Escuela Universitaria
Politécnica - La Almunia
Centro adscrito
Universidad Zaragoza

ESCUELA UNIVERSITARIA POLITÉCNICA
DE LA ALMUNIA DE DOÑA GODINA (ZARAGOZA)

ANEXOS

Ciberseguridad en entornos Cloud: estrategias de protección y transferencia segura de datos

Cybersecurity in Cloud Environments: Protection Strategies and Secure Data Transfer

625.25.51

Autor: Jesús Gabriel García Salomón

Director: Eduardo Peris Millán

Fecha: 07 2025

Página intencionadamente en blanco.

INDICE DE CONTENIDO

CÓDIGO FUENTE

1. CÓDIGO FUENTE

1.1. APLICACIÓN PRINCIPAL

```
from flask import Flask, render_template, request, redirect, url_for, session, flash, send_file
from datetime import datetime
import psycopg2
import requests
from db.dynamodb_connection import logs_table, get_logs  # Importa la tabla
from db.rds_connection import RDSConnection
from db.s3_connection import listar_documentos_usuario, generar_url_descarga, get_file_from_s3
import io
import json
app = Flask(__name__)
app.secret_key = 'tu_clave_secreta_super_segura'  # Necesaria para manejar sesiones


def registrar_log(username, success):
    timestamp = datetime.utcnow().isoformat()
    ip_address = request.remote_addr
    user_agent = request.headers.get('User-Agent')


    try:
        location = requests.get(f'https://ipapi.co/{ip_address}/city/').text
    except:
        location = "Desconocida"


    logs_table.put_item(Item={
```

```
            'username': username,
            'timestamp': timestamp,
            'ip_address': ip_address,
            'user_agent': user_agent,
            'success': success,
            'location': location
        })

    @app.route("/")
    def index():
        if 'username' in session:
            # Redirigir según el rol del usuario
            if session.get('role') == 'admin':
                return redirect(url_for('admin_panel'))
            else:
                return redirect(url_for('dashboard'))
        return redirect(url_for('login'))

    @app.route("/login", methods=['GET', 'POST'])
    def login():
        if request.method == 'POST':
            username = request.form['username']
            password = request.form['password']

            try:
                # ⚠ Solo para pruebas educativas
                user = RDSConnection.execute_query(
                    f"SELECT * FROM users WHERE username =
'{username}' AND password = '{password}'",
                    fetch_one=True
                )

                if user:
```

```
                stored_password = user[3]
                if password == stored_password:
                    session['username'] = username
                    # IMPORTANTE: Guardar el rol en la sesión
                    # Asumiendo que el rol está en el índice 7 de la tabla
users
                    # Ajusta el índice según tu estructura de base de
datos
                    session['role'] = user[7] if len(user) > 7 else 'user'

                    registrar_log(username, True)  # Éxito
                    flash('Inicio de sesión exitoso', 'success')

                    # Redirigir según el rol
                    if session['role'] == 'admin':
                        return redirect(url_for('admin_panel'))
                    else:
                        return redirect(url_for('dashboard'))
                else:
                    registrar_log(username, False)  # Fallo
                    flash('Contraseña incorrecta', 'danger')
            else:
                registrar_log(username, False)  # Fallo
                flash('Usuario no encontrado', 'danger')

        except Exception as e:
            flash(f'Error al conectar con la base de datos: {str(e)}',
'danger')

    return render_template('login.html')

def admin_required():
    return session.get("role") == "admin"
```

```
@app.route("/admin")
def admin_panel():
    if not admin_required():
        flash('Acceso denegado. Se requieren privilegios de administrador.', 'danger')
        return redirect(url_for('login'))

    # Obtener datos del usuario admin para mostrar en el panel
    try:
        user_data = RDSConnection.get_user_by_username(session['username'])
        return render_template("admin.html", user_data=user_data)
    except Exception as e:
        flash(f'Error al cargar el panel de administración: {str(e)}', 'danger')
        return redirect(url_for('login'))

@app.route('/dashboard')
def dashboard():
    if 'username' not in session:
        return redirect(url_for('login'))

    # Verificar que no sea admin (los admin van a su panel específico)
    if session.get('role') == 'admin':
        return redirect(url_for('admin_panel'))

    try:
        user_data = RDSConnection.get_user_by_username(session['username'])

        if user_data:
            return render_template('dashboard.html',
```

```
                    username=user_data[1],
                    email=user_data[2],
                    full_name=user_data[4],
                    phone=user_data[5],
                    ssn=user_data[6],  # ¡Dato sensible!
                    api_key=user_data[8])
            else:
                return "Usuario no encontrado", 404

        except Exception as e:
            return f"Error de base de datos: {str(e)}", 500

    @app.route("/logout")
    def logout():
        session.clear()  # Limpiar toda la sesión incluyendo el rol
        flash('Has cerrado sesión correctamente', 'info')
        return redirect(url_for('login'))

    @app.route("/admin/logs")
    def admin_logs():
        if not admin_required():
            flash('Acceso denegado. Se requieren privilegios de
administrador.', 'danger')
            return redirect(url_for('login'))

        return render_template('admin_logs.html')

    @app.route("/api/logs")
    def api_logs():
        if not admin_required():
            return {'error': 'Acceso denegado'}, 403

        try:
```

```
            # Obtener parámetros
            limit = int(request.args.get('limit', 20))
            last_key = request.args.get('lastKey')

            # Convertir last_key si existe
            start_key = json.loads(last_key) if last_key else None

            # Obtener logs
            logs_data = get_logs(limit=limit, last_key=start_key)

            return {
                'logs': logs_data['items'],
                'lastKey': json.dumps(logs_data['last_key']) if
logs_data['last_key'] else None
            }

        except Exception as e:
            return {'error': str(e)}, 500

    @app.route('/mis-documentos')
    def mis_documentos():
        if 'username' not in session:
            return redirect(url_for('login'))

        username = session['username']
        documentos = listar_documentos_usuario(username)

        archivos = []
        for doc in documentos:
            url = generar_url_descarga(doc)
            archivos.append({
                'nombre': doc.split('/')[-1],
                'categoria': doc.split('/')[1],  # ej: dni, nominas
```

```
            'url': url
        })

    return render_template('documents.html', archivos=archivos,
username=username)

@app.route('/descargar/<tipo>/<nombre>')
def descargar_documento(tipo, nombre):
    if 'username' not in session:
        return redirect(url_for('login'))

    full_path = f"documentos/{tipo}/{nombre}"

    try:
        # Obtener el archivo binario desde S3
        file_stream = get_file_from_s3(full_path)

        # Detectar MIME según extensión
        if nombre.endswith('.pdf'):
            mimetype = 'application/pdf'
        elif nombre.endswith('.jpeg') or nombre.endswith('.jpg'):
            mimetype = 'image/jpeg'
        elif nombre.endswith('.xlsx'):
            mimetype = 'application/vnd.openxmlformats-
officedocument.spreadsheetml.sheet'
        else:
            mimetype = 'application/octet-stream'

        return send_file(
            io.BytesIO(file_stream.read()),
            mimetype=mimetype,
            as_attachment=True,
            download_name=nombre
```

```
        )
    except Exception as e:
        return f"Error al descargar el archivo: {str(e)}"

if __name__ == "__main__":
    app.run(debug=True, port=8080)
```

1.2. Conexiones a Bases de Datos

1.2.1. Conexión a Dynamodb

```
import boto3
from boto3.dynamodb.conditions import Key
from dotenv import load_dotenv
import os

access_key = os.getenv("AWS_DYNAMO_ACCESS_KEY_ID")
secret_key = os.getenv("AWS_DYNAMO_SECRET_ACCESS_KEY")

dynamodb = boto3.resource(
    'dynamodb',
    region_name='us-east-1',
    aws_access_key_id=access_key,
    aws_secret_access_key=secret_key
)

# Conexión a tabla
logs_table = dynamodb.Table('login_logs')
```

```
    def get_logs(limit=20, last_key=None):
        """Obtiene logs con paginación, ordenados por timestamp descendente"""
        try:
            # Configuración básica del scan
            scan_params = {
                'Limit': limit,
                'ReturnConsumedCapacity': 'TOTAL'
            }

            # Paginación
            if last_key:
                scan_params['ExclusiveStartKey'] = last_key

            # Ejecutar scan
            response = logs_table.scan(**scan_params)
            items = response.get('Items', [])

            # Ordenar los items por timestamp descendente (ya que scan no soporta orden nativo)
            items.sort(key=lambda x: x['timestamp'], reverse=True)

            return {
                'items': items,
                'last_key': response.get('LastEvaluatedKey')
            }

        except Exception as e:
            print(f"Error en get_logs: {str(e)}")
            raise
```

1.2.2. Conexión a S3

```
import boto3
from botocore.exceptions import ClientError
import urllib.parse
from dotenv import load_dotenv
import os

access_key = os.getenv("AWS_S3_ACCESS_KEY_ID")
secret_key = os.getenv("AWS_S3_SECRET_ACCESS_KEY")

# Parámetros de conexión (cámbialos por los tuyos)
s3 = boto3.client(
    's3',
    region_name='us-east-1',
    aws_access_key_id=access_key,
    aws_secret_access_key=secret_key
)

BUCKET_NAME = 'empresa-informes'

def listar_documentos_usuario(username):
    """
    Lista todos los documentos del usuario según su prefijo.
    """
    try:
        prefijo = f'documentos/'
        objetos = s3.list_objects_v2(Bucket=BUCKET_NAME,
Prefix=prefijo)

        documentos_usuario = []
        if 'Contents' in objetos:
```

```
            for obj in objetos['Contents']:
                key = obj['Key']
                # Se filtran documentos que contengan el nombre del usuario (ej: dni_jlopez.jpeg)
                if username in key:
                    documentos_usuario.append(key)
        return documentos_usuario
    except ClientError as e:
        print(f"[ERROR] Fallo al listar documentos: {e}")
        return []


def generar_url_descarga(key):
    """
    Genera una URL presignada para descargar el archivo.
    """
    try:
        url = s3.generate_presigned_url(
            'get_object',
            Params={'Bucket': BUCKET_NAME, 'Key': key},
            ExpiresIn=600  # Expira en 10 minutos
        )
        return url
    except Exception as e:
        print(f"[ERROR] No se pudo generar URL: {e}")
        return None


def get_file_from_s3(key):
    """
    Obtiene el archivo desde S3 como un objeto binario.
    """
    try:
        response = s3.get_object(Bucket=BUCKET_NAME, Key=key)
        return response['Body']  # Devuelve un StreamingBody
```

```
    except ClientError as e:
        print(f"[ERROR] No se pudo obtener el archivo de S3: {e}")
        raise
```

1.2.3. Conexión a RDS

```
import psycopg2
from psycopg2 import OperationalError

# Configuración de la base de datos- Se borraron las credenciales por temas de privacidad.
DB_CONFIG = {
    "host": "",
    "dbname": "",
    "user": "postgres",
    "password": ""
}

class RDSConnection:
    @staticmethod
    def create_connection():
        """Crea y retorna una conexión a la base de datos"""
        try:
            conn = psycopg2.connect(**DB_CONFIG)
            return conn
        except OperationalError as e:
            raise Exception(f"Error al conectar a RDS: {str(e)}")

    @staticmethod
    def execute_query(query, params=None, fetch_one=False, fetch_results=True):
        """Ejecuta una consulta y retorna resultados"""
```

```
        conn = None
        try:
            conn = RDSConnection.create_connection()
            cur = conn.cursor()

            cur.execute(query, params)

            # Solo intentar hacer fetch si se esperan resultados
            if fetch_results:
                if fetch_one:
                    result = cur.fetchone()
                else:
                    result = cur.fetchall()
            else:
                result = None

            conn.commit()
            return result

        except Exception as e:
            if conn:
                conn.rollback()
            raise Exception(f"Error en consulta: {str(e)}")
        finally:
            if conn:
                conn.close()

    @staticmethod
    def get_user_by_username(username):
        """Obtiene un usuario por su nombre de usuario"""
        query = "SELECT * FROM users WHERE username = %s"
        return RDSConnection.execute_query(query, (username,),
fetch_one=True)
```

```
@staticmethod
def test_connection():
    """Prueba la conexión a la base de datos"""
    try:
        conn = RDSConnection.create_connection()
        cur = conn.cursor()

        # 1. Verificar versión
        cur.execute("SELECT version();")
        version = cur.fetchone()

        # 2. Listar tablas
        cur.execute("""
            SELECT table_name
            FROM information_schema.tables
            WHERE table_schema = 'public'
            ORDER BY table_name;
        """)
        tables = cur.fetchall()

        return {
            "version": version[0],
            "tables": [table[0] for table in tables]
        }

    except Exception as e:
        raise Exception(f"Error en test de conexión: {str(e)}")
    finally:
        if conn:
            conn.close()
```

1.3. INTERFAZ DE USUARIO

1.3.1. Plantillas HTML

1.3.1.1. Plantilla base

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>TFG - Testing de Vulnerabilidades AWS</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 40px;
            background-color: #f4f4f4;
        }
        .container {
            background: white;
            padding: 20px;
            border-radius: 8px;
            box-shadow: 0 2px 10px rgba(0,0,0,0.1);
        }
        .nav {
            margin-bottom: 20px;
        }
        .nav a {
            margin-right: 15px;
            text-decoration: none;
```

```
            color: #007bff;
        }
        .nav a:hover {
            text-decoration: underline;
        }
        h1 {
            color: #333;
        }
        .warning {
            background: #fff3cd;
            border: 1px solid #ffeaa7;
            padding: 10px;
            border-radius: 4px;
            margin: 20px 0;
        }
    </style>
</head>
<body>
    <div class="container">
        <div class="nav">
            <a href="/">Inicio</a>
            <a href="/login">Login</a>
            <a href="/admin">Admin</a>
            <a href="/buscar">Buscar</a>
            <a href="/contacto">Contacto</a>
            <a href="/testdb">Test DB</a>
        </div>

        <h1>Plataforma de Testing - Vulnerabilidades AWS</h1>

        <div class="warning">
```

```
            <strong>⚠ ENTORNO DE TESTING:</strong> Esta aplicación contiene vulnerabilidades intencionadas para propósitos educativos.
        </div>

        <p>Bienvenido a la plataforma de testing de vulnerabilidades en servicios cloud de AWS.</p>

        <h2>Funcionalidades disponibles:</h2>
        <ul>
            <li><a href="/login">Sistema de Login (vulnerable a SQL injection)</a></li>
            <li><a href="/admin">Panel de Administración</a></li>
            <li><a href="/buscar">Búsqueda de usuarios</a></li>
            <li><a href="/contacto">Formulario de contacto</a></li>
            <li><a href="/testdb">Test de conexión a base de datos</a></li>
        </ul>

        <h2>Vulnerabilidades implementadas:</h2>
        <ul>
            <li>SQL Injection</li>
            <li>Configuraciones inseguras de RDS</li>
            <li>Credenciales hardcodeadas</li>
            <li>Debug mode activado</li>
        </ul>
    </div>
</body>
</html>
```

1.3.1.2. Página de login.

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CloudSecure ERP - Portal de Gestión</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='css/login.css') }}">
</head>
<body>
    <div class="left-panel">
        <div class="branding">
            <h1>CloudSecure ERP</h1>
            <p>Gestión empresarial segura en entornos cloud</p>
            <ul class="features">
                <li>Integración con AWS RDS, DynamoDB y S3</li>
                <li>Autenticación segura de usuarios</li>
                <li>Panel de administración centralizado</li>
                <li>Protección contra vulnerabilidades</li>
            </ul>
        </div>
    </div>

    <div class="right-panel">
        <div class="login-form">
            <div class="form-header">
                <h2>Inicio de Sesión</h2>
```

```
            <p>Ingrese sus credenciales para acceder al
sistema</p>
        </div>

        <form action="/login" method="POST">
            <div class="form-group">
                <label for="username">Usuario</label>
                <input type="text" id="username"
name="username" class="form-control" placeholder="jlopez" required>
            </div>

            <div class="form-group">
                <label for="password">Contraseña</label>
                <input type="password" id="password"
name="password" class="form-control" placeholder="••••••••"
required>
            </div>

            <div class="form-options">
                <label class="checkbox-wrapper">
                    <input type="checkbox" name="remember">
                    Recordar sesión
                </label>
                <a href="#" class="forgot-link">¿Olvidó su
contraseña?</a>
            </div>

            <button type="submit" class="btn-primary">Acceder al
sistema</button>
        </form>

        <div class="divider">
            <span>O autentíquese con</span>
        </div>
```

```
            <div class="alternative-login">
                <button class="alt-btn">🔑 AWS IAM</button>
                <button class="alt-btn">👥 LDAP</button>
            </div>
        </div>
    </div>
</body>
</html>
```

1.3.1.3. Dashboard principal.

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <title>CloudSecure ERP - Dashboard</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='css/dashboard.css') }}" />
</head>
<body>
    <div class="left-panel">
        <div class="branding">
            <h1>CloudSecure ERP</h1>
            <p>Soluciones integrales de gestión empresarial</p>
            <nav>
                <ul class="menu">
                    <li><a href="/">Inicio</a></li>
                    <li><a href="/admin">Administración</a></li>
                    <li><a href="/buscar">Buscar</a></li>
                    <li><a href="/contacto">Contacto</a></li>
                </ul>
```

```
            </nav>
        </div>
    </div>

    <div class="right-panel">
        <header class="header">
            <h2>Bienvenido, {{ username }}</h2>
            <form action="{{ url_for('logout') }}" method="get"
class="logout-form">
                <button type="submit" class="btn-logout">Cerrar
sesión</button>
            </form>
        </header>

        <section class="profile-info">
            <h3>Perfil de usuario</h3>
            <table>
                <tbody>
                    <tr>
                        <th>Nombre completo</th>
                        <td>{{ full_name }}</td>
                    </tr>
                    <tr>
                        <th>Email</th>
                        <td>{{ email }}</td>
                    </tr>
                    <tr>
                        <th>Teléfono</th>
                        <td>{{ phone }}</td>
                    </tr>
                    <tr>
                        <th class="sensitive">Número de seguridad
social</th>
```

```
                    <td class="sensitive">{{ ssn }}</td>
                </tr>
                <tr>
                    <th>API Key</th>
                    <td class="monospace">{{ api_key }}</td>
                </tr>
            </tbody>
        </table>
    </section>

    <!-- Nueva sección para botón de documentos -->
    <section class="actions">
        <a href="{{ url_for('mis_documentos') }}" class="btn-
documents">Ver mis documentos</a>
    </section>

</div>
</body>
</html>
```

1.3.1.4. Panel de administración.

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CloudSecure ERP - Panel de Administración</title>
    <style>
        /* Reset básico */
        * {
            box-sizing: border-box;
            margin: 0;
            padding: 0;
        }

        body {
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
            background-color: #f9fafb;
            color: #333;
            display: flex;
            min-height: 100vh;
        }

        .left-panel {
            background-color: #2f3e4d;
            color: #f0f3f5;
            width: 280px;
            padding: 40px 25px;
            display: flex;
            flex-direction: column;
            justify-content: flex-start;
```

```
}

.branding h1 {
    font-weight: 700;
    font-size: 1.9rem;
    margin-bottom: 8px;
    letter-spacing: 1px;
}

.branding p {
    font-size: 0.95rem;
    margin-bottom: 35px;
    color: #abb2bc;
}

nav ul.menu {
    list-style: none;
}

nav ul.menu li {
    margin-bottom: 18px;
}

nav ul.menu li a {
    text-decoration: none;
    color: #a7b1bb;
    font-size: 1.05rem;
    transition: color 0.3s ease;
    display: block;
    padding: 10px 15px;
    border-radius: 5px;
}
```

```
nav ul.menu li a:hover {
    color: #fff;
    background-color: rgba(255,255,255,0.1);
}

nav ul.menu li a.active {
    background-color: #3498db;
    color: #fff;
}

.right-panel {
    flex-grow: 1;
    background-color: #ffffff;
    padding: 40px 60px;
    box-shadow: -2px 0 10px rgba(0,0,0,0.05);
    display: flex;
    flex-direction: column;
}

.header {
    display: flex;
    justify-content: space-between;
    align-items: center;
    margin-bottom: 40px;
    padding-bottom: 20px;
    border-bottom: 2px solid #ecf0f1;
}

.header h2 {
    font-weight: 600;
    font-size: 1.7rem;
    color: #2c3e50;
}
```

```
        .admin-badge {
            background-color: #e74c3c;
            color: white;
            padding: 5px 12px;
            border-radius: 15px;
            font-size: 0.8rem;
            font-weight: 600;
            margin-left: 10px;
        }

        .logout-form button.btn-logout {
            background-color: #e74c3c;
            border: none;
            padding: 10px 22px;
            color: white;
            font-weight: 600;
            font-size: 1rem;
            border-radius: 5px;
            cursor: pointer;
            transition: background-color 0.25s ease;
        }

        .logout-form button.btn-logout:hover {
            background-color: #c0392b;
        }

        .admin-stats {
            display: grid;
            grid-template-columns: repeat(auto-fit, minmax(250px,
1fr));
            gap: 20px;
            margin-bottom: 40px;
```

```
        }

        .stat-card {
            background: linear-gradient(135deg, #667eea 0%,
#764ba2 100%);
            color: white;
            padding: 25px;
            border-radius: 10px;
            box-shadow: 0 4px 15px rgba(0,0,0,0.1);
        }

        .stat-card h3 {
            font-size: 2.2rem;
            margin-bottom: 5px;
        }

        .stat-card p {
            font-size: 1rem;
            opacity: 0.9;
        }

        .profile-info h3 {
            font-size: 1.3rem;
            margin-bottom: 20px;
            color: #34495e;
            font-weight: 600;
        }

        .profile-info table {
            width: 100%;
            border-collapse: collapse;
            background: white;
            border-radius: 8px;
```

```
    overflow: hidden;
    box-shadow: 0 2px 10px rgba(0,0,0,0.1);
}

.profile-info th, .profile-info td {
    padding: 14px 12px;
    border-bottom: 1px solid #ecf0f1;
    text-align: left;
    font-size: 1rem;
    color: #34495e;
}

.profile-info th {
    background-color: #f8f9fa;
    font-weight: 600;
}

.profile-info th.sensitive {
    color: #c0392b;
}

.profile-info td.sensitive {
    color: #c0392b;
    font-family: monospace;
    font-weight: 600;
}

.profile-info td.monospace {
    font-family: monospace;
    color: #2c3e50;
}

.admin-actions {
```

```
            margin-top: 30px;
            display: flex;
            gap: 15px;
            flex-wrap: wrap;
        }

        .btn-admin {
            background-color: #3498db;
            color: white;
            padding: 12px 20px;
            border: none;
            border-radius: 5px;
            cursor: pointer;
            font-weight: 600;
            text-decoration: none;
            transition: background-color 0.3s ease;
        }

        .btn-admin:hover {
            background-color: #2980b9;
        }

        .btn-danger {
            background-color: #e74c3c;
        }

        .btn-danger:hover {
            background-color: #c0392b;
        }
    </style>
</head>
<body>
    <div class="left-panel">
```

```
            <div class="branding">
                <h1>CloudSecure ERP</h1>
                <p>Panel de Administración</p>
            </div>

            <nav>
                <ul class="menu">
                    <li><a href="#" class="active">🏠
Dashboard</a></li>
                    <li><a href="#">👥 Gestión de Usuarios</a></li>
                    <li><a href="#">📊 Reportes</a></li>
                    <li><a href="#">🔒 Seguridad</a></li>
                    <li><a href="#">⚙️ Configuración</a></li>
                    <li><a href="/testdb">🔧 Test BD</a></li>
                    <li><a href="/testusers">👤 Test Usuarios</a></li>
                </ul>
            </nav>
        </div>

        <div class="right-panel">
            <div class="header">
                <div>
                    <h2>Panel de Administración</h2>
                    <span class="admin-badge">ADMINISTRADOR</span>
                </div>
                <form action="/logout" method="POST" class="logout-
form">
                    <button type="submit" class="btn-logout">Cerrar
Sesión</button>
                </form>
            </div>

            <div class="admin-stats">
```

```
    <div class="stat-card">
        <h3>156</h3>
        <p>Usuarios Activos</p>
    </div>
    <div class="stat-card">
        <h3>23</h3>
        <p>Intentos de Login Fallidos</p>
    </div>
    <div class="stat-card">
        <h3>99.8%</h3>
        <p>Disponibilidad del Sistema</p>
    </div>
</div>

<div class="profile-info">
    <h3>Información del Administrador</h3>
    <table>
        <tr>
            <th>Campo</th>
            <th>Valor</th>
        </tr>
        {% if user_data %}
        <tr>
            <td><strong>Usuario</strong></td>
            <td>{{ user_data[1] }}</td>
        </tr>
        <tr>
            <td><strong>Email</strong></td>
            <td>{{ user_data[2] }}</td>
        </tr>
        <tr>
            <td><strong>Nombre Completo</strong></td>
```

```
                    <td>{{ user_data[4] if user_data|length > 4 else
'N/A' }}</td>
                </tr>
                <tr>
                    <td><strong>Rol</strong></td>
                    <td><span style="color: #e74c3c; font-weight:
600;">{{ user_data[7] if user_data|length > 7 else 'admin'
}}</span></td>
                </tr>
                <tr>
                    <td class="sensitive"><strong>API
Key</strong></td>
                    <td class="monospace">{{ user_data[8] if
user_data|length > 8 else 'N/A' }}</td>
                </tr>
                {% else %}
                <tr>
                    <td colspan="2">Error al cargar datos del
usuario</td>
                </tr>
                {% endif %}
            </table>

            <div class="admin-actions">
                <a href="/admin/logs" class="btn-admin">📋 Ver Logs
del Sistema</a>
                <a href="#" class="btn-admin">👥 Gestionar
Usuarios</a>
                <a href="#" class="btn-admin">🔧 Configuración
AWS</a>
                <a href="#" class="btn-admin btn-danger">⚠️ Modo
Mantenimiento</a>
            </div>
        </div>
    </div>
```

```
</body>
</html>
```

1.3.1.5. Logs de administración.

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CloudSecure ERP - Logs del Sistema</title>
    <style>
        /* Reset básico */
        * {
            box-sizing: border-box;
            margin: 0;
            padding: 0;
        }

        body {
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
            background-color: #f9fafb;
            color: #333;
            display: flex;
            min-height: 100vh;
        }

        .left-panel {
            background-color: #2f3e4d;
            color: #f0f3f5;
            width: 280px;
            padding: 40px 25px;
```

```
    display: flex;
    flex-direction: column;
    justify-content: flex-start;
}

.branding h1 {
    font-weight: 700;
    font-size: 1.9rem;
    margin-bottom: 8px;
    letter-spacing: 1px;
}

.branding p {
    font-size: 0.95rem;
    margin-bottom: 35px;
    color: #abb2bc;
}

nav ul.menu {
    list-style: none;
}

nav ul.menu li {
    margin-bottom: 18px;
}

nav ul.menu li a {
    text-decoration: none;
    color: #a7b1bb;
    font-size: 1.05rem;
    transition: color 0.3s ease;
    display: block;
    padding: 10px 15px;
```

```
    border-radius: 5px;
}

nav ul.menu li a:hover {
    color: #fff;
    background-color: rgba(255,255,255,0.1);
}

nav ul.menu li a.active {
    background-color: #3498db;
    color: #fff;
}

.right-panel {
    flex-grow: 1;
    background-color: #ffffff;
    padding: 40px 60px;
    box-shadow: -2px 0 10px rgba(0,0,0,0.05);
    display: flex;
    flex-direction: column;
}

.header {
    display: flex;
    justify-content: space-between;
    align-items: center;
    margin-bottom: 30px;
    padding-bottom: 20px;
    border-bottom: 2px solid #ecf0f1;
}

.header h2 {
    font-weight: 600;
```

```
    font-size: 1.7rem;
    color: #2c3e50;
}

.controls {
    display: flex;
    gap: 15px;
    align-items: center;
    margin-bottom: 25px;
}

.btn-control {
    background-color: #3498db;
    color: white;
    padding: 8px 16px;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    font-weight: 600;
    transition: background-color 0.3s ease;
}

.btn-control:hover {
    background-color: #2980b9;
}

.btn-control.active {
    background-color: #27ae60;
}

.btn-danger {
    background-color: #e74c3c;
}
```

```
        .btn-danger:hover {
            background-color: #c0392b;
        }

        .stats-bar {
            display: flex;
            gap: 20px;
            margin-bottom: 25px;
            padding: 15px;
            background: linear-gradient(135deg, #667eea 0%,
#764ba2 100%);
            border-radius: 8px;
            color: white;
        }

        .stat-item {
            text-align: center;
            flex: 1;
        }

        .stat-number {
            font-size: 1.8rem;
            font-weight: 700;
            margin-bottom: 5px;
        }

        .stat-label {
            font-size: 0.9rem;
            opacity: 0.9;
        }

        .logs-container {
```

```
    background: white;
    border-radius: 8px;
    box-shadow: 0 2px 10px rgba(0,0,0,0.1);
    overflow: hidden;
    flex-grow: 1;
    display: flex;
    flex-direction: column;
}

.logs-header {
    background-color: #f8f9fa;
    padding: 15px 20px;
    border-bottom: 1px solid #dee2e6;
    font-weight: 600;
    color: #495057;
}

.logs-table {
    flex-grow: 1;
    overflow-y: auto;
    max-height: 60vh;
}

table {
    width: 100%;
    border-collapse: collapse;
}

th, td {
    padding: 12px 15px;
    text-align: left;
    border-bottom: 1px solid #dee2e6;
}
```

```
th {
    background-color: #f8f9fa;
    font-weight: 600;
    color: #495057;
    position: sticky;
    top: 0;
    z-index: 10;
}

.log-row {
    transition: background-color 0.2s ease;
}

.log-row:hover {
    background-color: #f8f9fa;
}

.log-row.new-log {
    background-color: #d4edda;
    animation: fadeIn 0.5s ease;
}

@keyframes fadeIn {
    from { background-color: #28a745; }
    to { background-color: #d4edda; }
}

.status-success {
    color: #28a745;
    font-weight: 600;
}
```

```
.status-failed {
    color: #dc3545;
    font-weight: 600;
}

.timestamp {
    font-family: monospace;
    font-size: 0.9rem;
    color: #6c757d;
}

.username {
    font-weight: 600;
    color: #2c3e50;
}

.ip-address {
    font-family: monospace;
    color: #6c757d;
}

.user-agent {
    max-width: 200px;
    overflow: hidden;
    text-overflow: ellipsis;
    white-space: nowrap;
    font-size: 0.85rem;
    color: #6c757d;
}

.location {
    color: #495057;
}
```

```
.loading {
    text-align: center;
    padding: 20px;
    color: #6c757d;
}

.load-more {
    text-align: center;
    padding: 20px;
    border-top: 1px solid #dee2e6;
}

.auto-refresh-indicator {
    display: inline-flex;
    align-items: center;
    gap: 8px;
    padding: 5px 10px;
    background-color: #28a745;
    color: white;
    border-radius: 15px;
    font-size: 0.8rem;
}

.pulse {
    width: 8px;
    height: 8px;
    background-color: white;
    border-radius: 50%;
    animation: pulse 1.5s infinite;
}

@keyframes pulse {
```

```
            0%, 100% { opacity: 1; }
            50% { opacity: 0.3; }
        }

        .back-btn {
            background-color: #6c757d;
            color: white;
            padding: 10px 20px;
            border: none;
            border-radius: 5px;
            cursor: pointer;
            font-weight: 600;
            text-decoration: none;
            display: inline-block;
            transition: background-color 0.3s ease;
        }

        .back-btn:hover {
            background-color: #5a6268;
        }
    </style>
</head>
<body>
    <div class="left-panel">
        <div class="branding">
            <h1>CloudSecure ERP</h1>
            <p>Sistema de Logs</p>
        </div>

        <nav>
            <ul class="menu">
                <li><a href="/admin">🏠 Dashboard</a></li>
```

```
                <li><a href="#" class="active">📋 Logs del
Sistema</a></li>
                <li><a href="#">👥 Gestión de Usuarios</a></li>
                <li><a href="#">📊 Reportes</a></li>
                <li><a href="#">🔒 Seguridad</a></li>
                <li><a href="#">⚙️ Configuración</a></li>
            </ul>
        </nav>
    </div>

    <div class="right-panel">
        <div class="header">
            <div>
                <h2>📋 Logs del Sistema</h2>
            </div>
            <a href="/admin" class="back-btn">← Volver al
Panel</a>
        </div>

        <div class="stats-bar">
            <div class="stat-item">
                <div class="stat-number" id="totalLogs">-</div>
                <div class="stat-label">Total Logs</div>
            </div>
            <div class="stat-item">
                <div class="stat-number" id="successfulLogins">-
</div>
                <div class="stat-label">Inicios Exitosos</div>
            </div>
            <div class="stat-item">
                <div class="stat-number" id="failedLogins">-</div>
                <div class="stat-label">Intentos Fallidos</div>
            </div>
```

```
                <div class="stat-item">
                    <div class="stat-number" id="lastUpdate">-</div>
                    <div class="stat-label">Última Actualización</div>
                </div>
            </div>

            <div class="controls">
                <button class="btn-control active" id="autoRefreshBtn"
onclick="toggleAutoRefresh()">
                    <span id="refreshIcon">⏸</span> Auto-actualizar
                </button>
                <button class="btn-control" onclick="refreshLogs()">🔄
Actualizar Ahora</button>
                <button class="btn-control btn-danger"
onclick="clearLogs()">🗑️ Limpiar Vista</button>
                <span class="auto-refresh-indicator" id="refreshIndicator"
style="display: none;">
                    <div class="pulse"></div>
                    Actualizando...
                </span>
            </div>

            <div class="logs-container">
                <div class="logs-header">
                    Registro de Inicios de Sesión - Tiempo Real
                </div>
                <div class="logs-table">
                    <table>
                        <thead>
                            <tr>
                                <th>Timestamp</th>
                                <th>Usuario</th>
                                <th>Estado</th>
                                <th>IP</th>
```

```
                        <th>Ubicación</th>
                        <th>User Agent</th>
                    </tr>
                </thead>
                <tbody id="logsTableBody">
                    <tr>
                        <td colspan="6" class="loading">
                            🔄 Cargando logs del sistema...
                        </td>
                    </tr>
                </tbody>
            </table>
        </div>
        <div class="load-more" id="loadMoreSection"
style="display: none;">
            <button class="btn-control"
onclick="loadMoreLogs()">Cargar más logs</button>
        </div>
    </div>
</div>

<script>
    let autoRefresh = true;
    let refreshInterval;
    let lastKey = null;
    let allLogs = [];
    let stats = {
        total: 0,
        successful: 0,
        failed: 0
    };

    // Inicializar la página
```

```
document.addEventListener('DOMContentLoaded', function()
{
    loadLogs();
    startAutoRefresh();
});

function startAutoRefresh() {
    if (refreshInterval) clearInterval(refreshInterval);

    if (autoRefresh) {
        refreshInterval = setInterval(() => {
            loadLogs(true);
        }, 5000); // Actualizar cada 5 segundos
    }
}

function toggleAutoRefresh() {
    autoRefresh = !autoRefresh;
    const btn = document.getElementById('autoRefreshBtn');
    const icon = document.getElementById('refreshIcon');

    if (autoRefresh) {
        btn.classList.add('active');
        icon.textContent = '⏸';
        startAutoRefresh();
    } else {
        btn.classList.remove('active');
        icon.textContent = '▶';
        clearInterval(refreshInterval);
    }
}

function refreshLogs() {
```

```
            loadLogs(true);
        }

        function clearLogs() {
            if (confirm('¿Estás seguro de que quieres limpiar la vista de logs? (No eliminará los logs de la base de datos)')) {
                allLogs = [];
                lastKey = null;
                document.getElementById('logsTableBody').innerHTML = '<tr><td colspan="6" class="loading">Vista limpiada. Haz clic en "Actualizar Ahora" para recargar.</td></tr>';
                updateStats();
            }
        }

        function loadLogs(isRefresh = false) {
            if (!isRefresh) {
                showRefreshIndicator();
            }

            let url = '/api/logs?limit=20';
            if (!isRefresh && lastKey) {
                url += `&lastKey=${encodeURIComponent(lastKey)}`;
            }

            fetch(url)
                .then(response => response.json())
                .then(data => {
                    if (data.error) {
                        throw new Error(data.error);
                    }

                    if (isRefresh) {
```

```
                    // Actualización: verificar nuevos logs
                    const newLogs = data.logs.filter(log =>
                        !allLogs.some(existingLog =>
                            existingLog.username === log.username &&
                            existingLog.timestamp === log.timestamp
                        )
                    );

                    if (newLogs.length > 0) {
                        allLogs = [...newLogs, ...allLogs];
                        renderLogs(true);
                    }
                } else {
                    // Carga inicial o paginación
                    allLogs = [...allLogs, ...data.logs];
                    renderLogs();
                }

                lastKey = data.lastKey || null;
                updateLoadMoreButton(data.hasMore);
                updateStats();
                updateLastUpdateTime();
                hideRefreshIndicator();
            })
            .catch(error => {
                console.error('Error loading logs:', error);

document.getElementById('logsTableBody').innerHTML =
                    `<tr><td colspan="6" class="loading"
style="color: #dc3545;">❌ Error: ${error.message}</td></tr>`;
                hideRefreshIndicator();
            });
        }
```

```
        function loadMoreLogs() {
            if (lastKey) {
                loadLogs();
            }
        }

        function renderLogs(highlightNew = false) {
            const tbody = document.getElementById('logsTableBody');
            const existingRowsCount = tbody.children.length;

            tbody.innerHTML = '';

            allLogs.forEach((log, index) => {
                const row = document.createElement('tr');
                row.className = 'log-row';

                if (highlightNew && index < (allLogs.length -
existingRowsCount)) {
                    row.classList.add('new-log');
                }

                const timestamp = formatTimestamp(log.timestamp);
                const status = log.success ?
                    '<span class="status-success">✅ Exitoso</span>' :
                    '<span class="status-failed">❌ Fallido</span>';
                const userAgent = truncateText(log.user_agent, 30);

                row.innerHTML = `
                    <td class="timestamp">${timestamp}</td>
                    <td class="username">${log.username}</td>
                    <td>${status}</td>
                    <td class="ip-address">${log.ip_address}</td>
```

```
                    <td class="location">${log.location ||
'Desconocida'}</td>
                    <td class="user-agent"
title="${log.user_agent}">${userAgent}</td>
                `;

                tbody.appendChild(row);
            });
        }

        function updateStats() {
            stats.total = allLogs.length;
            stats.successful = allLogs.filter(log => log.success).length;
            stats.failed = allLogs.filter(log => !log.success).length;

            document.getElementById('totalLogs').textContent =
stats.total;
            document.getElementById('successfulLogins').textContent
= stats.successful;
            document.getElementById('failedLogins').textContent =
stats.failed;
        }

        function updateLastUpdateTime() {
            const now = new Date();
            const timeString = now.toLocaleTimeString('es-ES');
            document.getElementById('lastUpdate').textContent =
timeString;
        }

        function updateLoadMoreButton(hasMore) {
            const loadMoreSection =
document.getElementById('loadMoreSection');
            loadMoreSection.style.display = hasMore ? 'block' : 'none';
        }
```

```
            function showRefreshIndicator() {
                document.getElementById('refreshIndicator').style.display
= 'inline-flex';
            }

            function hideRefreshIndicator() {
                document.getElementById('refreshIndicator').style.display
= 'none';
            }

            function formatTimestamp(timestamp) {
                try {
                    const date = new Date(timestamp);
                    return date.toLocaleString('es-ES', {
                        year: 'numeric',
                        month: '2-digit',
                        day: '2-digit',
                        hour: '2-digit',
                        minute: '2-digit',
                        second: '2-digit'
                    });
                } catch (e) {
                    return timestamp;
                }
            }

            function truncateText(text, maxLength) {
                if (!text) return '';
                return text.length > maxLength ? text.substring(0,
maxLength) + '...' : text;
            }
        </script>
```

```
</body>
</html>
```

1.3.1.6. Búsqueda de documentos.

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Mis Documentos</title>
    <link rel="stylesheet" href="{{ url_for('static',
filename='css/dashboard.css') }}">
</head>
<body>
    <div class="left-panel">
        <div class="branding">
            <h1>CloudSecure ERP</h1>
            <p>Gestión empresarial segura</p>
            <nav>
                <ul class="menu">
                    <li><a href="/">Inicio</a></li>
                    <li><a href="/dashboard">Panel</a></li>
                    <li><a href="/mis-documentos">Mis
Documentos</a></li>
                    <li><a href="/logout">Cerrar sesión</a></li>
                </ul>
            </nav>
        </div>
    </div>

    <div class="right-panel">
        <header class="header">
            <h2>Documentos personales de {{ username }}</h2>
        </header>
```

```
            {% if archivos %}
            <section class="document-table">
               <table>
                  <thead>
                     <tr>
                        <th>Archivo</th>
                        <th>Categoría</th>
                        <th>Acción</th>
                     </tr>
                  </thead>
                  <tbody>
                     {% for archivo in archivos %}
                     <tr>
                        <td>{{ archivo.nombre }}</td>
                        <td>{{ archivo.categoria }}</td>
                        <td>
                           <a href="{{ url_for('descargar_documento',
tipo=archivo.categoria, nombre=archivo.nombre) }}"
                              class="btn-descargar">Descargar</a>
                        </td>
                     </tr>
                     {% endfor %}
                  </tbody>
               </table>
            </section>
            {% else %}
            <p>No se encontraron documentos asociados a tu
usuario.</p>
            {% endif %}
         </div>
      </body>
      </html>
```

1.4. HOJAS DE ESTILO CSS

1.4.1. Estilos de login (login.css)

```
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

body {
    font-family: 'Inter', -apple-system, BlinkMacSystemFont, sans-
serif;
    background: #f8fafc;
    color: #1f2937;
    min-height: 100vh;
    display: flex;
}

.left-panel {
    flex: 1;
    background: linear-gradient(135deg, #1e40af 0%, #3730a3
100%);
    display: flex;
    align-items: center;
    justify-content: center;
    position: relative;
    overflow: hidden;
}

.left-panel::before {
```

```
        content: '';
        position: absolute;
        top: 0;
        left: 0;
        right: 0;
        bottom: 0;
        background: url('data:image/svg+xml,<svg
xmlns="http://www.w3.org/2000/svg" viewBox="0 0 100
100"><defs><pattern id="grid" width="10" height="10"
patternUnits="userSpaceOnUse"><path d="M 10 0 L 0 0 0 10"
fill="none" stroke="rgba(255,255,255,0.05)" stroke-
width="1"/></pattern></defs><rect width="100" height="100"
fill="url(%23grid)"/></svg>');
    }

    .branding {
        text-align: center;
        z-index: 1;
        color: white;
    }

    .branding h1 {
        font-size: 42px;
        font-weight: 800;
        margin-bottom: 15px;
        letter-spacing: -0.02em;
    }

    .branding p {
        font-size: 18px;
        opacity: 0.9;
        margin-bottom: 30px;
        max-width: 400px;
    }
```

```
.features {
    list-style: none;
    text-align: left;
}

.features li {
    display: flex;
    align-items: center;
    margin-bottom: 12px;
    font-size: 16px;
    opacity: 0.9;
}

.features li::before {
    content: '✓';
    color: #10b981;
    font-weight: bold;
    margin-right: 12px;
    background: rgba(16, 185, 129, 0.2);
    width: 24px;
    height: 24px;
    border-radius: 50%;
    display: flex;
    align-items: center;
    justify-content: center;
    font-size: 12px;
}

.right-panel {
    flex: 1;
    display: flex;
```

```
    align-items: center;
    justify-content: center;
    padding: 40px;
}

.login-form {
    width: 100%;
    max-width: 400px;
}

.form-header {
    text-align: center;
    margin-bottom: 40px;
}

.form-header h2 {
    font-size: 28px;
    font-weight: 700;
    color: #111827;
    margin-bottom: 8px;
}

.form-header p {
    color: #6b7280;
    font-size: 15px;
}

.form-group {
    margin-bottom: 24px;
}

.form-group label {
    display: block;
```

```
    color: #374151;
    font-weight: 600;
    margin-bottom: 6px;
    font-size: 14px;
}

.form-control {
    width: 100%;
    padding: 14px 16px;
    border: 2px solid #d1d5db;
    border-radius: 8px;
    font-size: 16px;
    transition: all 0.2s ease;
    background: white;
    color: #1f2937;
}

.form-control:focus {
    outline: none;
    border-color: #3b82f6;
    box-shadow: 0 0 0 3px rgba(59, 130, 246, 0.1);
}

.form-control::placeholder {
    color: #9ca3af;
}

.btn-primary {
    width: 100%;
    padding: 14px;
    background: #3b82f6;
    color: white;
    border: none;
```

```
    border-radius: 8px;
    font-size: 16px;
    font-weight: 600;
    cursor: pointer;
    transition: all 0.2s ease;
    margin-top: 8px;
}

.btn-primary:hover {
    background: #2563eb;
    transform: translateY(-1px);
}

.btn-primary:active {
    transform: translateY(0);
}

.form-options {
    display: flex;
    justify-content: space-between;
    align-items: center;
    margin: 20px 0;
    font-size: 14px;
}

.checkbox-wrapper {
    display: flex;
    align-items: center;
    color: #6b7280;
}

.checkbox-wrapper input {
    margin-right: 8px;
```

```
    accent-color: #3b82f6;
}

.forgot-link {
    color: #3b82f6;
    text-decoration: none;
    font-weight: 500;
}

.forgot-link:hover {
    text-decoration: underline;
}

.divider {
    margin: 30px 0;
    text-align: center;
    position: relative;
}

.divider::before {
    content: '';
    position: absolute;
    top: 50%;
    left: 0;
    right: 0;
    height: 1px;
    background: #e5e7eb;
}

.divider span {
    background: #f8fafc;
    padding: 0 16px;
    color: #6b7280;
```

```
    font-size: 14px;
}

.alternative-login {
    display: grid;
    grid-template-columns: 1fr 1fr;
    gap: 12px;
}

.alt-btn {
    padding: 12px;
    border: 2px solid #e5e7eb;
    background: white;
    border-radius: 8px;
    font-size: 14px;
    font-weight: 500;
    cursor: pointer;
    transition: all 0.2s ease;
    display: flex;
    align-items: center;
    justify-content: center;
}

.alt-btn:hover {
    border-color: #d1d5db;
    background: #f9fafb;
}

@media (max-width: 768px) {
    body {
        flex-direction: column;
    }
```

```
    .left-panel {
        min-height: 200px;
    }

    .branding h1 {
        font-size: 32px;
    }

    .features {
        display: none;
    }
}
```

1.4.2. Estilos del dashboard (dashboard.css)

```
* {
    box-sizing: border-box;
    margin: 0;
    padding: 0;
}

body {
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    background-color: #f9fafb;
    color: #333;
    display: flex;
    min-height: 100vh;
}

.left-panel {
    background-color: #2f3e4d;
```

```
    color: #f0f3f5;
    width: 280px;
    padding: 40px 25px;
    display: flex;
    flex-direction: column;
    justify-content: flex-start;
}

.branding h1 {
    font-weight: 700;
    font-size: 1.9rem;
    margin-bottom: 8px;
    letter-spacing: 1px;
}

.branding p {
    font-size: 0.95rem;
    margin-bottom: 35px;
    color: #abb2bc;
}

nav ul.menu {
    list-style: none;
}

nav ul.menu li {
    margin-bottom: 18px;
}

nav ul.menu li a {
    text-decoration: none;
    color: #a7b1bb;
    font-size: 1.05rem;
```

```
    transition: color 0.3s ease;
}

nav ul.menu li a:hover {
    color: #fff;
}

.right-panel {
    flex-grow: 1;
    background-color: #ffffff;
    padding: 40px 60px;
    box-shadow: -2px 0 10px rgba(0,0,0,0.05);
    display: flex;
    flex-direction: column;
}

.header {
    display: flex;
    justify-content: space-between;
    align-items: center;
    margin-bottom: 40px;
}

.header h2 {
    font-weight: 600;
    font-size: 1.7rem;
    color: #2c3e50;
}

.logout-form button.btn-logout {
    background-color: #e74c3c;
    border: none;
    padding: 10px 22px;
```

```
    color: white;
    font-weight: 600;
    font-size: 1rem;
    border-radius: 5px;
    cursor: pointer;
    transition: background-color 0.25s ease;
}

.logout-form button.btn-logout:hover {
    background-color: #c0392b;
}

.profile-info h3 {
    font-size: 1.3rem;
    margin-bottom: 20px;
    color: #34495e;
    font-weight: 600;
}

.profile-info table {
    width: 100%;
    border-collapse: collapse;
}

.profile-info th, .profile-info td {
    padding: 14px 12px;
    border-bottom: 1px solid #ecf0f1;
    text-align: left;
    font-size: 1rem;
    color: #34495e;
}

.profile-info th.sensitive {
```

```
    color: #c0392b;
}

.profile-info td.sensitive {
    color: #c0392b;
    font-family: monospace;
}

.profile-info td.monospace {
    font-family: monospace;
    color: #2c3e50;
}
.btn-descargar {
    background-color: #2980b9;
    color: white;
    padding: 8px 16px;
    text-decoration: none;
    border-radius: 4px;
    font-weight: 600;
}

.btn-descargar:hover {
    background-color: #21618c;
}
.actions {
    margin-top: 30px;
}

.btn-documents {
    background-color: #3498db;
    color: #fff;
    padding: 12px 20px;
    text-decoration: none;
```

```
    font-weight: 600;
    border-radius: 5px;
    transition: background-color 0.3s ease;
}

.btn-documents:hover {
    background-color: #2980b9;
}
.document-table table {
    width: 100%;
    border-collapse: collapse;
    margin-top: 20px;
}

.document-table th, .document-table td {
    padding: 14px;
    border-bottom: 1px solid #ecf0f1;
    text-align: left;
    font-size: 1rem;
}

.document-table th {
    background-color: #f0f3f5;
    color: #2c3e50;
    font-weight: bold;
}

.btn-descargar {
    background-color: #2ecc71;
    color: white;
    padding: 8px 14px;
    border: none;
    border-radius: 4px;
```

```
        text-decoration: none;
        font-weight: 600;
        transition: background-color 0.3s ease;
    }

    .btn-descargar:hover {
        background-color: #27ae60;
    }
```

1.5. TOKENIZADOR DE DATOS

```
import random
import string
from db.rds_connection import RDSConnection


def generar_token():
    """Genera un token aleatorio de 16 caracteres"""
    return ''.join(random.choices(string.ascii_letters + string.digits,
k=16))


def guardar_token(original_value):
    """Inserta en tokens_map si no existe y devuelve el token"""
    # Verificar que el valor original no sea None o vacío
    if not original_value:
        return None

    print(f"Guardando token para valor: {original_value[:10]}...")
```

```
    # Verificar si ya existe el token
    result = RDSConnection.execute_query(
        "SELECT token FROM tokens_map WHERE original_value = %s",
        (original_value,),
        fetch_one=True,
        fetch_results=True
    )

    if result:
        print(f"Token ya existe para este valor")
        return result[0]

    # Si no existe, crear uno nuevo
    nuevo_token = generar_token()
    print(f"Generando nuevo token: {nuevo_token}")

    # Verificar que el token sea único
    while True:
        existing_token = RDSConnection.execute_query(
            "SELECT original_value FROM tokens_map WHERE token =
%s",
            (nuevo_token,),
            fetch_one=True,
            fetch_results=True
        )
        if not existing_token:
            break
        print(f"Token duplicado, generando otro...")
        nuevo_token = generar_token()

    try:
        # Insertar el nuevo token
```

```
        print(f"Insertando token en base de datos...")
        RDSConnection.execute_query(
            "INSERT INTO tokens_map (original_value, token) VALUES
(%s, %s)",
            (original_value, nuevo_token),
            fetch_results=False
        )
        print(f"Token insertado correctamente")

        # Verificar que se insertó correctamente
        verificacion = RDSConnection.execute_query(
            "SELECT token FROM tokens_map WHERE original_value =
%s",
            (original_value,),
            fetch_one=True,
            fetch_results=True
        )

        if verificacion:
            print(f"Verificación exitosa: token guardado")
            return verificacion[0]
        else:
            print(f"Error: token no se guardó correctamente")
            return nuevo_token

    except Exception as e:
        print(f"Error al insertar token: {str(e)}")
        raise e

    return nuevo_token
```

```
def tokenizar_usuario(user_id, username, email, phone, ssn, api_key, password_hash):
    """Tokeniza los datos sensibles de un usuario incluyendo password_hash"""
    print(f"\nTokenizando usuario: {username} (ID: {user_id})")

    # Contar valores no nulos para debug
    campos_a_tokenizar = [
        ("email", email),
        ("phone", phone),
        ("ssn", ssn),
        ("api_key", api_key),
        ("password_hash", password_hash)
    ]

    campos_validos = [(nombre, valor) for nombre, valor in campos_a_tokenizar if valor]
    print(f"Campos a tokenizar: {len(campos_validos)} de 5")

    # Tokenizar cada campo sensible
    token_email = guardar_token(email) if email else None
    token_phone = guardar_token(phone) if phone else None
    token_ssn = guardar_token(ssn) if ssn else None
    token_api_key = guardar_token(api_key) if api_key else None
    token_password_hash = guardar_token(password_hash) if password_hash else None

    print(f"Tokens generados - Email: {'Si' if token_email else 'No'}, Phone: {'Si' if token_phone else 'No'}, SSN: {'Si' if token_ssn else 'No'}, API: {'Si' if token_api_key else 'No'}, Password Hash: {'Si' if token_password_hash else 'No'}")

    # Insertar o actualizar en users_tokenized
    try:
        RDSConnection.execute_query("""
```

```
            INSERT INTO users_tokenized (id, username, token_email,
token_phone, token_ssn, token_api_key, token_password_hash)
            VALUES (%s, %s, %s, %s, %s, %s, %s)
            ON CONFLICT (id) DO UPDATE SET
                username = EXCLUDED.username,
                token_email = EXCLUDED.token_email,
                token_phone = EXCLUDED.token_phone,
                token_ssn = EXCLUDED.token_ssn,
                token_api_key = EXCLUDED.token_api_key,
                token_password_hash = EXCLUDED.token_password_hash,
                tokenized_at = CURRENT_TIMESTAMP
        """, (user_id, username, token_email, token_phone, token_ssn,
token_api_key, token_password_hash),
        fetch_results=False)
        print(f"Usuario tokenizado insertado/actualizado correctamente")
    except Exception as e:
        print(f"Error al insertar usuario tokenizado: {str(e)}")
        raise e


def tokenizar_todos_los_usuarios():
    """Tokeniza todos los usuarios de la tabla original"""
    print("Iniciando tokenización de usuarios...")

    # Obtener todos los usuarios con todos los campos incluyendo
password_hash
    usuarios = RDSConnection.execute_query(
        "SELECT id, username, email, phone, ssn, api_key,
password_hash FROM users ORDER BY id",
        fetch_results=True
    )

    if not usuarios:
        print("No se encontraron usuarios para tokenizar")
```

```
        return

    print(f"Se encontraron {len(usuarios)} usuarios para tokenizar")

    # Tokenizar cada usuario
    for i, user in enumerate(usuarios, 1):
        user_id, username, email, phone, ssn, api_key, password_hash =
user
        try:
            print(f"\n{'-'*50}")
            print(f"[{i}/{len(usuarios)}] Procesando usuario:
{username}")
            print(f"   - Email: {'Si' if email else 'No'}")
            print(f"   - Phone: {'Si' if phone else 'No'}")
            print(f"   - SSN: {'Si' if ssn else 'No'}")
            print(f"   - API Key: {'Si' if api_key else 'No'}")
            print(f"   - Password Hash: {'Si' if password_hash else 'No'}")

            tokenizar_usuario(user_id, username, email, phone, ssn,
api_key, password_hash)
            print(f"[{i}/{len(usuarios)}] Usuario tokenizado: {username}")

        except Exception as e:
            print(f"Error tokenizando usuario {username}: {str(e)}")
            # Continuar con el siguiente usuario en lugar de fallar
completamente
            continue

    print("Tokenización completa con mapeo reversible.")



def verificar_tokenizacion():
    """Verifica que la tokenización se haya realizado correctamente"""
    print("\nVerificando tokenización...")
```

```
    # Contar registros en cada tabla
    usuarios_originales = RDSConnection.execute_query(
        "SELECT COUNT(*) FROM users",
        fetch_one=True,
        fetch_results=True
    )[0]

    usuarios_tokenizados = RDSConnection.execute_query(
        "SELECT COUNT(*) FROM users_tokenized",
        fetch_one=True,
        fetch_results=True
    )[0]

    tokens_mapeados = RDSConnection.execute_query(
        "SELECT COUNT(*) FROM tokens_map",
        fetch_one=True,
        fetch_results=True
    )[0]

    # Verificar distribución de tokens por tipo
    try:
        token_stats = RDSConnection.execute_query("""
            SELECT
                COUNT(CASE WHEN token_email IS NOT NULL THEN 1 END)
as emails_tokenizados,
                COUNT(CASE WHEN token_phone IS NOT NULL THEN 1
END) as phones_tokenizados,
                COUNT(CASE WHEN token_ssn IS NOT NULL THEN 1 END)
as ssns_tokenizados,
                COUNT(CASE WHEN token_api_key IS NOT NULL THEN 1
END) as api_keys_tokenizados,
                COUNT(CASE WHEN token_password_hash IS NOT NULL
THEN 1 END) as password_hashes_tokenizados
```

```
        FROM users_tokenized
    """, fetch_one=True, fetch_results=True)

    emails_tok, phones_tok, ssns_tok, apis_tok,
password_hashes_tok = token_stats

    print(f"Usuarios originales: {usuarios_originales}")
    print(f"Usuarios tokenizados: {usuarios_tokenizados}")
    print(f"Tokens únicos creados: {tokens_mapeados}")
    print(f"Distribución de tokens:")
    print(f"   - Emails tokenizados: {emails_tok}")
    print(f"   - Teléfonos tokenizados: {phones_tok}")
    print(f"   - SSNs tokenizados: {ssns_tok}")
    print(f"   - API Keys tokenizados: {apis_tok}")
    print(f"   - Password Hashes tokenizados:
{password_hashes_tok}")

    total_tokens_esperados = emails_tok + phones_tok + ssns_tok +
apis_tok + password_hashes_tok
    print(f"Total tokens esperados: {total_tokens_esperados}")

    if usuarios_originales == usuarios_tokenizados:
        print("Número de usuarios coincide")
    else:
        print("Hay discrepancias en el número de usuarios")

    if tokens_mapeados == total_tokens_esperados:
        print("Número de tokens coincide perfectamente")
    else:
        print(f"Discrepancia en tokens: esperados
{total_tokens_esperados}, encontrados {tokens_mapeados}")

    # Mostrar algunos ejemplos de tokens (sin mostrar passwords)
    ejemplos = RDSConnection.execute_query(
```

```
            "SELECT original_value, token FROM tokens_map WHERE
original_value NOT LIKE '%@%' LIMIT 5",
            fetch_results=True
        )

        if ejemplos:
            print(f"\nEjemplos de tokens creados (excluyendo emails por
privacidad):")
            for original, token in ejemplos:
                print(f"   {original[:15]}... -> {token}")

    except Exception as e:
        print(f"Error en verificación detallada: {str(e)}")


def agregar_columna_token_password_hash():
    """Agrega la columna token_password_hash a la tabla
users_tokenized si no existe"""
    try:
        print("Verificando si existe la columna token_password_hash...")
        RDSConnection.execute_query("""
            ALTER TABLE users_tokenized
            ADD COLUMN IF NOT EXISTS token_password_hash character
varying(16)
        """, fetch_results=False)
        print("Columna token_password_hash agregada o ya existía")
    except Exception as e:
        print(f"Error al agregar columna token_password_hash: {str(e)}")
        raise e
```

```
if __name__ == "__main__":
    try:

        print("-" * 70)

        # Agregar columna si no existe
        agregar_columna_token_password_hash()

        tokenizar_todos_los_usuarios()
        verificar_tokenizacion()

    except Exception as e:
        print(f"Error general: {str(e)}")
        import traceback
        traceback.print_exc()
```

Relación de documentos

(_) Memoria NN páginas
(X) Anexos 83 páginas

La Almunia, a 01 de 07 de 2025

Firmado: Jesús Gabriel García Salomón