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

OBJETIVO

El presente documento tiene como objetivo justificar las decisiones de diseño adoptadas para el proyecto Plataforma de Trueque de Libros entre Estudiantes, considerando los patrones de diseño utilizados, la arquitectura del sistema, y la selección de tecnologías. Se toma en cuenta que se trata de un proyecto universitario con tiempo y recursos limitados, buscando un balance entre simplicidad, buenas prácticas y eficiencia.

ARQUITECTURA GENERAL

Se seleccionó una arquitectura Cliente-Servidor basada en el patrón Modelo-Vista-Controlador (MVC), implementado a través del framework Django (variante MTV: Model-Template-View).

Justificación:

- Separación de responsabilidades clara (modelo de datos, lógica de control, presentación).
- Facilidad de mantenimiento y extensión del sistema en el futuro.
- Simplicidad de implementación para equipos pequeños (ideal para proyectos universitarios).
- Compatibilidad nativa de Django con el patrón MTV.
- Soporte robusto para bases de datos y autenticación integrada.

PATRONES DE DISEÑO UTILIZADOS

Patrón	Descripción	Justificación de Uso
MVC / MTV	Separación entre lógica de presentación, lógica de negocio y datos.	Facilita el trabajo en equipo y el mantenimiento del sistema. Ideal para proyectos web rápidos.
Repository Pattern	(Aplicado en Django ORM implícitamente) Acceso estructurado a datos.	Permite cambiar la fuente de datos (base de datos) sin afectar la lógica de negocio.
Factory Pattern (Implícito en Django Forms/Models)	Creación de objetos de datos (instancias de modelos).	Estándar en Django para gestionar la creación y validación de formularios de entrada.
Singleton Pattern (para configuraciones de entorno)	Asegura una instancia única para configuraciones globales.	Implementado en Django settings.py para mantener un único punto de configuración.

CRUD Pattern	Diseño de operaciones básicas: Create, Read, Update, Delete.	Organiza la gestión de todas las entidades: Usuarios, Libros, Peticiones, etc.

SELECCIÓN DE COMPONENTES TECNOLÓGICOS

Componente	Selección	Justificación
Backend	Python + Django	Permite un desarrollo rápido, seguro y estructurado; ideal para proyectos universitarios de corta duración.
Base de Datos	MySQL	Motor de base de datos maduro, confiable, ampliamente soportado por Django y gratuito.
Frontend	Templates de Django (HTML5, CSS básico)	Suficiente para un prototipo funcional; permite integración rápida y responsiva.
Servidor de Producción	Gunicorn + Nginx (sugerido)	Solución estándar para despliegue seguro y escalable de aplicaciones Django.
Control de Versiones	GitHub (con Issues, Branches y Wiki)	Permite colaboración distribuida, control de cambios y evidencia de trabajo colaborativo.

MODELADO DE DATOS

Se diseñó un **modelo de datos relacional** que cumple con:

- Normalización adecuada de tablas.
- Integridad referencial usando claves primarias y foráneas.
- Relaciones claras entre entidades (Usuario, Libro, Petición, Categoría, etc.).
- Facilidad para consultas y reportes futuros.

El diseño fue pensado para mantener la **coherencia de los datos** y **reducir redundancia**.

CONSIDERACIONES ESPECIALES

- El **modelo de clases** refleja claramente la estructura del sistema y las relaciones entre entidades.
- El **modelo de secuencia** describe los principales flujos de interacción usuario-sistema.
- La **documentación** se organiza de acuerdo a las mejores prácticas de ingeniería de software.
- Se priorizó la **simplicidad** en la solución para asegurar que el proyecto fuera **entregable a tiempo**.

CONCLUSIONES

La selección de los patrones de diseño y componentes tecnológicos se fundamenta en la necesidad de:

- **Acelerar el desarrollo** sin comprometer la calidad.
- **Permitir escalabilidad razonable** en el futuro.
- **Fomentar la colaboración efectiva** del equipo mediante buenas prácticas estructurales.
- **Cumplir con los objetivos académicos** y de calidad del proyecto.

El enfoque adoptado busca garantizar un **prototipo funcional completo**, bien documentado, y fácilmente extensible para futuras mejoras o despliegues.