

**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 : 18 de mayo del 2025

Contenido

[**PROPÓSITO DEL DOCUMENTO** 1](#_Toc196669518)

[**ESTRUCTURA GENERAL DE COMPONENTES** 1](#_Toc196669519)

[**COMUNICACIÓN ENTRE NODOS** 2](#_Toc196669520)

[**FLUJO GENERAL DE LA PLATAFORMA** 2](#_Toc196669521)

[**REQUERIMIENTOS DE INFRAESTRUCTURA** 2](#_Toc196669522)

[**CONSIDERACIONES** 3](#_Toc196669523)

# **PROPÓSITO DEL DOCUMENTO**

El Diagrama de Despliegue de la plataforma de trueque de libros entre estudiantes tiene como objetivo describir cómo se distribuyen físicamente los componentes del sistema, qué dispositivos o servidores participan y cómo interactúan entre sí a nivel de infraestructura.

Este diagrama asegura que el sistema sea diseñado considerando su correcta operación, comunicación y escalabilidad.

**ELEMENTOS DEL DIAGRAMA**

**Imagen que contiene Tabla

El contenido generado por IA puede ser incorrecto.**

|  |  |
| --- | --- |
| Nodo | Descripción |
| Cliente Web | Dispositivo del usuario (PC, laptop o móvil) con navegador web que accede a la plataforma. |
| Servidor de Aplicaciones | Servidor donde se despliega el framework Django que procesa la lógica de negocio y maneja la interacción con los datos. |
| Servidor de Base de Datos | Servidor donde reside el motor de base de datos MySQL, que almacena toda la información del sistema (usuarios, libros, mensajes, catálogos). |

# **ESTRUCTURA GENERAL DE COMPONENTES**

|  |  |  |
| --- | --- | --- |
| Artefacto | Nodo asociado | Descripción |
| Código Django | Servidor de Aplicaciones | Lógica del backend, apps Django (usuarios, libros, catalogos, contacto, perfiles). |
| Base de Datos trueque\_libros | Servidor de Base de Datos | Estructura de tablas y relaciones donde se guarda la información persistente. |

# **COMUNICACIÓN ENTRE NODOS**

|  |  |  |  |
| --- | --- | --- | --- |
| De | Hacia | Protocolo | Descripción |
| Cliente Web | Servidor de Aplicaciones | HTTP / HTTPS | El navegador envía solicitudes y recibe respuestas renderizadas (páginas HTML). |
| Servidor de Aplicaciones | Servidor de Base de Datos | MySQL Protocol (TCP/IP) | Django se conecta al motor MySQL para realizar operaciones de lectura y escritura. |

# **FLUJO GENERAL DE LA PLATAFORMA**

# **REQUERIMIENTOS DE INFRAESTRUCTURA**

|  |  |
| --- | --- |
| Componente | Requerimiento mínimo sugerido |
| Cliente Web | Navegador actualizado (Chrome, Firefox, Edge) |
| Servidor de Aplicaciones | CPU: 2 núcleos / RAM: 4 GB / OS: Linux o Windows Server / Python 3.10+ / Django 4+ |
| Servidor de Base de Datos | CPU: 2 núcleos / RAM: 4 GB / MySQL 8+ instalado y configurado |
| Almacenamiento | 20 GB libres para base de datos y registros de la aplicación |
| Red | Acceso a Internet con HTTPS habilitado para producción |

# **PLAN DE DESPLIEGUE**

**Fase 1: Desarrollo local**

* Configuración de un entorno virtual (venv) en la máquina de desarrollo.
* Instalación de dependencias utilizando pip a partir del archivo requirements.txt.
* Ejecución del servidor de aplicaciones Django en modo desarrollo mediante el comando python manage.py runserver.
* Uso de una base de datos local (MySQL en localhost) para pruebas funcionales iniciales.

**Fase 2: Ambiente de pruebas**

* Instalación de Django y MySQL en un servidor de staging o prueba (preferentemente con sistema operativo Linux).
* Configuración del servidor para permitir el acceso interno del equipo de desarrollo.
* Despliegue de la aplicación en el servidor utilizando configuraciones básicas de producción (python manage.py collectstatic, python manage.py migrate).
* Realización de pruebas de integración de módulos principales (usuarios, libros, catálogos, contacto).
* Verificación de la conectividad segura con la base de datos MySQL en entorno de pruebas.

**Fase 3: Producción**

* Configuración de un servidor de producción con **NGINX** como proxy inverso y **Gunicorn** como servidor de aplicaciones WSGI para Django (alternativamente Apache + mod\_wsgi).
* Habilitación de comunicaciones seguras mediante la implementación de certificados SSL (Let's Encrypt o similar) para uso de HTTPS.
* Configuración de la base de datos MySQL en un servidor dedicado o instancia protegida dentro de una red privada o VPN.
* Ajuste de configuraciones críticas en Django:
  + Desactivar modo de depuración (DEBUG = False).
  + Configurar las variables de entorno y gestionar de forma segura los secretos de la aplicación (SECRET\_KEY, credenciales de base de datos, etc.).
  + Definición de políticas de seguridad de cabeceras HTTP (Content Security Policy, X-Frame-Options, etc.).
* Despliegue definitivo de la aplicación para acceso de usuarios finales.

# **CONSIDERACIONES**

* Todas las comunicaciones entre el cliente y el servidor deberían ser cifradas (HTTPS) en entornos de producción.
* El servidor Django debe gestionar la conexión de forma segura hacia la base de datos.
* El motor de base de datos puede estar en el mismo servidor o en un servidor separado, dependiendo de las necesidades de escalabilidad futura.