

**Nombre:**

Joao Alberto Navarro Sanchez

**Matrícula:**

22-1116

**Materia:**

Seguridad de sofware

**Docente:**

Joerlyn Morfe

**Tema:**

Top ten OWASP

**Tabla de contenido**

[**Documentación de la Aplicación de Gestión de Usuarios con Flask** 3](#_Toc163834488)

[**Introducción** 3](#_Toc163834489)

[**Razones para Utilizar las Bibliotecas Seleccionadas** 3](#_Toc163834490)

[**Flask** 3](#_Toc163834491)

[**Flask-SQLAlchemy** 4](#_Toc163834492)

[**Flask-WTF y WTForms** 4](#_Toc163834493)

[**Flask-Login** 5](#_Toc163834494)

[**Werkzeug y Jinja2** 5](#_Toc163834495)

[**Funcionalidades Principales** 5](#_Toc163834496)

[**Requisitos de Instalación** 7](#_Toc163834497)

[**Ejecución de la Aplicación** 7](#_Toc163834498)

[**Consideraciones de Seguridad** 8](#_Toc163834499)

# 

# **Documentación de la Aplicación de Gestión de Usuarios con Flask**

# **Introducción**

Esta es una aplicación web desarrollada utilizando el framework Flask en Python. Su propósito principal es gestionar usuarios, permitiéndoles registrarse, iniciar sesión, y visualizar el estado de su cuenta. Además, proporciona funcionalidades de administración para los usuarios con roles de administrador.

# **Razones para Utilizar las Bibliotecas Seleccionadas**

## **Flask**

Flask es un microframework web ligero y flexible para Python. Se eligió Flask debido a las siguientes razones:

Minimalismo: Flask es minimalista, lo que significa que proporciona solo las características esenciales para crear aplicaciones web. Esto lo hace ideal para proyectos pequeños y medianos, donde la simplicidad y la flexibilidad son importantes.

Extensibilidad: Aunque Flask es minimalista, es altamente extensible. Permite agregar funcionalidades adicionales a través de extensiones o paquetes de terceros según sea necesario, lo que brinda una gran flexibilidad para adaptarse a los requisitos del proyecto.

## **Flask-SQLAlchemy**

Flask-SQLAlchemy es una extensión de Flask que proporciona integración con SQLAlchemy, un popular ORM (Mapeo Objeto-Relacional) para Python. Las razones para su elección incluyen:

Facilidad de Uso: Flask-SQLAlchemy simplifica la configuración y el uso de SQLAlchemy en aplicaciones Flask. Proporciona una interfaz más amigable y simplificada para interactuar con bases de datos relacionales.

Abstracción de Base de Datos: SQLAlchemy ofrece una capa de abstracción sobre la base de datos subyacente, lo que permite trabajar con objetos Python en lugar de consultas SQL directas. Esto hace que el desarrollo y el mantenimiento del código sean más sencillos y menos propensos a errores.

## **Flask-WTF y WTForms**

Flask-WTF es una extensión de Flask para la validación de formularios web que se basa en WTForms, una biblioteca de validación de formularios flexible y fácil de usar. Las razones para su uso son las siguientes:

Validación de Formularios: Flask-WTF y WTForms simplifican la validación y el procesamiento de formularios web en aplicaciones Flask. Proporcionan una forma sencilla y robusta de definir campos de formulario y aplicar reglas de validación personalizadas.

Protección contra CSRF: Flask-WTF incluye protección contra ataques CSRF (Cross-Site Request Forgery) de forma predeterminada, lo que mejora la seguridad de la aplicación al prevenir solicitudes maliciosas desde otros sitios.

## **Flask-Login**

Flask-Login es una extensión de Flask que gestiona la autenticación de usuarios y las sesiones de usuario. Se eligió Flask-Login por las siguientes razones:

Gestión de Sesiones: Flask-Login simplifica la gestión de sesiones de usuario, lo que facilita la autenticación de usuarios y el control de acceso en la aplicación.

Integración con Flask: Flask-Login se integra perfectamente con Flask y proporciona una forma fácil de agregar autenticación de usuario a una aplicación Flask existente.

## **Werkzeug y Jinja2**

Werkzeug y Jinja2 son bibliotecas fundamentales que se utilizan internamente en Flask para manejar la solicitud y respuesta HTTP, así como para la generación de plantillas HTML. Se seleccionaron por las siguientes razones:

Parte Integral de Flask: Werkzeug y Jinja2 son componentes esenciales de Flask, y se utilizan internamente en el framework para manejar solicitudes HTTP y renderizar plantillas HTML. Al utilizar Flask, estas bibliotecas se incluyen automáticamente.

Robustez y Eficiencia: Werkzeug proporciona herramientas sólidas y eficientes para manejar solicitudes y respuestas HTTP, mientras que Jinja2 ofrece un potente sistema de plantillas para generar contenido HTML de manera dinámica.

# **Funcionalidades Principales**

Registro de Usuarios: Los usuarios pueden registrarse en la plataforma proporcionando un nombre de usuario único y una contraseña segura.

Inicio de Sesión: Los usuarios registrados pueden iniciar sesión en la aplicación utilizando sus credenciales.

Gestión de Sesiones: Se utiliza el paquete Flask-Login para gestionar las sesiones de usuario de forma segura.

Roles de Usuario: Los usuarios pueden tener roles de usuario estándar o de administrador, con diferentes niveles de acceso y privilegios.

Bloqueo de Usuarios: Los administradores pueden bloquear y desbloquear usuarios, así como gestionar usuarios bloqueados debido a intentos de inicio de sesión fallidos.

Protección contra Inyección SQL: La aplicación implementa medidas de seguridad para prevenir la inyección SQL.

Protección contra Cross-Site Scripting (XSS): Se han implementado técnicas para evitar ataques de XSS en la aplicación.

Protección contra Exposición de Datos Sensibles: Los datos sensibles, como las contraseñas de los usuarios, se almacenan de forma segura utilizando funciones de hash.

Control de Acceso Robusto: Se implementa un control de acceso adecuado para proteger las rutas y funcionalidades sensibles de la aplicación.

Estructura del Proyecto

El proyecto está organizado de la siguiente manera:

app.py: Este archivo contiene el código principal de la aplicación Flask, incluyendo las rutas y la lógica de negocio.

models.py: Aquí se definen los modelos de datos utilizando SQLAlchemy, incluyendo el modelo de Usuario y los intentos de inicio de sesión fallidos.

forms.py: Define los formularios web utilizando Flask-WTF para el registro y el inicio de sesión de los usuarios.

templates/: Contiene los archivos HTML para las diferentes páginas de la aplicación.

static/: Directorio para archivos estáticos como CSS, JavaScript, e imágenes.

# **Requisitos de Instalación**

Para ejecutar la aplicación localmente, necesitarás tener instalado Python y algunas dependencias específicas de Python. Puedes instalar estas dependencias utilizando pip y el archivo requirements.txt proporcionado.

# **Ejecución de la Aplicación**

Para ejecutar la aplicación, simplemente ejecuta el archivo app.py desde tu terminal:

Copy code

python app.py

Esto iniciará el servidor de desarrollo de Flask y podrás acceder a la aplicación desde tu navegador web en la dirección http://localhost:5000.

# **Consideraciones de Seguridad**

Cookies Seguras: Se ha configurado la aplicación para utilizar cookies seguras para la gestión de sesiones.

CSRF Protection: Se utiliza CSRF Protection para prevenir ataques de falsificación de solicitudes entre sitios.

Hashing de Contraseñas: Las contraseñas de los usuarios se almacenan utilizando funciones de hash seguras.

Bloqueo de Usuarios: Se implementa un mecanismo para bloquear temporalmente a los usuarios después de un número específico de intentos de inicio de sesión fallidos.