

**Universidad Espíritu Santo**

**Modalidad en Línea**

Ingeniería en Ciencias de la Computación

**PLATAFORMA DE TRUEQUE DE LIBROS ENTRE ESTUDIANTES**

Estudiante : Alex Mendoza Morante

Oscar Vallejo Mino

Bryan Cuenca Guerrero

Materia : Diseño de Software

Docente : MTI Vanessa Jurado

Fecha de Entrega : 21 de abril del 2025

Contenido

[**Gestión de Seguridad en la Arquitectura** 2](#_Toc195624724)

[**Autenticación y control de acceso** 2](#_Toc195624725)

[**Roles de usuario** 2](#_Toc195624726)

[**Validación de formularios** 2](#_Toc195624727)

[**Protección contra inyecciones** 2](#_Toc195624728)

[**Manejo seguro de contraseñas** 2](#_Toc195624729)

[**Buenas prácticas en despliegue (proyectado a futuro)** 2](#_Toc195624730)

[**CONCLUSIÓN** 3](#_Toc195624731)

# **Gestión de Seguridad en la Arquitectura**

La seguridad en la plataforma será abordada desde múltiples niveles, siguiendo buenas prácticas de desarrollo seguro y aprovechando las funcionalidades integradas que ofrece el framework Django.

## **Autenticación y control de acceso**

* Se implementará el sistema de **autenticación por sesión** que Django proporciona por defecto, permitiendo el ingreso de usuarios registrados y restringiendo funcionalidades sensibles como publicar libros o contactar usuarios.
* Los formularios estarán protegidos contra ataques de tipo **CSRF (Cross-Site Request Forgery)** gracias al middleware automático de Django.

## **Roles de usuario**

* En esta versión inicial se contemplará un único tipo de usuario: **estudiante registrado**.
* En futuras versiones, se podrá implementar un rol **moderador o administrador** para la validación de publicaciones o gestión del contenido.

## **Validación de formularios**

* Todos los formularios del sistema incluirán validación del lado del servidor para evitar:
  + Campos vacíos obligatorios
  + Inyecciones de datos
  + Ingreso de información no estructurada

## **Protección contra inyecciones**

* Django utiliza **ORM (Object-Relational Mapping)**, lo cual evita el uso directo de consultas SQL y, por tanto, **reduce significativamente el riesgo de inyecciones SQL**.
* Se desactivará el modo DEBUG en ambientes de producción para no mostrar errores con detalles sensibles.

## **Manejo seguro de contraseñas**

* Las contraseñas se almacenarán de forma segura usando **hashing con sal**, según los algoritmos recomendados por Django (PBKDF2 con SHA256 por defecto).
* En ningún caso se guardarán contraseñas en texto plano.

## **Buenas prácticas en despliegue (proyectado a futuro)**

* Uso de HTTPS (no aplicable en localhost, pero considerado para despliegue real).
* Protección contra clickjacking y otras vulnerabilidades web básicas mediante los encabezados de seguridad que Django ya incorpora.

# **CONCLUSIÓN**

La seguridad será gestionada de manera integral desde el diseño, el desarrollo y el despliegue, aprovechando las herramientas nativas de Django y siguiendo estándares básicos de ciberseguridad. Aunque se trata de un proyecto académico, se busca construir una base sólida para una posible evolución futura hacia un sistema más robusto y público.