

UNIVERSIDAD AUTÓNOMA DE SINALOA
FACULTAD DE INFORMÁTICA DE CULIACÁN
DESARROLLO WEB DEL LADO DEL SERVIDOR



Nombre: Ana Maribel Felipe Rodriguez
Docente: José Manuel Cazarez Alderete
Grupo: 2-3
Turno: Matutino

Actividad final: CRUD completo

BACKEND

Inicialmente en este proyecto se realizó un inventario de ropa donde se puede agregar, editar, guardar y eliminar los productos que se van agregando, en base a esto se va conectando a una base de datos de SQLite.

Express: Construye aplicaciones web en Node.js

BodyParser: Procesa los datos del cuerpo (body) de las solicitudes HTTP, como formularios JSON o datos enviados por POST.

Inventario: Contiene funciones, datos o rutas relacionadas con el inventario.

Cors: permite que tu servidor acepte solicitudes desde otros dominios o puertos. Con este puedes trabajar con FRONTEND Y BACKEND por separado.

App: Define las rutas

Puerto: Define el servidor que se ejecuta en el puerto 3000



The screenshot shows a code editor interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, ...
- Title Bar:** Proyecto desarrollo web
- Editor Area:** The file `app.js` is open, showing the following code:

```
const express = require('express')
const bodyParser = require('body-parser')
const inventario = require('./inventario')
const cors = require('cors')
const app = express()
const puerto = 3000

app.use(bodyParser.json())
app.use(cors())

app.listen(puerto, () => {
    console.log('servicio iniciado')
})
```

Explorer Panel: Shows the project structure:

- OPEN EDITORS: `app.js`
- PROYECTO DESARROLLO...:
 - Inventario
 - Inventario r...
 - node_modules
 - app.js
 - conexion.js
 - inventario.js
 - Inventory...
 - package-lock.json
 - package.json
 - tempCodeRunn...
 - 1.txt
 - README.md

En este existen los métodos:

- **GET:** En este se muestra los datos que se insertaran en la tabla
- **POST:** En este guarda los datos en la tabla
- **PUT:** En este se actualizan los datos de la tabla
- **DELETE:** En este se elimina los datos de la tabla

The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows the project structure: PROYECTO DESARROLLO... (Inventory) containing app.js, conexion.js, inventario.js, package-lock.json, package.json, tempCodeRunn..., 1.txt, and README.md.
- Editor:** The app.js file is open, displaying code for a Node.js application. The code handles HTTP requests for '/Mostrar', '/Guardar', '/Actualizar/:id', and '/Eliminar/:id' routes using async/await syntax and Sequelize for database operations.
- Bottom Status Bar:** Shows the author (Ana Maribel Felipe Rodriguez), last save time (2 hours ago), current position (In 32, Col 41), and file encoding (UTF-8).

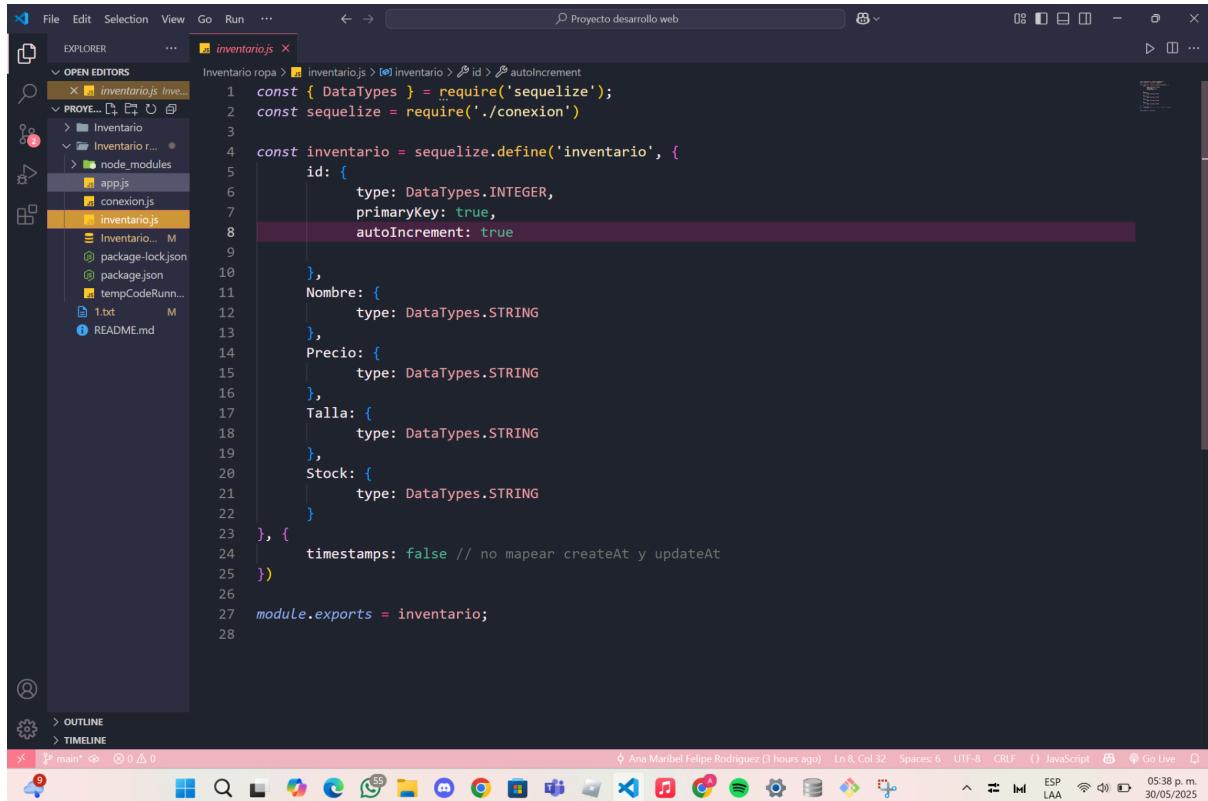
Con este archivo js se realiza la conexión a la base de datos donde con este se puede interactuar usando JavaScript

The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows the project structure: PROYECTO DESARROLLO... (Inventory) containing app.js,conexion.js, inventario.js, package-lock.json, package.json, tempCodeRunn..., 1.txt, and README.md.
- Editor:** The conexion.js file is open, showing the Sequelize configuration for a SQLite database named './InventarioRopa.db'.
- Bottom Status Bar:** Shows the author (Ana Maribel Felipe Rodriguez), last save time (2 hours ago), current position (In 8, Col 28), and file encoding (UTF-8).

En este archivo principalmente en las constantes hay una conexión a mi base de datos y otro sequelize donde definimos los campos de la tablas.

También mapeamos una tabla llamada inventario en la base de datos y los campos de la tabla y exportamos el modelo.



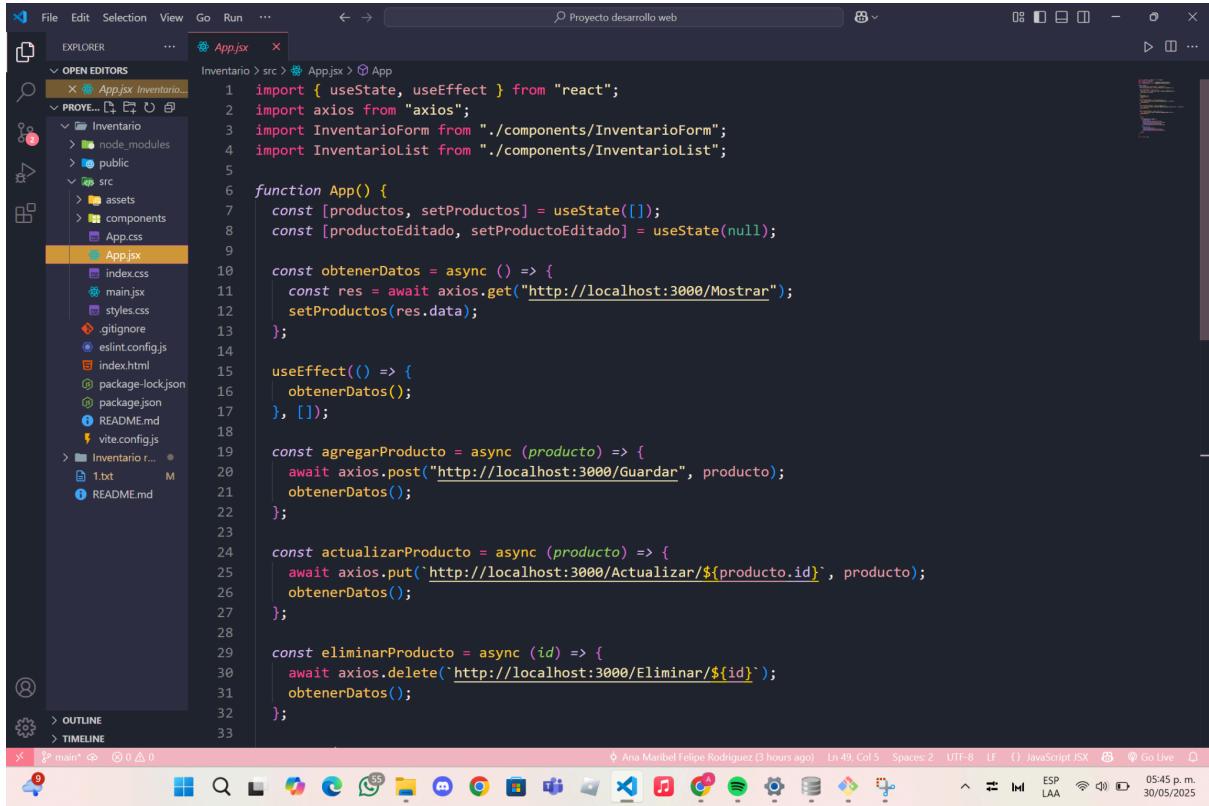
```
const { DataTypes } = require('sequelize');
const sequelize = require('../conexion');

const inventario = sequelize.define('inventario', {
    id: {
        type: DataTypes.INTEGER,
        primaryKey: true,
        autoIncrement: true
    },
    Nombre: {
        type: DataTypes.STRING
    },
    Precio: {
        type: DataTypes.STRING
    },
    Talla: {
        type: DataTypes.STRING
    },
    Stock: {
        type: DataTypes.STRING
    }
}, {
    timestamps: false // no mapear createdAt y updatedAt
});

module.exports = inventario;
```

FRONTEND

Aquí en este archivo de App.jsx comenzamos con las importaciones usamos axios para hacer peticiones http al backend y también tenemos a dos componentes hijos uno del formulario y otra para la lista de los productos. En si este archivo es la base de una app CRUD (crear, leer, actualizar y eliminar) para un inventario se comunica con el backend y muestra y gestiona los productos de ropa.



The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure with files like index.css, main.jsx, styles.css, .gitignore, .eslint.config.js, index.html, package-lock.json, package.json, README.md, vite.config.js, and several Inventory-related files.
- Editor:** The main editor window displays the code for `App.jsx`. The code uses React hooks (`useState`, `useEffect`) and the `axios` library to interact with a backend API at `http://localhost:3000`.
- Bottom Status Bar:** Shows the author (`Ana Maribel Felipe Rodriguez`), last save time (`3 hours ago`), file statistics (`Ln 49, Col 5, Spaces: 2, UTF-8, LF`), language (`JavaScript JSX`), and other system information.

```
import { useState, useEffect } from "react";
import axios from "axios";
import InventoryForm from "./components/InventoryForm";
import InventoryList from "./components/InventoryList";

function App() {
  const [productos, setProductos] = useState([]);
  const [productoEditado, setProductoEditado] = useState(null);

  const obtenerDatos = async () => {
    const res = await axios.get("http://localhost:3000/Mostrar");
    setProductos(res.data);
  };

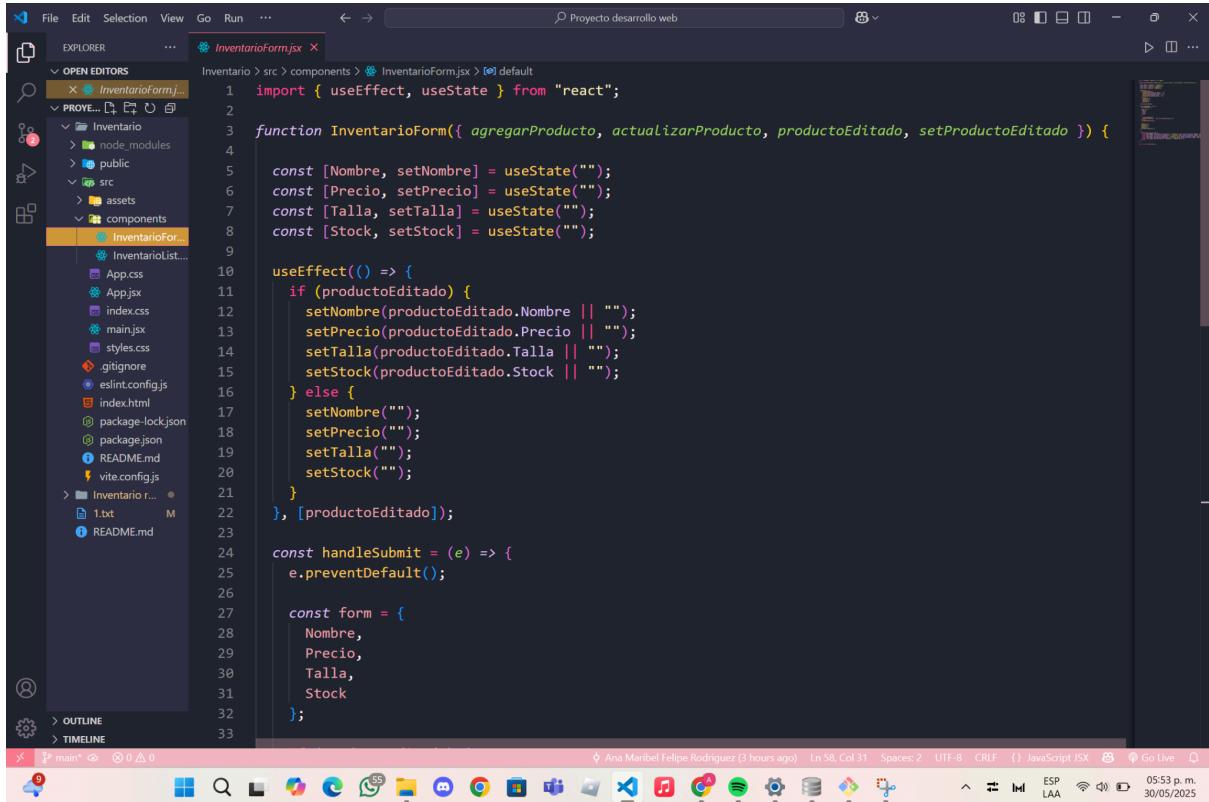
  useEffect(() => {
    obtenerDatos();
  }, []);

  const agregarProducto = async (producto) => {
    await axios.post("http://localhost:3000/Guardar", producto);
    obtenerDatos();
  };

  const actualizarProducto = async (producto) => {
    await axios.put(`http://localhost:3000/Actualizar/${producto.id}`, producto);
    obtenerDatos();
  };

  const eliminarProducto = async (id) => {
    await axios.delete(`http://localhost:3000/Eliminar/${id}`);
    obtenerDatos();
  };
}
```

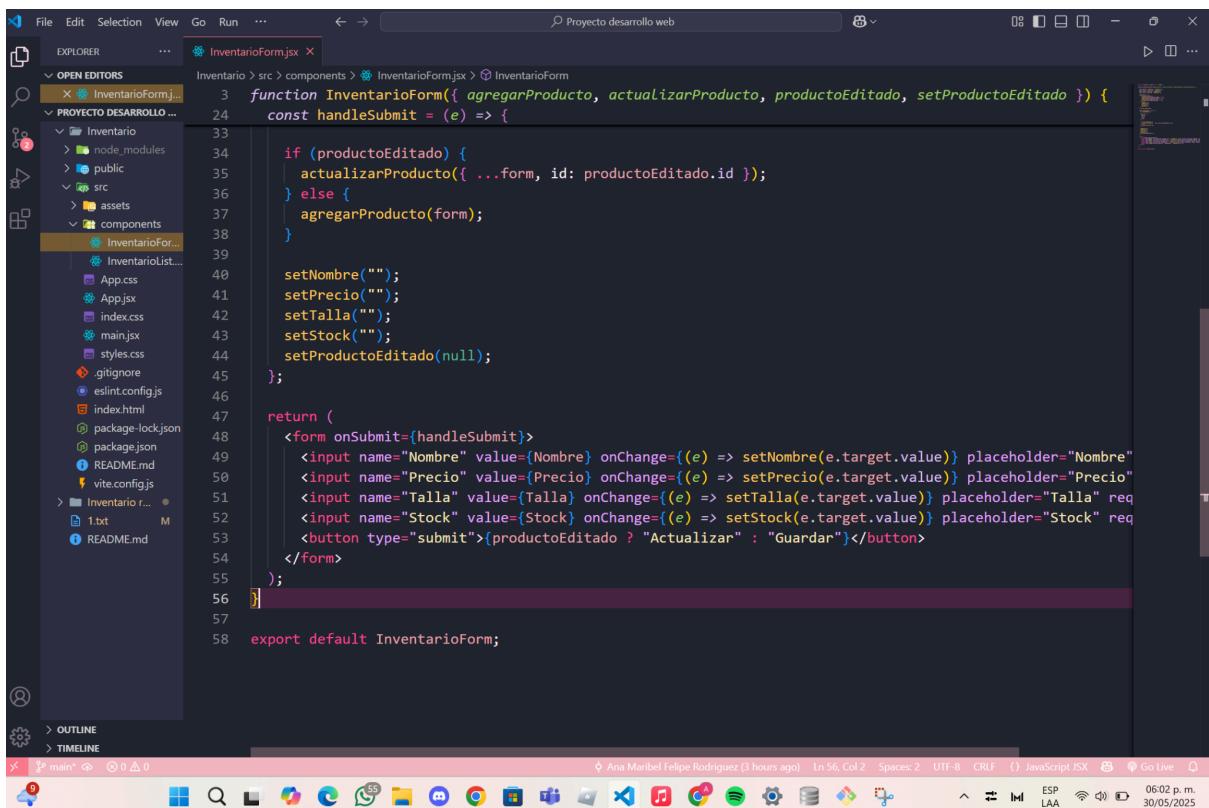
Este archivo es un formulario para poder agregar o editar productos de un inventario. Donde usa las constantes para guardar los valores del campo del formulario, actualiza los campos del formulario, muestra los campos para que el usuario escriba y muestra los botones para editar o guardar un producto en la tabla



```

File Edit Selection View Go Run ...
Proyecto desarollo web
OPEN EDITORS
InventarioForm.jsx
Inventario > src > components > InventarioForm.jsx > default
1 import { useEffect, useState } from "react";
2
3 function InventarioForm({ agregarProducto, actualizarProducto, productoEditado, setProductoEditado }) {
4
5     const [Nombre, setNombre] = useState("");
6     const [Precio, setPrecio] = useState("");
7     const [Talla, setTalla] = useState("");
8     const [Stock, setStock] = useState("");
9
10    useEffect(() => {
11        if (productoEditado) {
12            setNombre(productoEditado.Nombre || "");
13            setPrecio(productoEditado.Precio || "");
14            setTalla(productoEditado.Talla || "");
15            setStock(productoEditado.Stock || "");
16        } else {
17            setNombre("");
18            setPrecio("");
19            setTalla("");
20            setStock("");
21        }
22    }, [productoEditado]);
23
24    const handleSubmit = (e) => {
25        e.preventDefault();
26
27        const form = {
28            Nombre,
29            Precio,
30            Talla,
31            Stock
32        };
33
34        const handleSubmit = (e) => {
35            if (productoEditado) {
36                actualizarProducto({ ...form, id: productoEditado.id });
37            } else {
38                agregarProducto(form);
39            }
40
41            setNombre("");
42            setPrecio("");
43            setTalla("");
44            setStock("");
45            setProductoEditado(null);
46
47        return (
48            <form onSubmit={handleSubmit}>
49                <input name="Nombre" value={Nombre} onChange={(e) => setNombre(e.target.value)} placeholder="Nombre" />
50                <input name="Precio" value={Precio} onChange={(e) => setPrecio(e.target.value)} placeholder="Precio" />
51                <input name="Talla" value={Talla} onChange={(e) => setTalla(e.target.value)} placeholder="Talla" required />
52                <input name="Stock" value={Stock} onChange={(e) => setStock(e.target.value)} placeholder="Stock" required />
53                <button type="submit">{productoEditado ? "Actualizar" : "Guardar"}</button>
54            </form>
55        );
56    };
57
58    export default InventarioForm;

```

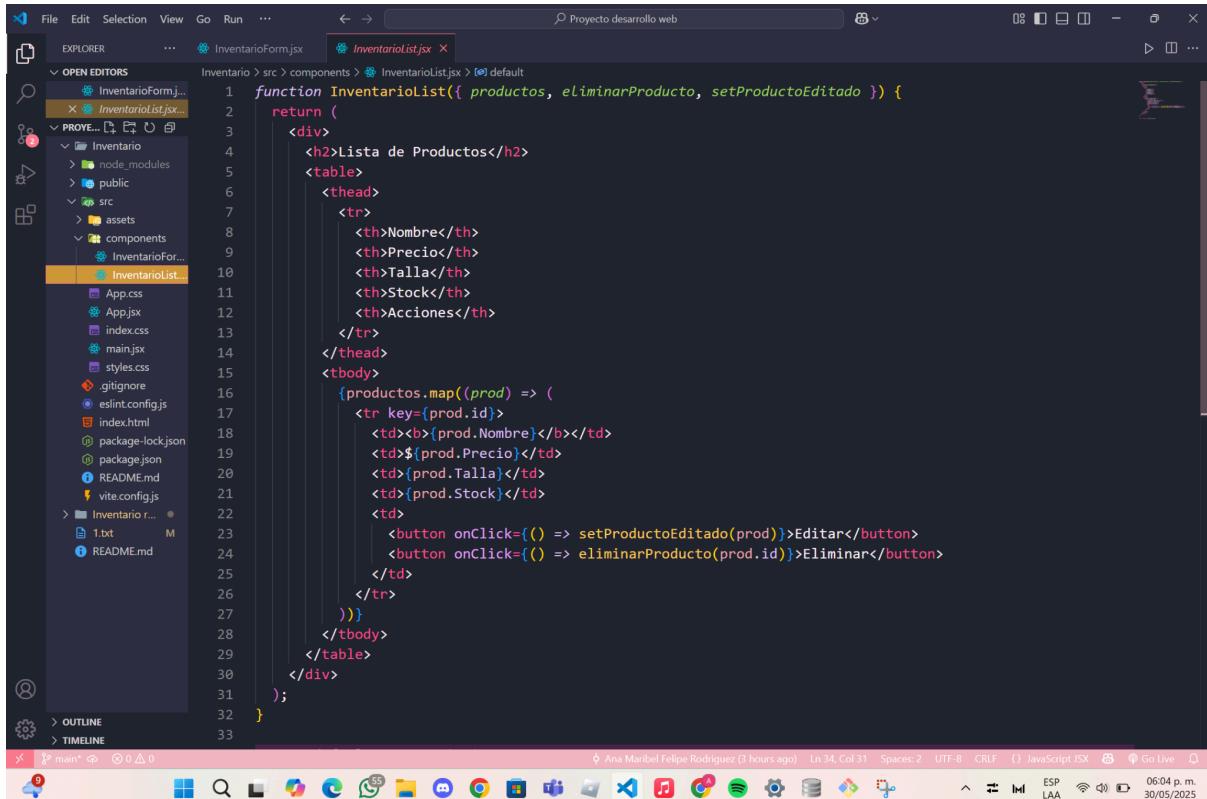


```

File Edit Selection View Go Run ...
Proyecto desarollo web
OPEN EDITORS
InventarioForm.jsx
Inventario > src > components > InventarioForm.jsx > InventarioForm
PROYECTO DESARROLLO ...
Inventario > src > components > InventarioForm.jsx > default
3 function InventarioForm({ agregarProducto, actualizarProducto, productoEditado, setProductoEditado }) {
4
5     const handleSubmit = (e) => {
6
7         if (productoEditado) {
8             actualizarProducto({ ...form, id: productoEditado.id });
9         } else {
10            agregarProducto(form);
11        }
12
13        setNombre("");
14        setPrecio("");
15        setTalla("");
16        setStock("");
17        setProductoEditado(null);
18
19    return (
20        <form onSubmit={handleSubmit}>
21            <input name="Nombre" value={Nombre} onChange={(e) => setNombre(e.target.value)} placeholder="Nombre" />
22            <input name="Precio" value={Precio} onChange={(e) => setPrecio(e.target.value)} placeholder="Precio" />
23            <input name="Talla" value={Talla} onChange={(e) => setTalla(e.target.value)} placeholder="Talla" required />
24            <input name="Stock" value={Stock} onChange={(e) => setStock(e.target.value)} placeholder="Stock" required />
25            <button type="submit">{productoEditado ? "Actualizar" : "Guardar"}</button>
26        </form>
27    );
28
29    export default InventarioForm;

```

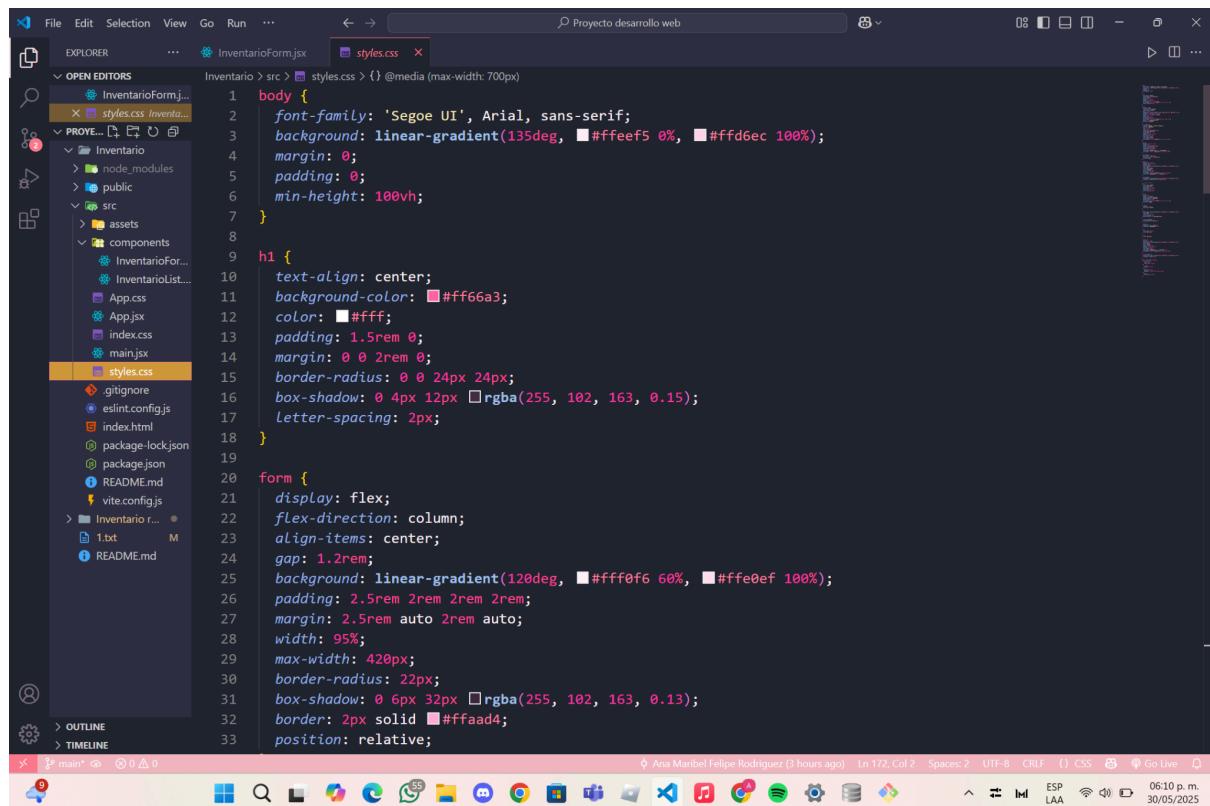
Este es una archivo que se enfoca en el listado del inventario en pocas palabras es el encargado de mostrar todos los productos del inventario en una tabla y aquí se realiza las acciones de editar y eliminar productos que están hechos botones



The screenshot shows a code editor interface with the following details:

- File Path:** Proyecto desarollo web / Inventario / src / components / InventoryList.jsx
- Code Content:** The code is a functional component named `InventoryList`. It returns a `<div>` containing an `<h2>Lista de Productos</h2>`, a `<table>` with a `<thead>` and `<tbody>`. The `<thead>` has columns for Nombre, Precio, Talla, Stock, and Acciones. The `<tbody>` contains rows for each product in the `productos` array, with buttons for Editar and Eliminar.
- Editor Features:** The interface includes a top navigation bar with File, Edit, Selection, View, Go, Run, etc., and a bottom status bar showing the author (Ana Maribel Felipe Rodriguez), last save time (3 hours ago), and other details like file size and encoding.

Y los estilos

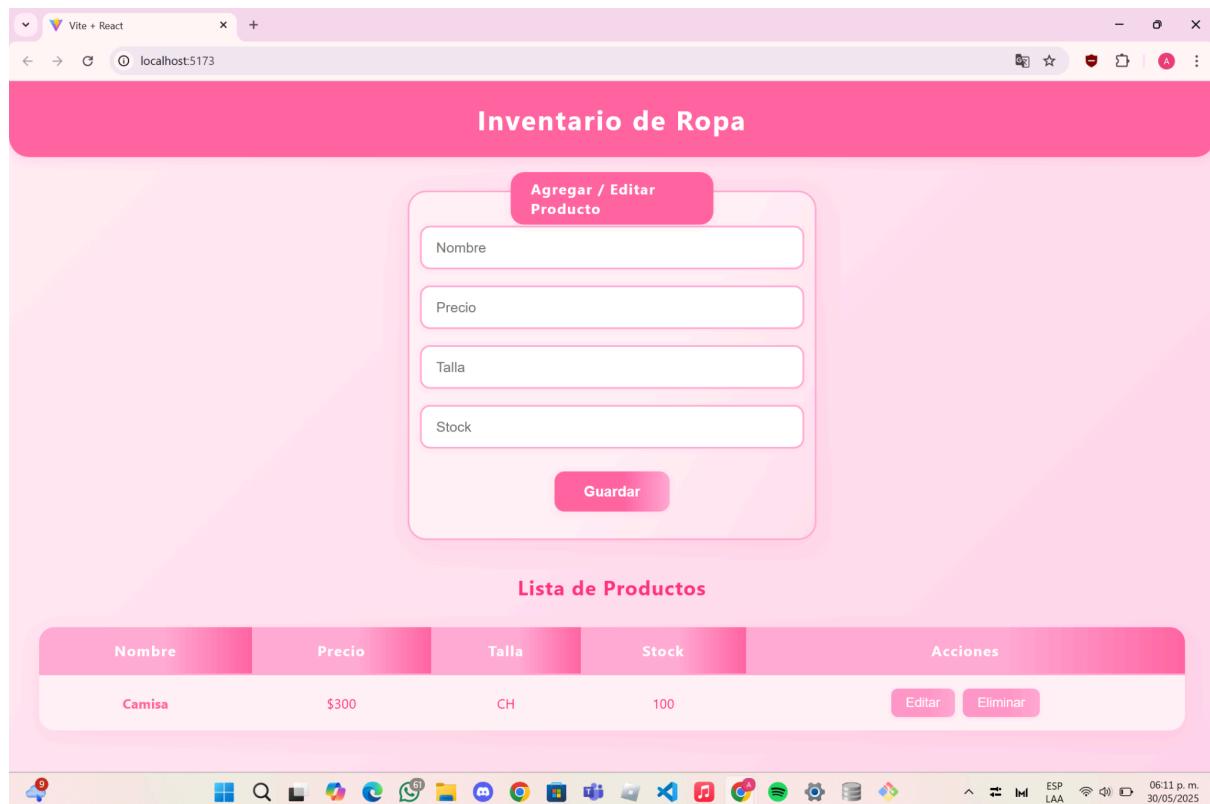


```
body {
    font-family: 'Segoe UI', Arial, sans-serif;
    background: linear-gradient(135deg, #fffeef 0%, #ffd6ec 100%);
    margin: 0;
    padding: 0;
    min-height: 100vh;
}

h1 {
    text-align: center;
    background-color: #ff66a3;
    color: #fff;
    padding: 1.5rem 0;
    margin: 0 0 2rem 0;
    border-radius: 0 0 24px 24px;
    box-shadow: 0 4px 12px rgba(255, 102, 163, 0.15);
    letter-spacing: 2px;
}

form {
    display: flex;
    flex-direction: column;
    align-items: center;
    gap: 1.2rem;
    background: linear-gradient(120deg, #fffff6 60%, #ffe0ef 100%);
    padding: 2.5rem 2rem 2rem 2rem;
    margin: 2.5rem auto 2rem auto;
    width: 95%;
    max-width: 420px;
    border-radius: 22px;
    box-shadow: 0 6px 32px rgba(255, 102, 163, 0.13);
    border: 2px solid #ffaad4;
    position: relative;
}
```

Aquí está mi inventario y se muestra como se agrega un producto:



The screenshot shows a web application titled "Inventario de Ropa". At the top, there is a navigation bar with links for "Home", "About", "Contact", and "Logout". Below the navigation, there is a search bar and a button labeled "Nuevo". The main content area has two sections: "Agregar / Editar Producto" (Add/Edit Product) and "Lista de Productos" (List of Products).

Agregar / Editar Producto

Nombre	Precio	Talla	Stock	Acciones
Camisa	\$300	CH	100	<button>Editar</button> <button>Eliminar</button>

Lista de Productos

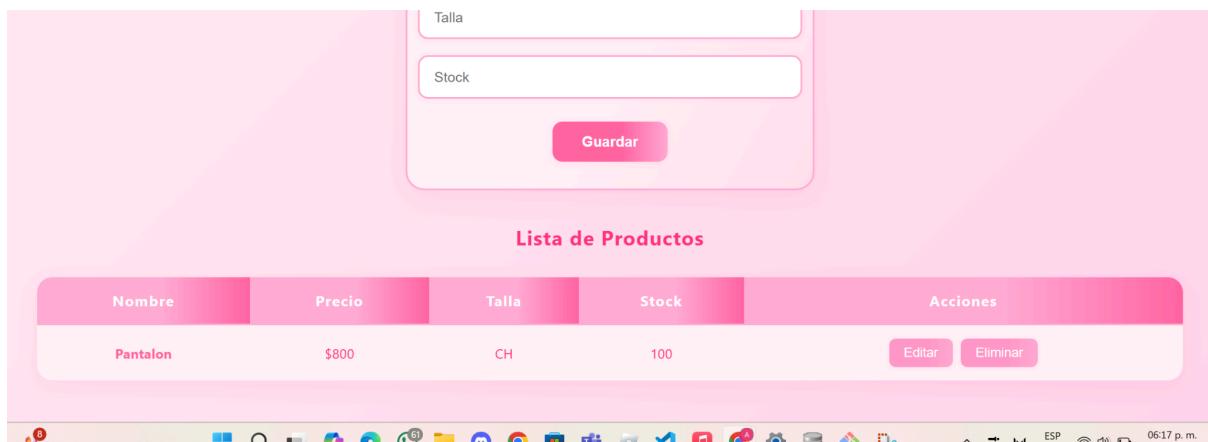
Nombre	Precio	Talla	Stock	Acciones
Camisa	\$300	CH	100	<button>Editar</button> <button>Eliminar</button>

Aquí se muestra como mi conexión a mi base de datos fue exitosa ya que se pudo agregar exitosamente el producto

The screenshot shows the DB Browser for SQLite application interface. The main window displays a table named 'Inventarios' with columns: Nombre, Precio, Talla, Stock, and Id. A single row is present: 'Camisa' with a price of '\$300', size 'CH', stock '100', and ID '12'. On the right side, there's an 'Edit Database Cell' panel where the 'Id' field is being edited. The status bar at the bottom right indicates the connection is to 'Local' and the date and time as '06/12 p.m. 30/05/2025'.

Ahora en un ejemplo de que queramos actualizar el producto de una camisa a un pantalón y su precio también:

The screenshot shows a web browser displaying a Vite + React application titled 'Inventario de Ropa'. In the center, there's a modal dialog titled 'Agregar / Editar Producto' containing fields for Nombre ('Pantalon'), Precio ('800'), Talla ('CH'), and Stock ('100'). Below the modal is a table titled 'Lista de Productos' showing a single row: 'Camisa' with a price of '\$300', size 'CH', and stock '100'. There are 'Editar' and 'Eliminar' buttons next to the row. The status bar at the bottom right indicates the connection is to 'Local' and the date and time as '06/17 p.m. 30/05/2025'.

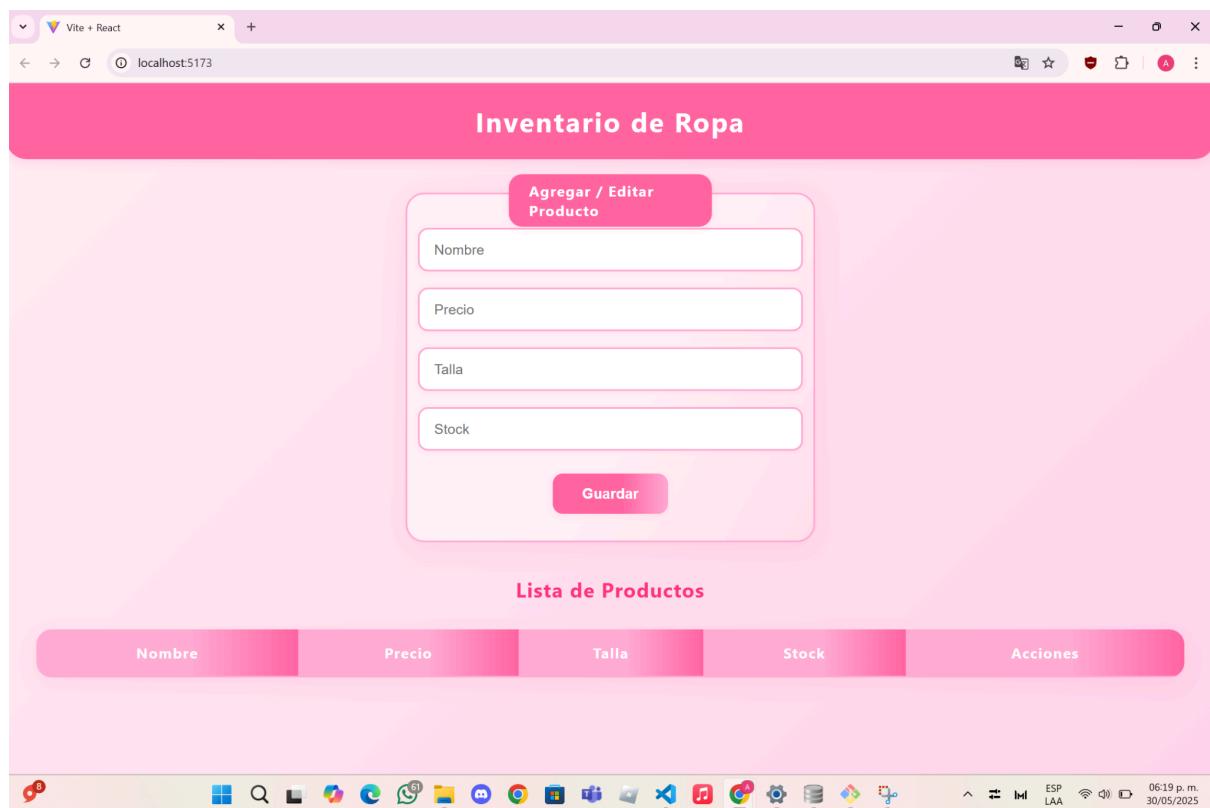


Aquí se actualizó a mi base de datos con éxito

Nombre	Precio	Talla	Stock	Id
Pantalon	800	CH	100	12

Aquí mismo podemos darnos cuenta de un botón de eliminar dónde nomas le picas y se borra el producto tanto de la lista como de la base de datos

Nombre	Precio	Talla	Stock	Acciones
Pantalon	\$800	CH	100	<button>Editar</button> <button>Eliminar</button>



Y la base de datos ahora está vacía

