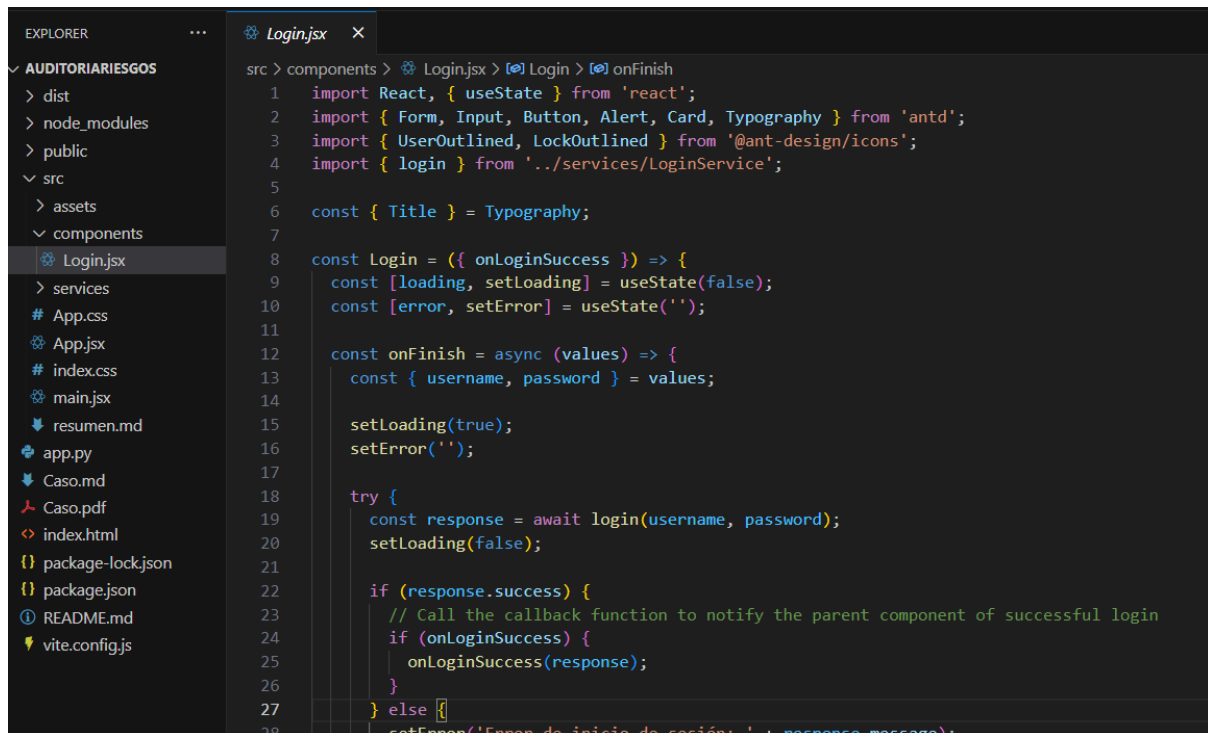


Frontend Sistema de login	3
Detalles Técnicos de la Implementación.....	4
1. Instalación de paquetes.....	4
2. Servicio de Autenticación (LoginService.js).....	4
3. Componente de Login (Login.jsx).....	4
4. Integración en App.jsx.....	5
5. Resultados.....	5
Backend Python y Ollama.....	6
1. Configuración básica:.....	6
2. Explicación de funcionamiento.....	6
2. Funciones principales:.....	6
3. Configuración del servidor.....	6
4. Resultados.....	6
Resultados finales.....	8
Iniciando sesión.....	8
Añadir 5 activos.....	8
Analizando los activos con el backend + ollama.....	9

Frontend Sistema de login

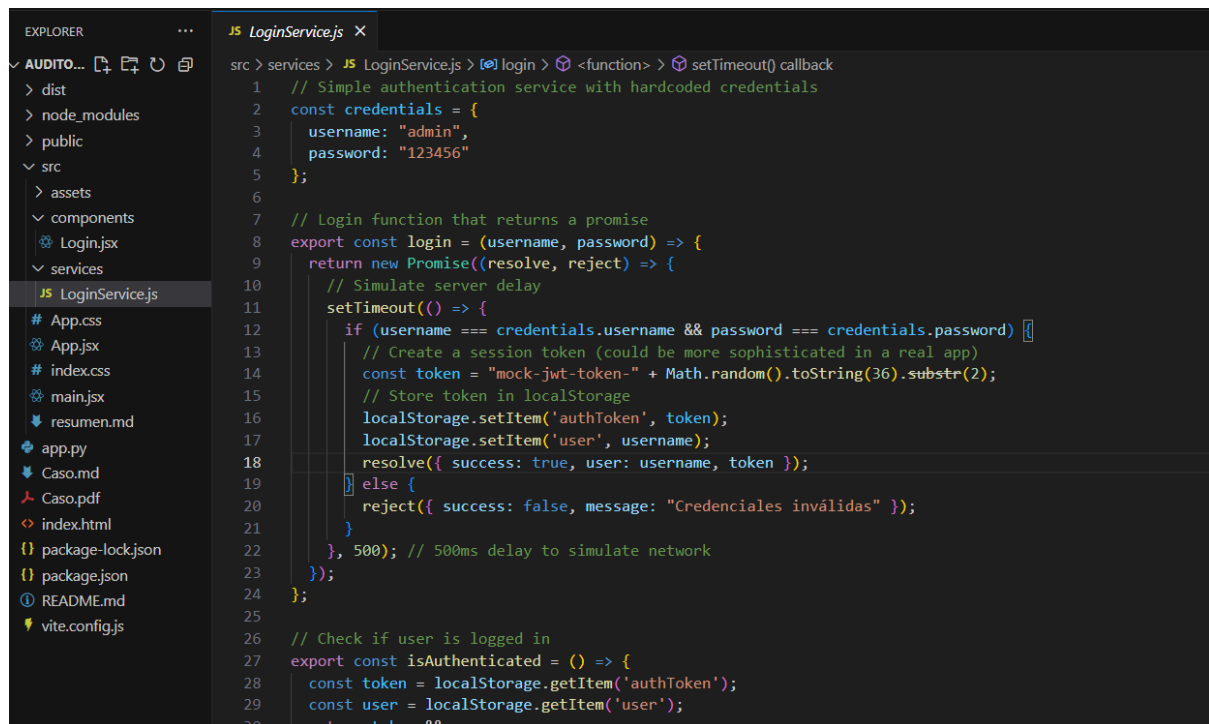
Para lograr el login se añadió los siguientes componentes y servicios

- Components/Login.jsx: Componente que maneja el formulario de inicio de sesión y la lógica de autenticación del usuario.



```
src > components > Login.jsx > Login > onFinish
1  import React, { useState } from 'react';
2  import { Form, Input, Button, Alert, Card, Typography } from 'antd';
3  import { UserOutlined, LockOutlined } from '@ant-design/icons';
4  import { login } from '../services/LoginService';
5
6  const { Title } = Typography;
7
8  const Login = ({ onLoginSuccess }) => {
9    const [loading, setLoading] = useState(false);
10   const [error, setError] = useState('');
11
12   const onFinish = async (values) => {
13     const { username, password } = values;
14
15     setLoading(true);
16     setError('');
17
18     try {
19       const response = await login(username, password);
20       setLoading(false);
21
22       if (response.success) {
23         // Call the callback function to notify the parent component of successful login
24         if (onLoginSuccess) {
25           onLoginSuccess(response);
26         }
27       } else {
28         setError('Error de inicio de sesión: ' + response.message);
```

- Services/LoginService.js: Servicio que proporciona funciones para iniciar sesión, verificar autenticación y cerrar sesión.



```
1 // Simple authentication service with hardcoded credentials
2 const credentials = {
3   username: "admin",
4   password: "123456"
5 };
6
7 // Login function that returns a promise
8 export const login = (username, password) => {
9   return new Promise((resolve, reject) => {
10     // Simulate server delay
11     setTimeout(() => {
12       if (username === credentials.username && password === credentials.password) {
13         // Create a session token (could be more sophisticated in a real app)
14         const token = "mock-jwt-token-" + Math.random().toString(36).substr(2);
15         // Store token in localStorage
16         localStorage.setItem('authToken', token);
17         localStorage.setItem('user', username);
18         resolve({ success: true, user: username, token });
19       } else {
20         reject({ success: false, message: "Credenciales inválidas" });
21       }
22     }, 500); // 500ms delay to simulate network
23   });
24 };
25
26 // Check if user is logged in
27 export const isAuthenticated = () => {
28   const token = localStorage.getItem('authToken');
29   const user = localStorage.getItem('user');
30   return token && user;
```

Detalles Técnicos de la Implementación

1. Instalación de paquetes

- npm install

2. Servicio de Autenticación (LoginService.js)

- Autenticación Simulada: Verifica las credenciales contra valores hardcodeados.
- Token de Sesión: Genera un token aleatorio como simulación de un JWT.
- Almacenamiento: Utiliza localStorage para mantener la sesión activa entre recargas de página.
- Funciones Expuestas:
 - login(username, password): Verifica credenciales y crea sesión
 - isAuthenticated(): Verifica si existe una sesión activa
 - logout(): Elimina la sesión activa

3. Componente de Login (Login.jsx)

- Estado del Formulario: Maneja los campos de usuario y contraseña.
- Validación: Verifica que se ingresen todos los campos requeridos.
- Manejo de Errores: Muestra mensajes de error en caso de credenciales inválidas.
- Feedback Visual: Indica al usuario cuando se está procesando el inicio de sesión.
- Notificación: Informa al padre del componente cuando el login es exitoso.

4. Integración en App.jsx

- Estado de autenticación: Utiliza useState para mantener el estado de autenticación.
- Renderizado Condicional: Muestra el login o la aplicación según el estado de autenticación.
- Protección de Rutas: Verifica la autenticación antes de mostrar contenido protegido.
- Persistencia: Verifica el estado de autenticación en localStorage al iniciar.

5. Resultados

Levantamos el servicio con el comando

- **npm run dev**

Sesión cerrada correctamente

Sistema de Auditoría de Riesgos

Ingresa tus credenciales para acceder

Usuario

Contraseña

Iniciar Sesión

Usuario de demo: admin

Contraseña: 123456

Sistema de Auditoría de Riesgos

Bienvenido, admin!

admin Cerrar Sesión

Agregar activo

Recomendar tratamientos

Activo	Riesgo	Impacto	Tratamiento	Operación
Servidor de base de datos	Pérdida de Servidor de base de datos	Pérdida de información valiosa relacionada con Servidor de base de datos	Monitoreo continuo de accesos	Eliminar
API Transacciones	Pérdida de API Transacciones	Pérdida de información valiosa relacionada con API Transacciones	Copias de seguridad periódicas	Eliminar
Aplicación Web de Banca	Pérdida de Aplicación Web de Banca	Pérdida de información valiosa relacionada con Aplicación Web de Banca	Implementación de firewall de nueva generación	Eliminar
Servidor de Correo	Pérdida de Servidor de Correo	Pérdida de información valiosa relacionada con Servidor de Correo	Cifrado de datos sensibles	Eliminar
Firewall Perimetral	Pérdida de Firewall Perimetral	Pérdida de información valiosa relacionada con Firewall Perimetral	Monitoreo continuo de accesos	Eliminar

< 1 >

Backend Python y Ollama

1. Configuración básica:

- Instalar los siguientes paquetes
 - `pip install flask`
 - `pip install openai`
- Ejecutar el servicio Python
 - `python app.py`

2. Explicacion de funcionamiento

- Se conecta a un modelo de IA local a través de Ollama (en localhost:11434)
- Endpoints API:
 - `'/analizar-riesgos'` (POST):
 - Recibe un activo tecnológico

2. Funciones principales:

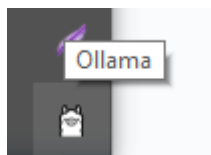
- `obtener_riesgos()`: Utiliza IA para generar 5 posibles riesgos para un activo dado
- `obtener_tratamiento()`: Utiliza IA para sugerir tratamientos específicos para un riesgo

3. Configuración del servidor

- Ollama corre en el puerto 11434
- Backend python en el puerto 5000

4. Resultados

- Ollama en ejecución



- Backend python en ejecución
Comando: `python app.py`

```
def obtener_riesgos( activo ):
    response = client.chat.completions.create(
        model="ramiro:instruct",
        messages=[
            {"role": "system", "content": "Responde en español, eres una herramienta para gestion de riesgos de la iso 27000, el usuario, te ingresara un asset te"},
            {"role": "user", "content": "mi raspberry pi"},
            {"role": "assistant", "content": ""}
        ]
    )

    """* **Acceso no autorizado***: terceros pueden acceder a la información almacenada o procesada en el Raspberry Pi sin"""

    """* **Pérdida o daño de datos***: los archivos y datos almacenados en el Raspberry Pi se pierden o dañan debido a un error en el sistema, un fallo en el hardware"""

    """* **Vulnerabilidades de seguridad***: El software o firmware instalados en el Raspberry Pi contienen vulnerabilidades de seguridad no detectadas y son explotables"""

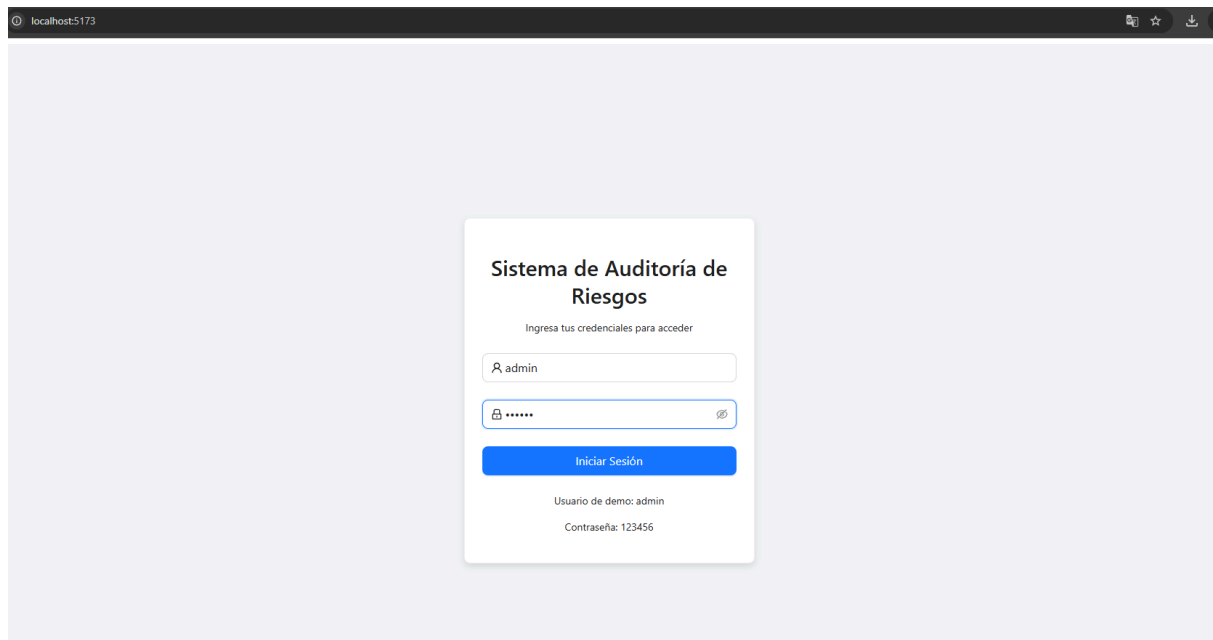
    """* **Inseguridad de la conexión***: la conexión del Raspberry Pi a la red local o internet no esté segura y un atacante intercepta datos confidenciales o información sensible"""

    """* **Fallos hardware***: daño debido a causas como sobrecalentamiento, sobrecarga eléctrica o errores en la manufactura, lo que lleva a una pérdida de datos"""

    answer = response.choices[0].message.content
    patron = r'\\*\\*\\s*(.+?)\\*\\*:\\s*(.+?)\\.?(?!=\\s\\n|\\s*$)'
```

Resultados finales

Iniciando sesión

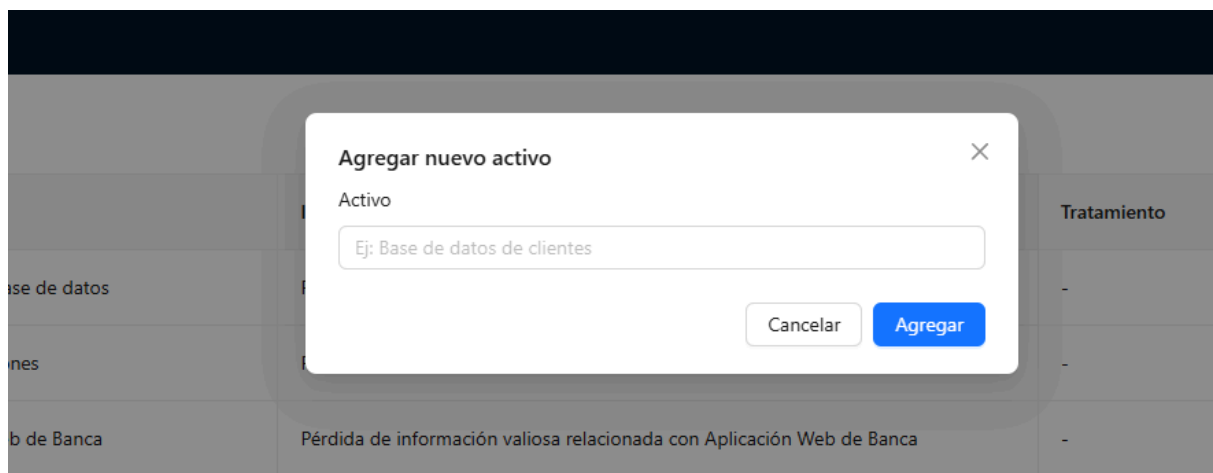


The screenshot shows a web browser window with the address bar displaying 'localhost:5173'. The main content area features a light gray background with a white login form in the center. The form is titled 'Sistema de Auditoría de Riesgos' and includes the instruction 'Ingresa tus credenciales para acceder'. It contains two input fields: the first is labeled 'admin' and the second is masked with '*****'. Below these fields is a blue button labeled 'Iniciar Sesión'. At the bottom of the form, it states 'Usuario de demo: admin' and 'Contraseña: 123456'.

Añadir 5 activos

Una vez logueado debemos añadir 5 activos

para esto usaremos el botón de “Agregar activo”



The screenshot shows a modal dialog titled 'Agregar nuevo activo' with a close button (X) in the top right corner. The dialog has a label 'Activo' and a text input field containing the placeholder text 'Ej: Base de datos de clientes'. At the bottom right of the dialog are two buttons: 'Cancelar' and 'Agregar'. The background is a blurred view of a table with columns and data.

Sistema de Auditoría de Riesgos

admin

Cerrar Sesión

+ Agregar activo

Recomendar tratamientos

Activo	Riesgo	Impacto	Tratamiento	Operación
Servidor de base de datos	Pérdida de Servidor de base de datos	Pérdida de información valiosa relacionada con Servidor de base de datos	-	Eliminar
API Transacciones	Pérdida de API Transacciones	Pérdida de información valiosa relacionada con API Transacciones	-	Eliminar
Aplicación Web de Banca	Pérdida de Aplicación Web de Banca	Pérdida de información valiosa relacionada con Aplicación Web de Banca	-	Eliminar
Servidor de Correo	Pérdida de Servidor de Correo	Pérdida de información valiosa relacionada con Servidor de Correo	-	Eliminar
Firewall Perimetral	Pérdida de Firewall Perimetral	Pérdida de información valiosa relacionada con Firewall Perimetral	-	Eliminar

< 1 >

Analizando los activos con el backend + ollama

Una vez agregado los 5 activos debemos de Recomendar tratamientos

Presionamos el botón de “Recomendar tratamientos”

Sistema de Auditoría de Riesgos

Tratamientos recomendados con éxito

admin

Cerrar Sesión

+ Agregar activo

Recomendar tratamientos

Activo	Riesgo	Impacto	Tratamiento	Operación
Servidor de base de datos	Pérdida de Servidor de base de datos	Pérdida de información valiosa relacionada con Servidor de base de datos	Monitoreo continuo de accesos	Eliminar
API Transacciones	Pérdida de API Transacciones	Pérdida de información valiosa relacionada con API Transacciones	Copias de seguridad periódicas	Eliminar
Aplicación Web de Banca	Pérdida de Aplicación Web de Banca	Pérdida de información valiosa relacionada con Aplicación Web de Banca	Implementación de firewall de nueva generación	Eliminar
Servidor de Correo	Pérdida de Servidor de Correo	Pérdida de información valiosa relacionada con Servidor de Correo	Cifrado de datos sensibles	Eliminar
Firewall Perimetral	Pérdida de Firewall Perimetral	Pérdida de información valiosa relacionada con Firewall Perimetral	Monitoreo continuo de accesos	Eliminar

< 1 >