Auditoría de Sistemas

Evaluación de Activos de Información en un Banco

Instrucciones Generales

Como auditor externo contratado por un banco, debe evaluar los activos de información, utilizando modelos de lenguaje avanzados ejecutados localmente. Este proceso incluye la generación automática de perfiles de riesgo, análisis de impactos, recomendaciones de mitigación alineadas con ISO 27001 y una interfaz intuitiva para la gestión de casos identificados.

Por tanto deberá:

. Crear su propio repositorio en GitHub (fork o clon del repositorio base) y subir todo el código fuente mejorado.

. Clonar y ejecutar el repositorio base proporcionado. 🔗 URL GitHub

. Modificar el sistema para incluir una funcionalidad de inicio de sesión ficticio sin base de datos y mejorar el motor de IA en el código.

. Evaluar 5 activos de información del entorno bancario (Lista Anexo 1).

. Elaborar un informe de auditoría, según instrucciones del apartado "Entregable", este informe estará desarrollado en el propio README.md de su proyecto y EXPORTADO a PDF para ser entregado en el aula virtual.

Entregable

Copie y pegue la siguiente estructura en su README.md para que al términar todo el informe de auditoría lo convierta en PDF y suba al aula virtual.

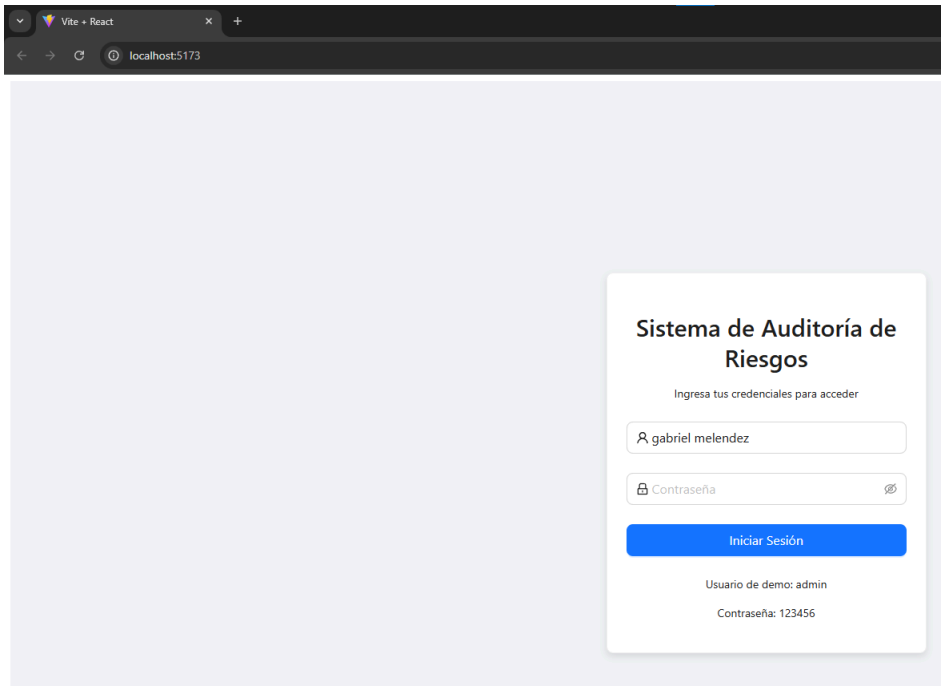
Informe de Auditoría de Sistemas - Examen de la Unidad I

Nombres y apellidos: Gabriel Fari Melendez Huarachi
Fecha: 10/09/2025
URL GitHub: https://github.com/GabrielFMH/EXAMENU1_AUDITORIA.git

1. Proyecto de Auditoría de Riesgos

Login

- Evidencia:



[Captura del login]

- Descripción: El sistema de login descrito opera sin una base de datos real, empleando credenciales hardcodeadas y almacenamiento local (localStorage) para la autenticación y gestión de sesiones.

Funcionamiento del Sistema de Login:

Credenciales Hardcodeadas: En src/services/LoginService.js, las credenciales (username: "admin", password: "123456") están fijas en el código, sin conexión a una base de datos externa, siendo adecuadas para demostraciones o prototipos, pero no para producción debido a su inseguridad.

Proceso de Login:

El componente Login (en src/components/Login.jsx) gestiona un formulario de usuario y contraseña.

Al enviar el formulario, la función onFinish invoca login(username, password) de LoginService.js.

Esta función simula un retardo de red con setTimeout y compara los datos ingresados con las credenciales hardcodeadas.

Si coinciden, se genera un token mock y se guarda en localStorage junto con el nombre de usuario (authToken y user).

Si el login es exitoso, onLoginSuccess actualiza el estado de la aplicación, marcándola como autenticada y mostrando un mensaje de bienvenida.

Verificación de Autenticación: App.jsx comprueba el estado inicial de autenticación mediante isAuthenticated() de LoginService.js, que verifica la presencia de authToken y user en localStorage. Si están presentes, el usuario se considera autenticado.

Logout: Al cerrar sesión desde App.jsx, la función handleLogout llama a logout() de LoginService.js, que elimina authToken y user de localStorage, restableciendo el estado de la aplicación para mostrar la pantalla de login nuevamente.

Consideraciones Clave:

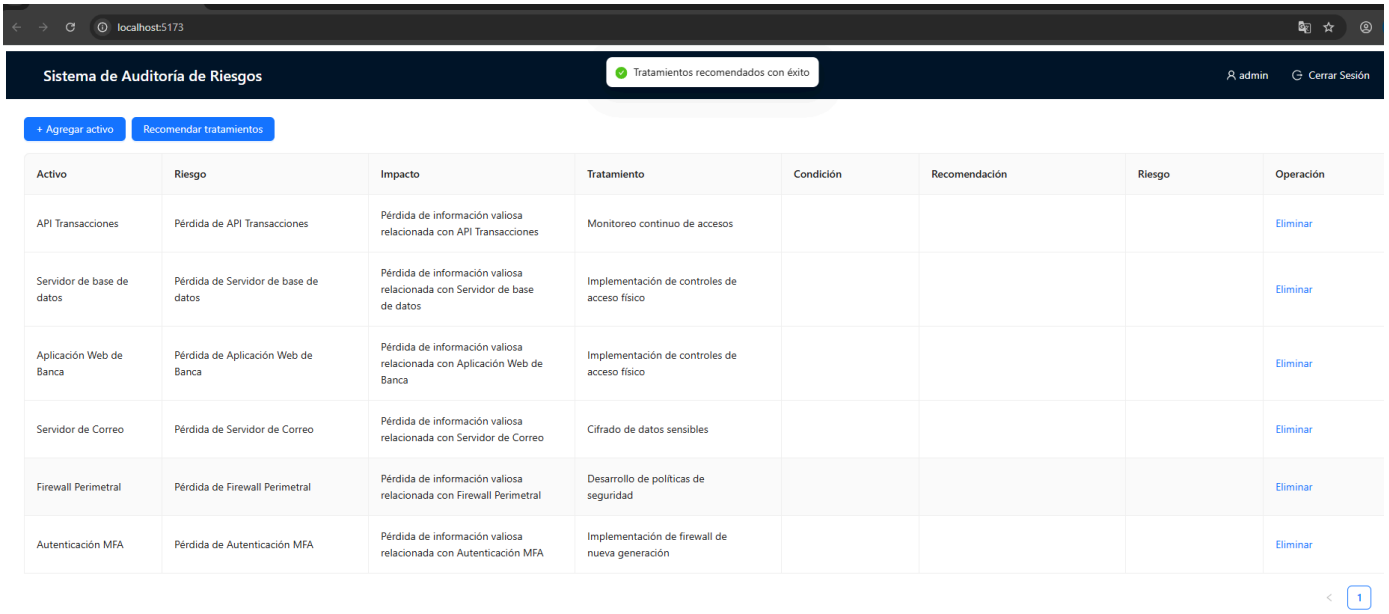
Seguridad: Este enfoque es altamente inseguro para entornos de producción, ya que las credenciales están expuestas directamente en el código fuente y carecen de cualquier tipo de encriptación.

Persistencia: Los datos de sesión no son persistentes y se pierden si el navegador se cierra o si localStorage se limpia, debido a la ausencia de un backend de almacenamiento persistente.

Simulación: El sistema simula interacciones con una API a través de setTimeout, pero no realiza llamadas reales a un servidor.

Motor de Inteligencia Artificial

- Evidencia:




```python
from openai import OpenAI
from flask import Flask, send_from_directory, request, jsonify, Response
import re
import logging

# Configurar logging
logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s - %(message)s')


app = Flask(__name__)

@app.after_request
def after_request(response):
    response.headers.add('Access-Control-Allow-Origin', '*')
    response.headers.add('Access-Control-Allow-Headers', 'Content-Type,Authorization')
    response.headers.add('Access-Control-Allow-Methods', 'GET,PUT,POST,DELETE,OPTIONS')
    return response

# Ruta para servir el index.html desde la carpeta dist
@app.route('/',  methods=["GET",'POST'])
def serve_index():
    return send_from_directory('dist', 'index.html')

# Ruta para servir los archivos estáticos generados
@app.route('/<path:path>')
def serve_static(path):
    return send_from_directory('dist', path)

client = OpenAI(
    base_url = 'http://localhost:11434/v1',
    api_key='ollama', # required, but unused
)

@app.route('/analizar-riesgos', methods=['POST'])
def analizar_riesgos():
    try:
        data = request.get_json()  # Obtener datos JSON enviados al endpoint
        if not data:
            return jsonify({"error": "Cuerpo de la solicitud debe ser JSON"}), 400
        activo = data.get('activo')  # Extraer el valor del activo
        if not activo or not isinstance(activo, str) or len(activo.strip()) == 0:
            return jsonify({"error": "El campo 'activo' es necesario y debe ser una cadena no vacía"}), 400

        riesgos, impactos = obtener_riesgos(activo.strip())  # Llamar a la función para obtener riesgos e impactos
        if not riesgos:
            return jsonify({"error": "No se pudieron generar riesgos para el activo proporcionado"}), 500
```

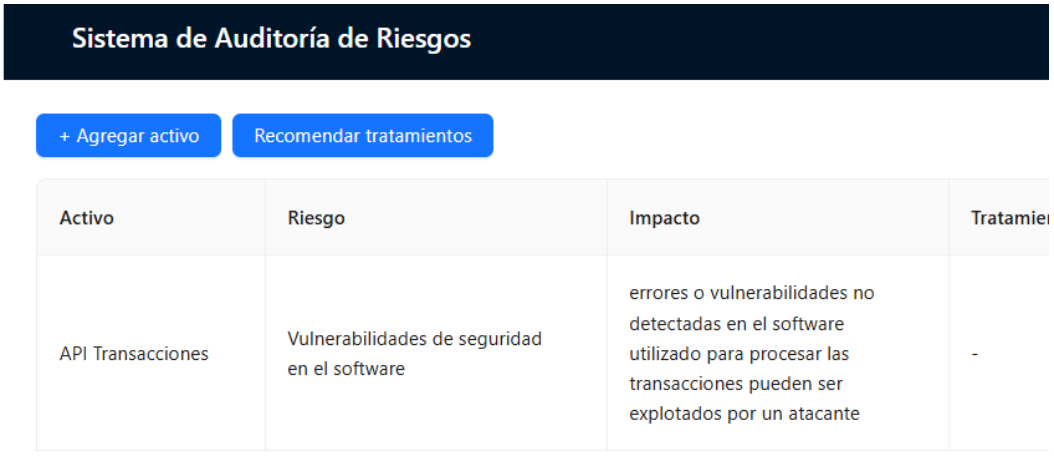
```python
def obtener_tratamiento(entrada_tratamiento):
    try:
        logging.info(f"Solicitando tratamiento para: {entrada_tratamiento}")
        response = client.chat.completions.create(
            model="llama2:7b",
            messages=[
                {"role": "system", "content": "Eres un experto en gestión de riesgos según ISO 27001. Responde en español. El usuario proporcionará un activo tecnológico, un riesgo y un impacto separados por ';'. Debes respond
                {"role": "user", "content": "mi telefono movil;Acceso no autorizado;un atacante puede acceder a la información personal y confidencial almacenada en el teléfono móvil, como números de teléfono, correos electrón
                {"role": "assistant", "content": "Condición: Acceso no autorizado al teléfono móvil permitiendo la exposición de datos personales.\nRecomendación: Establecer un bloqueo de la pantalla de inicio que requiera aut
                {"role": "user", "content": "servidor web;Inyección SQL;un atacante puede ejecutar consultas maliciosas en la base de datos, comprometiendo la integridad de los datos"},
                {"role": "assistant", "content": "Condición: Vulnerabilidad a inyección SQL en el servidor web que permite ejecución de consultas maliciosas.\nRecomendación: Implementar validación de entradas y uso de prepared
                {"role": "user", "content": entrada_tratamiento}
            ]
        )
        answer = response.choices[0].message.content
        logging.info(f"Tratamiento generado: {answer}")

        # Parsear la respuesta para extraer Condición, Recomendación y Riesgo
        patron_condicion = r'Condición:\s*(.*?)(?=\nRecomendación:|$)'
        patron_recomendacion = r'Recomendación:\s*(.*?)(?=\nRiesgo:|$)'
        patron_riesgo = r'Riesgo:\s*Probabilidad\s*(Baja|Media|Alta)'

        match_condicion = re.search(patron_condicion, answer, re.DOTALL)
        match_recomendacion = re.search(patron_recomendacion, answer, re.DOTALL)
        match_riesgo = re.search(patron_riesgo, answer)

        if match_condicion and match_recomendacion and match_riesgo:
            condicion = match_condicion.group(1).strip()
            recomendacion = match_recomendacion.group(1).strip()
            riesgo = match_riesgo.group(1).strip()
            return {
                "condicion": condicion,
                "recomendacion": recomendacion,
                "riesgo": riesgo
```

- Descripción: Se modifico la obtención de tratamientos para utilizar directamente la api flask en lugar de usar el mock de tratamientos, mediante la llamada /analizar-riesgos se hace uso del modelo local para que de su respuesta.

2.Hallazgos

Activo 1: Api Transacciones

- Evidencia:



- Condición: Vulnerabilidad de riesgos
- Recomendación: (Acción correctiva o preventiva) • Riesgo: Probabilidad (Baja/Media/Alta)

Activo 2: (título del activo)

Activo 3: (título del activo)

Activo 4: (título del

activo)

Activo 5: (título del activo)

Anexo 1: Activos de información

#	Activo	Tipo
1	Servidor de base de datos	Base de Datos
2	API Transacciones	Servicio Web
3	Aplicación Web de Banca	Aplicación
4	Servidor de Correo	Infraestructura
5	Firewall Perimetral	Seguridad
6	Autenticación MFA	Seguridad
7	Registros de Auditoría	Información
8	Backup en NAS	Almacenamiento
9	Servidor DNS Interno	Red
10	Plataforma de Pagos Móviles	Aplicación
11	VPN Corporativa	Infraestructura
12	Red de Cajeros Automáticos	Infraestructura
13	Servidor FTP	Red
14	CRM Bancario	Aplicación
15	ERP Financiero	Aplicación

#	Activo	Tipo
16	Base de Datos Clientes	Información
17	Logs de Seguridad	Información
18	Servidor Web Apache	Infraestructura
19	Consola de Gestión de Incidentes	Seguridad
20	Políticas de Seguridad Documentadas	Documentación
21	Módulo KYC (Know Your Customer)	Aplicación
22	Contraseñas de Usuarios	Información
23	Dispositivo HSM	Seguridad
24	Certificados Digitales SSL	Seguridad
25	Panel de Administración de Usuarios	Aplicación
26	Red Wi-Fi Interna	Red
27	Sistema de Control de Acceso Físico	Infraestructura
28	Sistema de Video Vigilancia	Infraestructura
29	Bot de Atención al Cliente	Servicio Web
30	Código Fuente del Core Bancario	Información
31	Tabla de Usuarios y Roles	Información
32	Documentación Técnica	Documentación
33	Manuales de Usuario	Documentación
34	Script de Backups Automáticos	Seguridad
35	Datos de Transacciones Diarias	Información
36	Herramienta SIEM	Seguridad
37	Switches y Routers	Red
38	Plan de Recuperación ante Desastres	Documentación
39	Contratos Digitales	Información Legal
40	Archivos de Configuración de Servidores	Información
41	Infraestructura en la Nube	Infraestructura
42	Correo Electrónico Ejecutivo	Información
43	Panel de Supervisión Financiera	Aplicación
44	App Móvil para Clientes	Aplicación
45	Token de Acceso a APIs	Seguridad

#	Activo	Tipo
46	Base de Datos Histórica	Información
47	Entorno de Desarrollo	Infraestructura
48	Sistema de Alertas de Seguridad	Seguridad
49	Configuración del Cortafuegos	Seguridad
50	Redundancia de Servidores	Infraestructura

Anexo 2: Rúbrica de Evaluación

La nota final es la suma de todos los criterios (máx. 20 puntos).

Criterio	0 pts	5 pts	Puntaje Máximo
Login	No presenta evidencia o está incorrecto	Login ficticio completo, funcional y con evidencia clara	5
IA Funcionando	No presenta IA o está incorrecta	IA implementada, funcionando y con evidencia clara	5
Evaluación de 5 Activos	Menos de 5 activos evaluados o sin hallazgos válidos	5 activos evaluados con hallazgos claros y evidencias	5
Informe claro y completo	Informe ausente, incompleto o poco entendible	Informe bien estructurado y completo según lo requerido	5