

**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

[**PROPÓSITO DEL DOCUMENTO** 2](#_Toc195623524)

[**ESTILO ARQUITECTÓNICO ADOPTADO** 2](#_Toc195623525)

[**TECNOLOGÍAS UTILIZADAS** 3](#_Toc195623526)

[**COMPONENTES O MÓDULOS PRINCIPALES** 3](#_Toc195623527)

[**DIAGRAMA GENERAL (REPRESENTACIÓN TEXTUAL)** 3](#_Toc195623528)

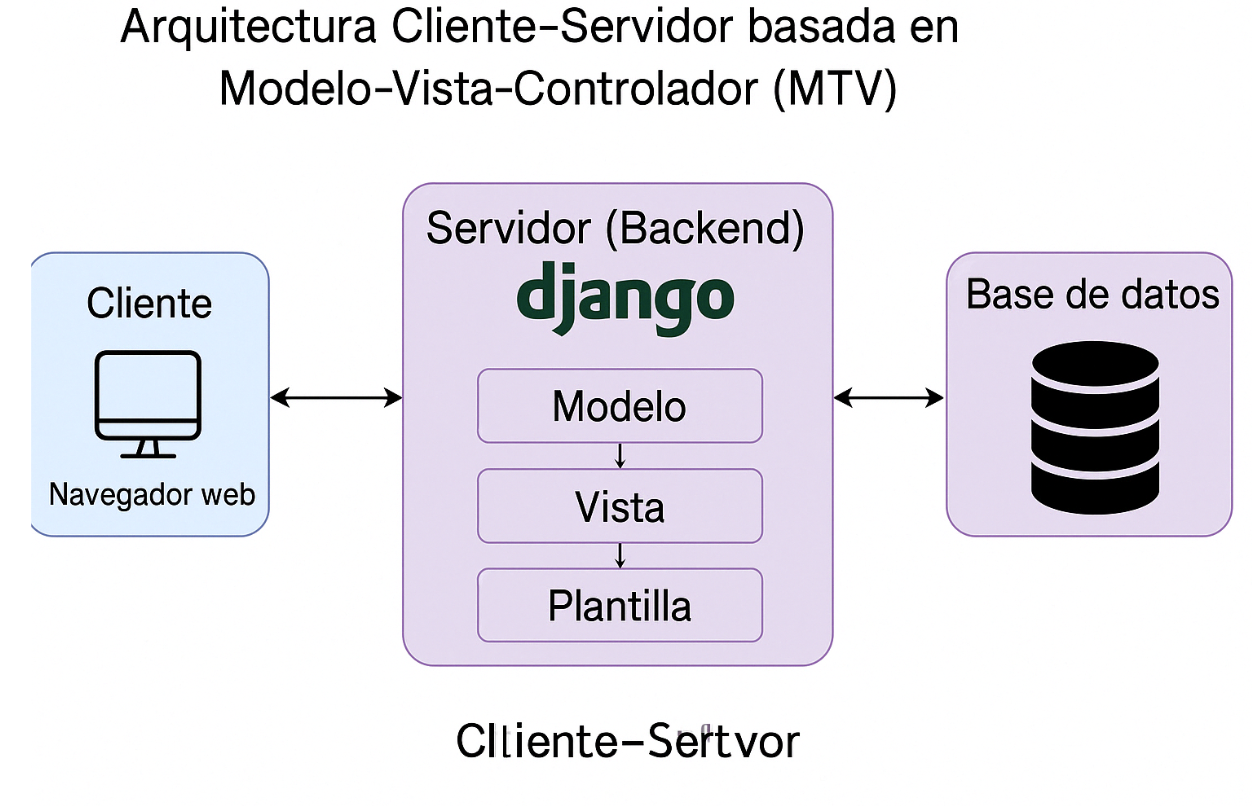
[**DIAGRAMA DE DESPLIEGUE (ENTORNO ACTUAL)** 3](#_Toc195623529)

[**JUSTIFICACIÓN DE DECISIONES TÉCNICAS** 4](#_Toc195623530)

# **PROPÓSITO DEL DOCUMENTO**

Este documento tiene como finalidad describir la arquitectura de software adoptada para el desarrollo de la plataforma de “**Trueque de libros entre estudiantes**”, proyecto correspondiente a la asignatura **Diseño de Software** de la Universidad Espíritu Santo (UEES). Aquí se presentan las decisiones técnicas clave, el estilo arquitectónico adoptado, los módulos que conforman el sistema, las tecnologías seleccionadas y los fundamentos detrás de cada elección.

# **ESTILO ARQUITECTÓNICO ADOPTADO**



La plataforma implementa una arquitectura Cliente-Servidor basada en el patrón Modelo-Vista-Controlador (MVC), adoptado a través del framework Django, que aplica la variante MTV (Model-Template-View). Esta elección permite una adecuada separación de responsabilidades, facilitando el mantenimiento y la escalabilidad del sistema.

El cliente (usuario final) accede a la aplicación desde un navegador web. El servidor (backend) procesa las solicitudes, interactúa con la base de datos y devuelve respuestas dinámicas a través de vistas HTML generadas desde plantillas.

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

# **TECNOLOGÍAS UTILIZADAS**

* Lenguaje de programación: Python 3.10+
* Framework web: Django
* Base de datos: MySQL
* Control de versiones: Git + GitHub
* Frontend: HTML5, CSS3, JavaScript (mediante plantillas de Django)
* IDE/Editor: Visual Studio Code
* Navegador: Chrome, Firefox, Edge, Safari (cualquier moderno)
* Despliegue local: Servidor de desarrollo de Django

# **COMPONENTES O MÓDULOS PRINCIPALES**

* Módulo de Usuarios: Manejo de autenticación, registro, inicio de sesión, cierre de sesión.
* Módulo de Libros: Registro, edición, eliminación, visualización y almacenamiento de publicaciones.
* Módulo de Búsqueda: Filtros dinámicos por título, autor o categoría.
* Módulo de Contacto: Permite a los estudiantes comunicarse para gestionar un intercambio.
* Módulo de Moderación (futuro): Validación de publicaciones antes de ser visibles.
* Módulo de Reportes (opcional): Estadísticas de uso, libros más demandados, categorías populares.

# **DIAGRAMA GENERAL (REPRESENTACIÓN TEXTUAL)**

Diagrama

El contenido generado por IA puede ser incorrecto.

[Usuario] ⇄ [Frontend (Template HTML)] ⇄ [Django View (Controlador)] ⇄ [Modelo] ⇄ [Base de datos MySQL

# **DIAGRAMA DE DESPLIEGUE (ENTORNO ACTUAL)**

El sistema se diseñará para ejecutarse en entorno local:

* Cliente: Navegador del usuario.
* Servidor: Framework Django ejecutado en entorno de desarrollo local.
* Base de datos: MySQL instalada localmente o en servidor de desarrollo.

# **JUSTIFICACIÓN DE DECISIONES TÉCNICAS**

* Django ofrece una arquitectura robusta y rápida de implementar, ideal para proyectos académicos con tiempos limitados.
* MySQL es una base de datos estable, bien documentada y ampliamente usada, ideal para sistemas relacionales.
* El patrón MTV permite separar claramente lógica de negocio, presentación e interacción con datos.
* GitHub facilita el trabajo colaborativo, control de versiones y documentación del proceso de desarrollo.

|  |  |  |  |
| --- | --- | --- | --- |
| **Criterio** | **Django (Python)** | **ASP.NET MVC (C#)** | **React + Node.js** |
| Arquitectura base | MVC (MTV en Django) | MVC tradicional | SPA (basado en componentes + API REST) |
| Lenguaje | Python | C# (compilado) | JavaScript / JSX |
| Facilidad de aprendizaje | Alta (muy legible y rápido de implementar) | Media/Alta (más robusto, pero más complejo) | Alta para frontend, media para backend |
| Integración rápida | Excelente para MVP o prototipos rápidos | Buena pero más pesada | Alta pero más distribuida (requiere configurar backend + frontend) |
| Estructura y convenciones | Muy clara y estructurada (baterías incluidas) | Muy organizada (convención sobre configuración) | Altamente libre, más propenso a desorden |
| Curva de mantenimiento | Manejable con pocos desarrolladores | Requiere más planificación y experiencia | Depende del stack y fragmentación |
| Comunidad académica | Muy usada en educación y prototipado | Más orientada a empresas y entornos corporativos | Popular entre desarrolladores frontend |

# **CURVA DE APRENDIZAJE Y TIEMPO DE DESARROLLO**

La elección de tecnologías para este proyecto no solo se basó en aspectos técnicos y arquitectónicos, sino también en factores humanos y académicos, como la **curva de aprendizaje** y el **tiempo estimado de desarrollo**.

## **Curva de aprendizaje**

Python y Django ofrecen una curva de aprendizaje considerablemente más suave comparada con otros entornos como ASP.NET (C#) o stacks divididos como React + Node.js. Su sintaxis clara, la documentación oficial extensa y una comunidad enfocada tanto en educación como en productividad, hacen que nuevos desarrolladores puedan incorporarse rápidamente al equipo y aportar de forma efectiva.

En comparación, tecnologías como C# con ASP.NET requieren dominar herramientas como Visual Studio, estructuras más rígidas y configuraciones adicionales para correr un entorno MVC completo. React + Node.js, aunque flexibles y modernos, dividen frontend y backend, lo que obliga a gestionar múltiples entornos, rutas y controladores desde el inicio, generando una carga de configuración inicial mayor.

## **Tiempo estimado de desarrollo**

Gracias a la estructura integrada de Django, que incluye herramientas como ORM, autenticación, manejo de formularios, plantillas y panel de administración, el equipo puede enfocarse en la lógica del negocio más que en la infraestructura técnica.

Esto representa una gran ventaja para proyectos universitarios, donde los recursos humanos y el tiempo son limitados. En nuestro caso, permitió avanzar con:

* Registro de usuarios
* Publicación de libros
* Búsqueda y visualización de publicaciones
* Flujo básico de contacto entre usuarios

Todo esto en un lapso estimado de 3 a 4 semanas de trabajo colaborativo con integración a GitHub.

# **CONCLUSIÓN**

Django fue seleccionado no solo por su potencia técnica, sino por su equilibrio entre facilidad de aprendizaje, velocidad de implementación y claridad estructural, lo que lo convierte en una herramienta ideal para equipos pequeños que buscan entregar resultados funcionales en poco tiempo y con buena calidad.