

# **SISTEMA DE INFORMACIÓN PARA LA GESTIÓN DE DEVOLUCIÓN JOHAN UNIFORMS**

.....



# CONTENIDO

---

Objetivo general

Objetivos específicos

Planteamiento del problema

Alcance del proyecto

Api externa

Api interna

Control de versiones

---

# Objetivo general

Diseñar, desarrollar y poner en funcionamiento un sistema informático que opere en la devolución de uniformes, con el objetivo de hacer fácil y rápido este proceso.

# Objetivos específicos

- Implementar un módulo de devoluciones dentro del sistema de información.
- Integrar módulos de inicio de sesión y registro para llevar un orden y una visualización dependiendo del rango en el que se encuentre.
- Mantener actualizaciones del sistema teniendo en cuenta aspectos como la organización y seguridad de la información.
- Poner en funcionamiento el sistema de información por medio del acceso a las ventas necesarias para gestionar las devoluciones

# Planteamiento del problema

Por medio de una entrevista realizada al administrador de la tienda se dio a conocer como conclusión la identificación del problema en las devoluciones de la tienda, debido a que la atención al cliente es un proceso muy demorado al momento de devolver una prenda o el uniforme completo.

# Alcance del proyecto

Poder identificar las necesidades e implementar un sistema donde se evidencien las devoluciones por medio de módulos para se pueda garantizar los requisitos brindando un soporte en la información de los clientes que se ve reflejada en las devoluciones.



# API EXTERNA



---

---

# LEAFLET

Visualizar datos geoespeciales, esta Api permite mostrar información geográfica en este caso la ubicación de la empresa de respaldo.

- La API se cargará en el proyecto mediante una llamada al script en HTML y utilizando librerías de React específicas como react-leaflet.



# Consumo

```
function Principal() {  
  const mapRef = useRef(null);  
  
  useEffect(() => {  
    if (!mapRef.current) {  
      mapRef.current = L.map("map").setView([4.6289784, -74.1362913], 13);  
  
      L.tileLayer("https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png", {  
        attribution: '&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors',  
      }).addTo(mapRef.current);  
  
      L.marker([4.6289784, -74.1362913]).addTo(mapRef.current)  
        .bindPopup("Ubicación de Johan Uniforms")  
        .openPopup();  
    }  
  
    return () => {  
      if (mapRef.current) {  
        mapRef.current.remove();  
        mapRef.current = null;  
      }  
    };  
  }, []);  
}
```

# Consumo



# API INTERNA



# CRUD

Manipular datos de los clientes permitiendo hacer operaciones como crear, leer, actualizar y eliminar implementando la seguridad en el manejo de la información.

- En la Api se implementaron métodos get, post, put y delete en este caso necesarios para hacer operaciones CRUD.

# Método

```
app.post("/create", (req, res) => {  
  const { id, nombre, cedula, fecha } = req.body;  
  
  db.query('INSERT INTO crud (id,nombre,cedula,fecha) VALUES (?, ?, ?, ?)',  
    [id, nombre, cedula, fecha],  
    (err) => {  
      if (err) {  
        console.log(err);  
        return res.status(500).send("Error al registrar el usuario");  
      }  
      res.send("Usuario registrado con éxito");  
    });  
});
```

# Método

```
app.get("/crud", (req, res) => {  
  db.query('SELECT * FROM crud', (err, result) => {  
    if (err) {  
      console.log(err);  
      return res.status(500).send("Error al obtener usuarios");  
    }  
    res.send(result);  
  });  
});
```

```
app.put("/actualizar", (req, res) => {  
  const { id, nombre, cedula, fecha } = req.body;  
  
  db.query('UPDATE crud SET nombre=?, cedula=?, fecha=? WHERE id=?',  
    [nombre, cedula, fecha, id],  
    (err) => {  
      if (err) {  
        console.log(err);  
        return res.status(500).send("Error al actualizar el usuario");  
      }  
      res.send("Usuario actualizado con éxito");  
    });  
});
```

# Método

```
app.delete("/crud/:id", (req, res) => {  
  const { id } = req.params;  
  
  db.query('DELETE FROM crud WHERE id = ?', [id], (err, result) => {  
    if (err) {  
      console.error("Error al eliminar el usuario:", err);  
      return res.status(500).send("Error al eliminar el usuario");  
    }  
  
    if (result.affectedRows === 0) {  
      return res.status(404).send("Usuario no encontrado");  
    }  
  
    res.send("Deceas eliminar al usuario?");  
  });  
});
```

# Consumo

IDENTIFICACIÓN

Nombre

Cédula

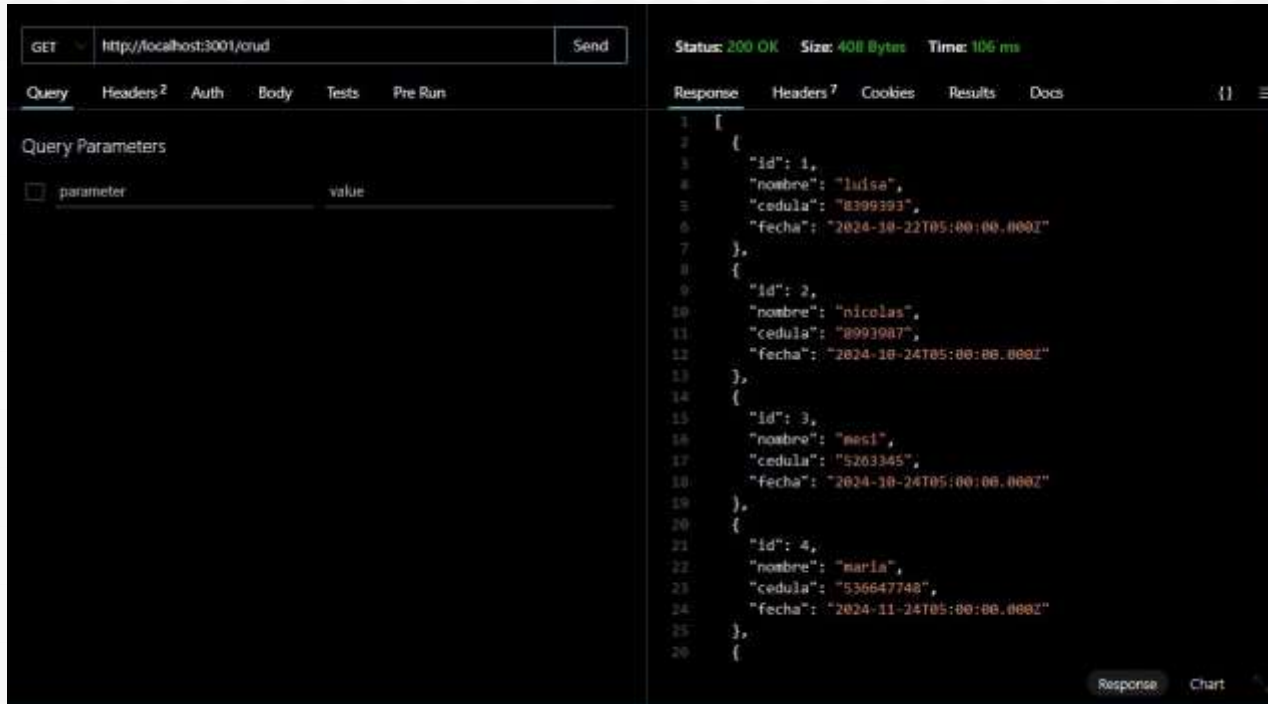
dd/mm/aaaa

Agregar

IDENTIFICACIÓN	Nombre	Cédula	Fecha	Acciones
1	Luisa	8399393	2024-10-22T05:00:00.000Z	<div>Editar</div> <div>Eliminar</div>
2	Nicolás	8993987	2024-10-24T05:00:00.000Z	<div>Editar</div> <div>Eliminar</div>
3	mes	5283345	2024-10-24T05:00:00.000Z	<div>Editar</div> <div>Eliminar</div>
4	Maria	538647748	2024-11-24T05:00:00.000Z	<div>Editar</div> <div>Eliminar</div>



# Prueba



The screenshot displays a REST client interface with a GET request to `http://localhost:3001/crud`. The response is a JSON array of four user objects, each containing `id`, `nombre`, `cedula`, and `fecha` fields. The status is 200 OK, and the response size is 408 Bytes.

**Query** Headers 2 Auth Body Tests Pre Run

Query Parameters

parameter	value
-----------	-------

**Status:** 200 OK **Size:** 408 Bytes **Time:** 106 ms

**Response** Headers 7 Cookies Results Docs {}

```
1 {
2   {
3     "id": 1,
4     "nombre": "luisa",
5     "cedula": "8399393",
6     "fecha": "2024-10-22T05:00:00.000Z"
7   },
8   {
9     "id": 2,
10    "nombre": "nicolas",
11    "cedula": "8993987",
12    "fecha": "2024-10-24T05:00:00.000Z"
13  },
14  {
15    "id": 3,
16    "nombre": "nesi",
17    "cedula": "5263345",
18    "fecha": "2024-10-24T05:00:00.000Z"
19  },
20  {
21    "id": 4,
22    "nombre": "maria",
23    "cedula": "536647748",
24    "fecha": "2024-11-24T05:00:00.000Z"
25  },
26  }
```

Response Chart

# Recuperar contraseña

Actualizar la contraseña en caso de pérdida u olvido, teniendo la posibilidad de renovarla por medio del correo para obtener mayor seguridad.

- .En la Api se implementó el método post con el propósito de crear un nuevo recurso en este caso la contraseña.



# Método

```
try {  
  const response = await fetch('http://localhost:5004/RecoverPassword', {  
    method: 'POST',  
    headers: {  
      'Content-Type': 'application/json',  
    },  
    body: JSON.stringify({ email }), // Enviar el correo al backend  
  });  
  
  const data = await response.json();  
}
```



# Consumo

Sistema De Información Jhuán Uniforms

[Principal](#) [Formulario de consultas](#) [Ingresar](#) [Registrarse](#) [Devolución](#)

Correo enviado correctamente. Por favor revisa tu bandeja de entrada.

Correo Electrónico

ENVIAR CORREO

## Restablecer Contraseña

Nueva Contraseña

Confirmar Contraseña

RESTABLECER  
CONTRASEÑA



# Prueba

```
import React, { useState } from 'react';
import { Link } from 'react-router-dom';
import './style/recuperar.css';

const RecoverPassword = () => {
  const [email, setEmail] = useState('');
  const [errorMessage, setErrorMessage] = useState('');
  const [successMessage, setSuccessMessage] = useState('');

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

    // Validar que el email no esté vacío
    if (!email) {
      setErrorMessage('Por favor ingresa un correo electrónico.');
```

---

```
      return;
    }

    try {
      const response = await fetch('http://localhost:8080/recover-password', {
        method: 'POST',
```



# Factura

comprobante de la transacción y facilitar el proceso de pago los componentes clave y el funcionamiento de la factura incluyendo los todos los datos necesarios como nombre, numero de factura, la fecha en que se realizó, los productos, y el total

- Se utilizo el método del post permite crear, actualizar y almacenar información relacionada con transacciones comerciales de manera segura y eficiente.



# Método

```
const confirmDevolucion = window.confirm("¿Seguro que quieres devolver este producto?");
if (confirmDevolucion) {
  try {
    const productosArray = [productoSeleccionado];

    const response = await axios.post('http://localhost:3002/api/devolucion', {
      numeroFactura,
      nombre: factura.nombre,
      telefono: factura.telefono,
      productos: productosArray,
      total: factura.total,
      metodoPago: factura.metodoPago,
      comentarios: comentarios,
    });
  }
```



# Factura de compra

laura

1234567

100002

## Información de Venta

Código	Producto	Cantidad	Precio	Total parcial
MONPP1D7	Jardinera Oscura	1	\$ 45500.00	\$ 45500.00

Total: \$ 45500.00

## Métodos de pago

- ☐  Crédito con Mastercard
- ☐  Débito con Nequi
- ☐  Débito con Daviplata
- ☐  Débito con PayPal
- ☒  Efectivo

Imprimir Factura





# Prueba

```
✓ const Factura = () => {  
  const location = useLocation();  
  const navigate = useNavigate();  
  const { carrito = [], total = 0 } = location.state || { productos: [] };  
  
  const [nombre, setNombre] = useState('');  
  const [telefono, setTelefono] = useState('');  
  const [numeroFactura, setNumeroFactura] = useState(0);  
  const [metodoPago, setMetodoPago] = useState('');  
  const [productosConCodigo, setProductosConCodigo] = useState([]);  
  
  ✓ useEffect(() => {  
    const lastInvoiceNumber = localStorage.getItem('lastInvoiceNumber') || 100000;  
    const nextInvoiceNumber = Number(lastInvoiceNumber) + 1;  
    setNumeroFactura(nextInvoiceNumber);  
    localStorage.setItem('lastInvoiceNumber', nextInvoiceNumber);  
  
    ✓ const carritoConCodigo = carrito.map(producto => ({  
      ...producto,  
      codigo: producto.codigo || generarCodigoAleatorio()  
    }));  
  });  
};
```



# Registro e inicio

procesos fundamentales que permiten a los usuarios interactuar con las plataformas digitales de manera segura y personalizada establecer una identidad única y proteger la información personal, estos procesos mejoran la experiencia del usuario y fomentan la confianza en la plataforma.

- Se utilizaron por medio de los métodos post, Su uso asegura que la información personal esté protegida y facilita la validación y autenticación necesarias.



# Método

```
const handleSubmit = async () => {  
  if (action === "Registro") {  
    if (nombre && apellido && correo && contraseña) {  
      try {  
        const nuevoUsuario = { nombre, apellido, correo, contraseña };  
        await axios.post('http://localhost:5001/api/registro', nuevoUsuario);  
        setAction("Inicio Sesión");  
        setError('');  
        resetFields();  
      } catch (error) {  
        setError('Error al registrar el usuario');  
      }  
    } else {  
      setError('Por favor, completa todos los campos.');    }  
  } else {
```



# Método

```
    } else {  
      try {  
        const response = await axios.post('http://localhost:5001/api/login', { correo, contraseña });  
        if (response.status === 200) {  
          navigate('/carrito');  
        }  
      } catch (error) {  
        setError('Correo o contraseña incorrectos.');      }  
    }  
  } else {  
    setError('Por favor, completa todos los campos.');  }  
}
```



# Consumo

**Registro**

nombre

Apellido

correo

contraseña

[¿Tienes cuenta? Iniciar sesión](#)

REGISTRADOR



# Prueba

```
const InicioYRegistro = () => {  
  const navigate = useNavigate();  
  const [action, setAction] = useState("Registro");  
  const [nombre, setNombre] = useState('');  
  const [apellido, setApellido] = useState('');  
  const [correo, setCorreo] = useState('');  
  const [contraseña, setContraseña] = useState('');  
  const [error, setError] = useState('');  
  
  const adminAccounts = [  
    { correo: "dkim44243@gmail.com", contraseña: "twicebestgg" },  
    { correo: "valentinadb13l@gmail.com", contraseña: "12345" },  
    { correo: "vallejolorena37@gmail.com", contraseña: "54321" },  
    { correo: "valentinavaquezrodriguez00@gmail.com", contraseña: "56789" }  
  ];  
  
  const handleSubmit = async () => {  
    if (action === "Registro") {  
      if (nombre && apellido && correo && contraseña) {  
        try {  
          const nuevoUsuario = { nombre, apellido, correo, contraseña };  
          await axios.post('http://localhost:5001/api/registro', nuevoUsuario);  
          setAction("Inicio Sesión");  
          setError('');  
        }  
      }  
    }  
  }  
}
```



# Devolución

Gestionar una devolución por medio de un número consecutivo, esto con la finalidad de localizar la compra y así generar correctamente la devolución

- Se utilizarán los métodos post y get puesto que al momento de incluir un módulo de venta y devolución es necesario el reemplazo de algún recurso existente y asimismo la creación de uno.



# Método

```
const response = await axios.post('http://localhost:3002/api/devolucion', {  
  numeroFactura,  
  nombre: factura.nombre,  
  telefono: factura.telefono,  
  productos: productosArray,  
  total: factura.total,  
  metodoPago: factura.metodoPago,  
  comentarios: comentarios,  
});
```





# Método

```
const handleBuscarFactura = async () => {
  try {
    const response = await axios.get(`http://localhost:5000/api/factura/numeroFactura/${numeroFactura}`);
    if (response.data) {
      setFactura(response.data);
      setMensaje('Factura encontrada');

      const productosData = response.data.productos;

      if (Array.isArray(productosData)) {
        setProductos(productosData);
      } else if (typeof productosData === 'string') {
        setProductos(productosData.split(',').map(producto => producto.trim()));
      } else if (typeof productosData === 'object') {
        setProductos(Object.values(productosData));
      } else {
        setProductos([]);
      }
    }
  }
}
```



# Consumo

Devolución del producto

Buscar Factura



# Consumo

## Devolución del producto

100002

Buscar Factura

Factura localizada

### Detalles de la Devolución:

**Factura:** 100002

**Nombre:** Laura

**Teléfono:** 1234567

**Total:** \$ 45500.00

**Método de pago:** Efectivo

Seleccione un producto para devolver: Seleccionar un producto

producto pequeño

Devolver el producto



# Prototipo



# Control de versiones

 Link



# THANKS!

Luisa Castillo  
Valentina Diaz  
Valentina Vasquez  
Lorena Vallejo

