Proyecto Final — Sistema de Biblioteca

Portada
• **Nombre:** [Tu Nombre]
• **Matrícula:** [Tu Matrícula]
• **Título del proyecto:** Biblioteca System
• **Fecha:** 10 de diciembre de 2025

Índice

---

1. Inicio
Breve introducción al proyecto: Sistema web para gestionar préstamos, reservas y catálogo de una biblioteca.

2. Estrategia de Trabajo (Planificación)

2.1 Nombre del proyecto
Biblioteca System

2.2 Tecnología para aplicar
• Lenguaje: Python 3.10+
• Framework: Flask
• ORM: SQLAlchemy (Flask-SQLAlchemy)
• Autenticación: Flask-Login
• Tests: pytest
• Base de datos: SQLite (desarrollo), opcional PostgreSQL en producción
• Control de versiones: Git (GitHub)

2.3 Objetivo del proyecto
Desarrollar un sistema web que permita a usuarios registrarse, consultar el catálogo, solicitar préstamos y devolver libros. El sistema también deberá permitir a administradores gestionar el inventario.

2.4 Alcance del proyecto
• Gestión de usuarios (registro, login)
• Catálogo de libros con disponibilidad
• Préstamos y devoluciones
• Reservas de libros
• Panel administrativo para agregar libros
• Reportes básicos (historial de préstamos)

2.5 Cronograma (tabla resumida)

Actividad	Duración	Responsable
Planificación y diseño	3 días	Equipo

| Implementación modelos y DB | 4 días | Backend
| Rutas y autenticación | 4 días | Backend
| Frontend básico (templates) | 3 días | Frontend
| Tests y QA | 4 días | QA
| Documentación y demo | 2 días | Equipo

2.6 Definición del primer Release
El Release 1 incluirá:
• Registro/login de usuarios
• Catálogo navegable
• Solicitar préstamo (si hay copias disponibles)
• Devolución por el usuario
• Panel admin para agregar libros

Requerimientos funcionales (ejemplos)
• RF1: El usuario debe crear una cuenta.
• RF2: Visualizar catálogo de libros.
• RF3: Solicitar préstamo de libro con disponibilidad.

Requerimientos no funcionales
• RNF1: Respuesta en menos de 2s para páginas comunes.
• RNF2: Autenticación segura (hash de contraseñas).

3. Metodología Scrum

3.1 Tareas a ejecutar (resumidas)
• Diseño de DB
• Implementación de modelos
• Implementación de rutas y vistas
• Tests unitarios e integración
• Documentación y entrega

3.2 Equipo de trabajo (sugerido)
• Product Owner: Coordina requisitos y prioridad.
• Scrum Master: Facilita ceremonias.
• Developers: Backend (2), Frontend (1).
• QA: 1.

3.3 Herramientas
• Gestión de tareas: GitHub Issues o Jira
• Repositorio: GitHub
• CI: GitHub Actions

3.4 Épicas
• Epic: Gestión de usuarios (registro, login, perfiles)
• Epic: Gestión de catálogo (CRUD libros)
• Epic: Préstamos y reservas

3.5 Ceremonias
• Sprint planning: Inicio de cada sprint (1 día antes)
• Daily stand-up: 15 min diarios
• Sprint review: Al final del sprint

3.6 Historias de usuario (10 ejemplos)
1. Como usuario no registrado quiero registrarme para acceder al sistema.

- Criterios de aceptación: formulario con username, email, password; usuario creado en DB; redirección a login. (3 puntos)
2. Como usuario quiero iniciar sesión para acceder a funciones de préstamo.
- CA: credenciales válidas inician sesión; fallo muestra mensaje. (2 pts)
3. Como usuario quiero ver el catálogo de libros para elegir un libro.
- CA: lista de libros con disponibilidad; búsqueda por título/autor. (3 pts)
4. Como usuario quiero solicitar préstamo de un libro disponible.
- CA: sólo si available_copies>0; crea registro de préstamo; decrementa disponible. (5 pts)
5. Como usuario quiero devolver un libro que presté.
- CA: marca préstamo como devuelto; incrementa disponible. (3 pts)
6. Como usuario quiero reservar un libro no disponible.
- CA: crea reserva activa; notifica al usuario cuando disponible. (5 pts)
7. Como admin quiero agregar nuevos libros al catálogo.
- CA: formulario admin; libro creado con copias. (3 pts)
8. Como admin quiero ver listado de préstamos activos.
- CA: mostrar lista con usuario, libro y fecha. (3 pts)
9. Como usuario quiero ver mi historial de préstamos.
- CA: listar préstamos pasados y actuales. (3 pts)
10. Como admin quiero eliminar un libro obsoleto.
- CA: operación protegida; libro eliminado si no tiene préstamos activos. (5 pts)

4. Plan de Pruebas

4.1 Requerimientos mapeados
• RF1-RF3 mapeados a historias 1-4.

4.2 Criterios de aceptación/rechazo
• Aceptación: el resultado esperado coincide con el criterio; pruebas automatizadas pasan.
• Rechazo: error, comportamiento inesperado o incumplimiento de performance.

4.3 Herramientas de pruebas
• `pytest` para unit/integration
• Selenium o Playwright para pruebas funcionales UI (opcional)
• GitHub Actions para ejecutar tests en CI

4.4 Cronograma de ejecución de pruebas
• Sprint 1: tests unitarios modelos y auth
• Sprint 2: tests integración rutas principales
• Sprint 3: pruebas funcionales y regresión

4.5 Plantilla para caso de prueba (ejemplo)
• ID: TC-01
• Historia: Registro de usuario
• Precondición: Base de datos vacía
• Pasos: 1) Abrir /register 2) Enviar formulario
• Resultado esperado: Usuario creado, mensaje de éxito
• Estado: Pass/Fail

4.6 Equipos y responsabilidades
• Developers: preparar y ejecutar tests unitarios
• QA: escribir casos funcionales y ejecutar Selenium

4.7 Plan de automatización
• Automatizar pruebas unitarias con `pytest`.
• Integrar tests en CI con GitHub Actions (workflow).

4.8 Evidencia de automatización
• En el repositorio se incluirán scripts de tests y resultados (logs). Para la entrega, incluir capturas de pantalla o vídeo de ejecución de tests en local/CI.

5. Demostración y Entregables
• Video demostrativo con las funcionalidades del primer Release.
• Repositorio con código (GitHub).
• Tablero de gestión (Jira/GitHub Projects) con historias y tareas.
• Código de pruebas automatizadas en `tests/`.

6. Conclusiones
Se entrega un MVP funcional que cubre las funciones principales y un plan para escalar y mejorar la calidad mediante pruebas automáticas y CI.

7. Bibliografía

• Documentación Flask: https://flask.palletsprojects.com/
• SQLAlchemy: https://docs.sqlalchemy.org/
• pytest: https://docs.pytest.org/