <ProtectedRoute>
          <Dashboard />
        </ProtectedRoute>
```

Nos permite colocar nuestra ruta privada Dashboard la cual se realiza un archivo privado llamado ProtectedRoute el cual contiene el siguiente código.

```
import { Navigate, useLocation } from "react-router-dom"

export const ProtectedRoute = ({children}) => {

    const {state} = useLocation()

  return state?.logged ? children : <Navigate to='/login'/>;

}
```

- En el archivo ProtectedRoute podemopver los diferente archivos Navigate que nos permite la navegación entre página y cuando el archivo no este logiado nos dejara en la página login.



En las imágenes evidenciadas anteriormente podemos ver la utilización de las rutas react-router-dom que nos permite la navegación en la primera podemos evidenciar el login y cuando se logea el usuario entra una pagina privada que nos muestra el nombre del usuario logiado y botón de cerrar sesión para volver a login.

LOGIN Y REGISTRO DE USUARIO

El login y el registro de usuario son dos características fundamentales decual cualquier desarrollo web el cual se utilizaron diferentes bibliotecas como Node.js, MySQL y Express.

Este método recibe los datos del usuario, como el nombre de usuario y la contraseña, como parámetros. Luego, la aplicación verifica los datos del usuario en la base de datos. Si los datos son correctos, la aplicación devuelve un token de autorización. El token de autorización se utiliza para identificar al usuario en futuras solicitudes a la API.

El siguiente código evidenciaremos las conexiones y bibliotecas que nos permitirá utilizar nuestro desarrollo del login y registro de usuario con diferentes advertencias.

```
const express = require('express');
const { engine } = require('express-handlebars');
const myconnection = require('express-myconnection');
const mysql = require('mysql');
const session = require('express-session');
const bodyParser = require('body-parser');
const loginRoutes = require('./routes/login');

const app = express();
app.set('port', 5000)

app.set('views', __dirname + '/views');
app.engine('.hbs', engine({
    extname: '.hbs',
}));
app.set('view engine', 'hbs');
```

```
app.use(bodyParser.urlencoded({
    extended: true
}));

app.use(bodyParser.json());

app.use(myconnection(mysql,  {
    host: 'localhost',
    user: 'root',
    password: '',
    port: 3306,
    database: 'nodelogin'
}))

app.use(session({
    secret: 'secret',
    resave: true,
    saveUninitialized: true
}));

app.listen(app.get('port'), () => {
    console.log('Listening on port', app.get('port'));
})

app.use('/', loginRoutes);

app.get('/', (req, res) => {
    if(req.session.loggedin == true){
        res.render('home', {name: req.session.name});
    }else{
        res.redirect('/login');
    }
});
```

EN el código podemos evidenciar el puerto a utilizar la conexión de las rutas para la navegación del login y registro de usuario la conexión de la base de datos con mysql la base de datos creada llamada nodelogin el cual prosearemos los diferentes registros de los usuarios.

```
const express = require('express');
const LoginController = require ('../controllers/LoginController');
const router = express.Router();

router.get('/login', LoginController.login);
```

```
router.post('/login', LoginController.auth);
router.get('/register', LoginController.register);
router.post('/register', LoginController.storeUser);
router.get('/logout', LoginController.logout);


module.exports = router;
```

podemos observar los diferentes enrutamientos de nuestro registro y login.

```
const bcrypt = require('bcrypt')

function login(req, res) {
    if(req.session.loggedin != true){
        res.render('login/index');
    }else{
        res.redirect('/');
    }

}

function auth(req, res) {
    const data = req.body;
    req.getConnection((err, conn) =>{
        conn.query('SELECT * FROM  users WHERE email = ?',
[data.email],(err, userdata) =>{
            if(userdata.length > 0){
                userdata.forEach(element =>{
                    bcrypt.compare(data.password, element.password, (err,
isMatch) =>{

                            if(!isMatch){
                                res.render('login/index', {error: 'Error:
Contraseña incorrecta'});
                            }else{
                                req.session.loggedin = true;
                                req.session.name = element.name;

                                res.redirect('/');
                            }

                        });

                    })
            }else{
```

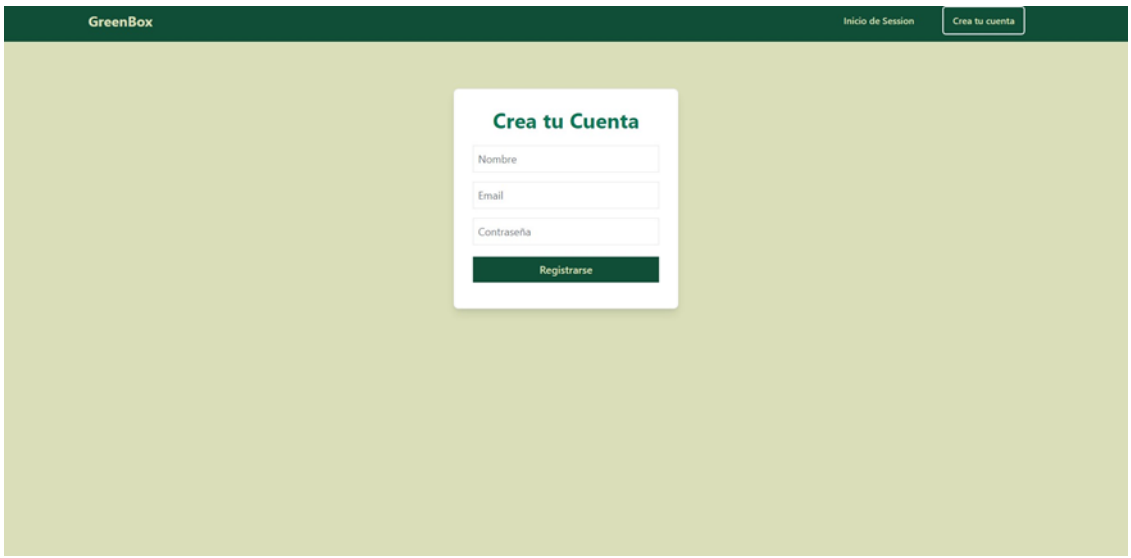
```
                res.render('login/index', {error: 'Error: El usuario no
existe'})
}
});
});
}

function register (req, res){
    if(req.session.loggedin != true){
        res.render('login/register');
    }else{
        res.redirect('/');
    }
}

function storeUser(req, res) {
    const data = req.body;

    req.getConnection((err, conn) =>{
        conn.query('SELECT * FROM  users WHERE email = ?',
[data.email],(err, userdata) =>{
            if(userdata.length > 0){
                res.render('login/register', {error: 'Error: usuario ya se
encuentra registrado'})
            }else{
                bcrypt.hash(data.password, 12).then(hash => {
                    data.password = hash;

                    req.getConnection((err, conn) =>{
                        conn.query('INSERT INTO users SET ?', [data], (err,
rows) =>{



                            res.redirect('/')
                        });
                    });
                });
            }
        });
    })


}
function logout(req, res){
```

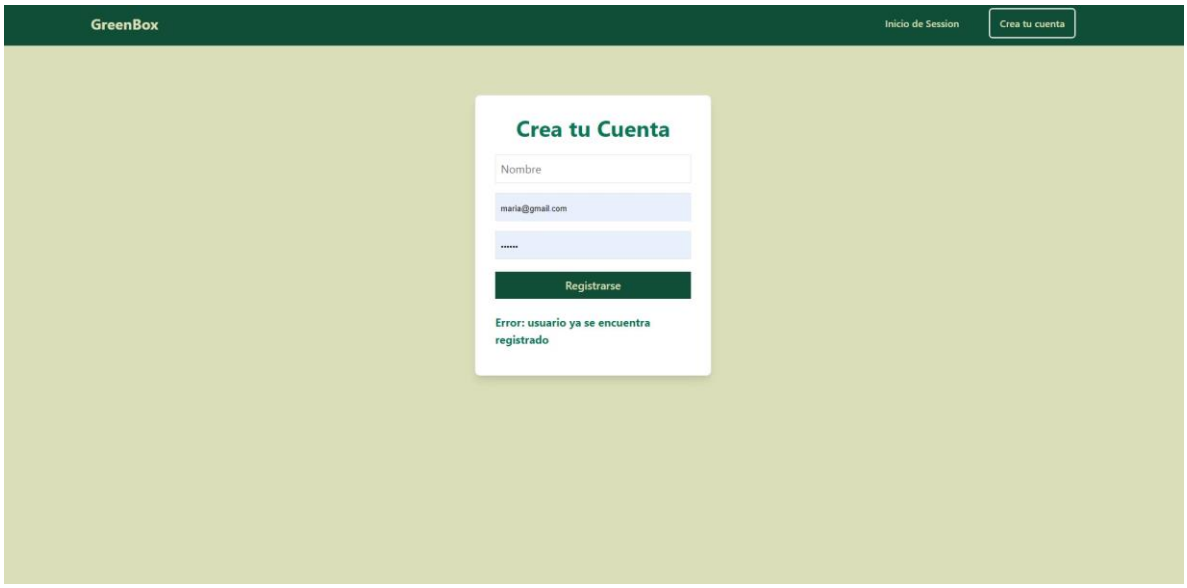
```
    if(req.session.loggedin == true) {

        req.session.destroy();

    }
        res.redirect('/login');
}


module.exports = {
    login,
    register,
    storeUser,
    auth,
    logout,
}
```

Al iniciar sesión o registrarse, el sistema muestra diferentes mensajes de alerta en función del estado del usuario en la base de datos.

Registro de usuario

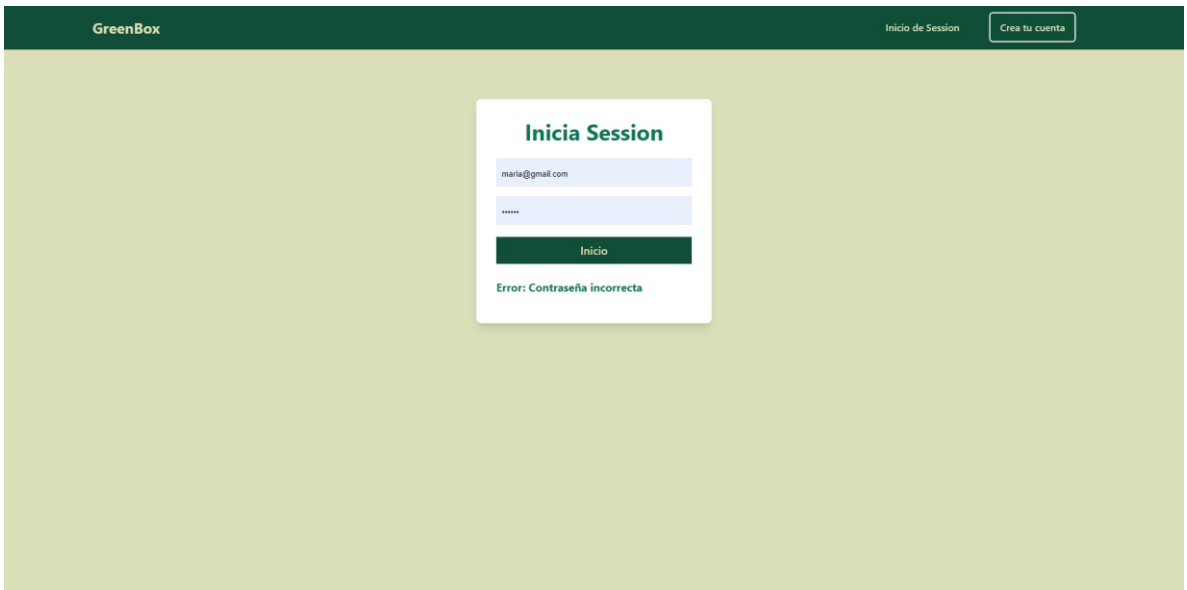
- Si el usuario no está registrado, se le permite completar el formulario de registro y enviar sus datos a la base de datos.
- Si el usuario ya está registrado, se le muestra un mensaje de alerta informándole de ello.

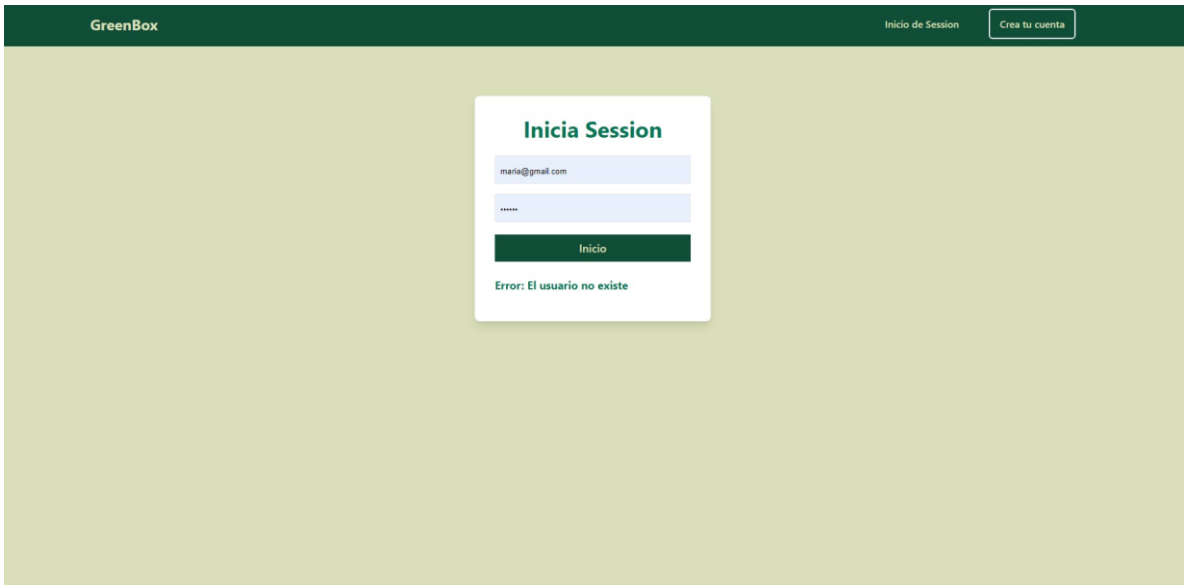




Inicio de sesión

- Si el usuario está registrado, se le permite iniciar sesión con su correo electrónico y contraseña.
- Si el usuario no está registrado, se le muestra un mensaje de alerta informándole de ello.
- Si la contraseña del usuario es incorrecta, se le muestra un mensaje de alerta informándole de ello.





Página de bienvenida

Una vez que el usuario se ha registrado correctamente e iniciado sesión, se le redirige a una página de bienvenida. Esta página muestra el nombre del usuario registrado y un botón para cerrar sesión.

Ejemplo

Registro de usuario

Para registrarte, completa el siguiente formulario:

- Nombre
- Correo electrónico
- Contraseña

Haz clic en "Registrar" para enviar tus datos.

Inicio de sesión

Para iniciar sesión, introduce tu correo electrónico y contraseña:

- Correo electrónico
- Contraseña

Haz clic en "Iniciar sesión" para acceder a tu cuenta.

Página de bienvenida

Bienvenido, [nombre de usuario].

Haz clic en "Cerrar sesión" para salir de tu cuenta.

Carrito de compras

Agregar productos:

Para agregar un producto al carrito de compras, haz clic en el botón "Agregar al carrito" que aparece junto al producto que deseas comprar. El producto se agregará a una lista en el carrito de compras y su precio se sumará al precio total del carrito.

Eliminar productos:

Para eliminar un producto del carrito de compras, haz clic en el botón "Eliminar" que aparece junto al producto que deseas eliminar. El producto se eliminará de la lista del carrito de compras.

Realizar la compra:

Cuando estés listo para realizar la compra, haz clic en el botón "Comprar" que aparece en el carrito de compras. Esto te redirigirá a una página donde podrás completar la información de envío y pago.

```
import React, { useState } from 'react';
import { Navegacion } from '../components/Navegacion';
import { Link } from 'react-router-dom';
import '../components/css/ProductosGeneral.css';

import img11 from '../img/frutas/Banano/image3.jpg';
import img12 from '../img/frutas/Manzana/image1.jpg';
import img13 from '../img/frutas/Zapote/image1.jpg';
import img14 from '../img/frutas/Pera/image3.jpg';

export const Frutas = () => {
```

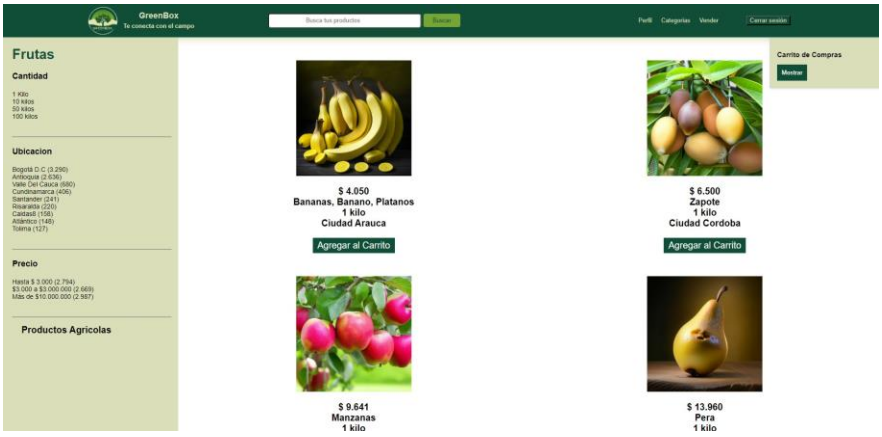
```
const [carritoVisible, setCarritoVisible] = useState(false);
const [carrito, setCarrito] = useState([]);

const agregarAlCarrito = (producto) => {
  setCarrito([...carrito, producto]);
};

const eliminarDelCarrito = (index) => {
  const nuevoCarrito = [...carrito];
  nuevoCarrito.splice(index, 1);
  setCarrito(nuevoCarrito);
};

const calcularTotal = () => {
  return carrito.reduce((total, item) => total + item.precio, 0);
};

return (
  <div>
    <div>
      <Navegacion />
    </div>
    <div className="contenido-frutas">
      <div className="textoEncabezadoIzquierdo">
        <div className="color30">
          <h1>Frutas</h1>
          <div className="culm">
            <p>Cantidad</p>
          </div>
          <div className="culm1">
            <p>1 Kilo <br />
              10 kilos <br />
              50 kilos <br />
              100 kilos</p>
            <div className="linea">
              <p>_____</p>
            </div>
          </div>
          <div className="culm">
            <p>Ubicacion</p>
          </div>
          <div className="culm1">
            <p>Bogotá D.C (3.290) <br />
              Antioquia (2.636) <br />
              Valle Del Cauca (680) <br />
```

```
              Cundinamarca (406) <br />
              Santander (241) <br />
              Risaralda (220) <br />
              Caldas8 (158) <br />
              Atlántico (148) <br />
              Tolima (127)</p>
            <div className="linea">
              <p>_____</p>
            </div>
          </div>
          <div className="culm">
            <p>Precio</p>
          </div>
          <div className="culm1">
            <p>Hasta $ 3.000 (2.794) <br />
              $3.000 a $3.000.000 (2.669) <br />
              Más de $10.000.000 (2.987)</p>
            <div className="linea">
              <p>_____</p>
            </div>
          </div>
          <div className="culm">
            <Link to="/categorias">Productos Agricolas</Link>
          </div>
        </div>
      </div>
      <div className="img1">
        <div className="contenidoImagenes ">
          <div className="img-frutas">
            <img src={img11} alt="Banana" />
            <p>$ 4.050 <br />
              Bananas, Banano, Platanos <br />
              1 kilo <br />
              Ciudad Arauca</p>
            <button onClick={() => agregarAlCarrito({ nombre: 'Bananas',
precio: 4050, cantidad: 1 })}>
              Agregar al Carrito
            </button>
          </div>

          <div className="img-frutas">
            <img src={img12} alt="Manzana" />
            <p>$ 9.641 <br />
              Manzanas <br />
              1 kilo <br />
```

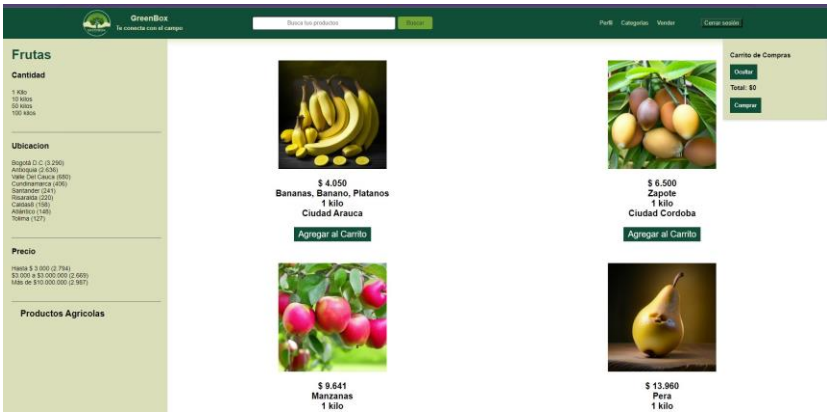
```
              Ciudad Antioquia</p>
            <button onClick={() => agregarAlCarrito({ nombre: 'Manzanas',
precio: 9641, cantidad: 1 })}>
              Agregar al Carrito
            </button>
          </div>
        </div>
      </div>
      <div className="img2">
        <div className="contenidoImagenes ">
          <div className="img-frutas">
            <img src={img13} alt="Zapote" />
            <p>$ 6.500 <br />
              Zapote <br />
              1 kilo <br />
              Ciudad Cordoba</p>
            <button onClick={() => agregarAlCarrito({ nombre: 'Zapote',
precio: 6500, cantidad: 1 })}>
              Agregar al Carrito
            </button>
          </div>
          <div className="img-frutas">
            <img src={img14} alt="Pera" />
            <p>$ 13.960 <br />
              Pera <br />
              1 kilo <br />
              Ciudad Boyaca</p>
            <button onClick={() => agregarAlCarrito({ nombre: 'Pera',
precio: 13960, cantidad: 1 })}>
              Agregar al Carrito
            </button>
          </div>
        </div>
      </div>
      <div className='carrito-compras'>
        <h2>Carrito de Compras</h2>
        <button onClick={() => setCarritoVisible(!carritoVisible)}>
          {carritoVisible ? 'Ocultar' : 'Mostrar'}
        </button>
        {carritoVisible && (
          <div>
            <ul>
              {carrito.map((item, index) => (
                <li key={index}>
                  {item.nombre} - ${item.precio}
```

```
                    <button onClick={() =>
eliminarDelCarrito(index)}>Eliminar</button>
                  </li>
                ))}
              </ul>
              <p>Total: ${calcularTotal()}</p>
              <Link to="/comprar"><button>Comprar</button></Link>
            </div>
          )}
        </div>
      </div>
    </div>
  );
};
```

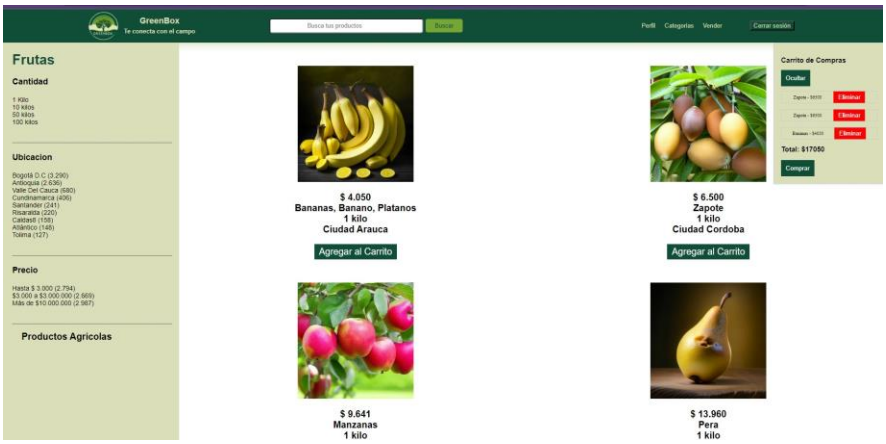
Se utiliza la bibliotecas import { Link } from 'react-router-dom'; y import React, { useState } from 'react'; para permitir la interacion adecuadamente.



En la parte derecha de la pagina podemos evidenciar el carrito de compras con el botón mostrar para poder visualizar los productos que se agreguen.



También podemos evidenciar los productos agregado en el carrito de compra el cual se pueden eliminar y se ira sumando los precios de los artículos que estén agregado.



Ya cuando el usuario esta seguro de su compra le puede dar el botón comprar el cual dirigirá al usuario a una simulación de ruta.

Conclusiones

- Desarrollo Estructurado: La documentación refleja un enfoque estructurado para el desarrollo del proyecto. Se mencionan claramente las etapas del desarrollo, desde la instalación de dependencias hasta la implementación de características específicas.
- Uso de Tecnologías Actuales: Se destaca el uso de tecnologías actuales y populares, como React y React Router, para el desarrollo del proyecto. Además, se mencionan bibliotecas específicas utilizadas para el manejo de rutas y la interfaz de usuario.
- Seguridad en el Registro y Inicio de Sesión: La inclusión de bcrypt para el cifrado de contraseñas en el servidor es un aspecto positivo en términos de seguridad. También se implementa una lógica coherente para el registro de usuarios y la autenticación.
- Carrito de Compras Funcionalidad: La implementación del carrito de compras es clara y proporciona funciones esenciales, como agregar productos, eliminar productos y calcular el total. La interacción con el carrito se realiza de manera efectiva en la interfaz de usuario.
- Detalles de Productos y Compra: Se presenta de manera clara la información de los productos, incluyendo imágenes, precios y descripciones. La simulación de compra también está bien manejada, brindando al usuario la capacidad de revisar y confirmar su selección.
- Rutas Públicas y Privadas: Se utiliza React Router para gestionar las rutas del proyecto, diferenciando entre páginas públicas y privadas. Además, se implementa un componente ProtectedRoute para asegurar que ciertas páginas solo sean accesibles para usuarios autenticados.