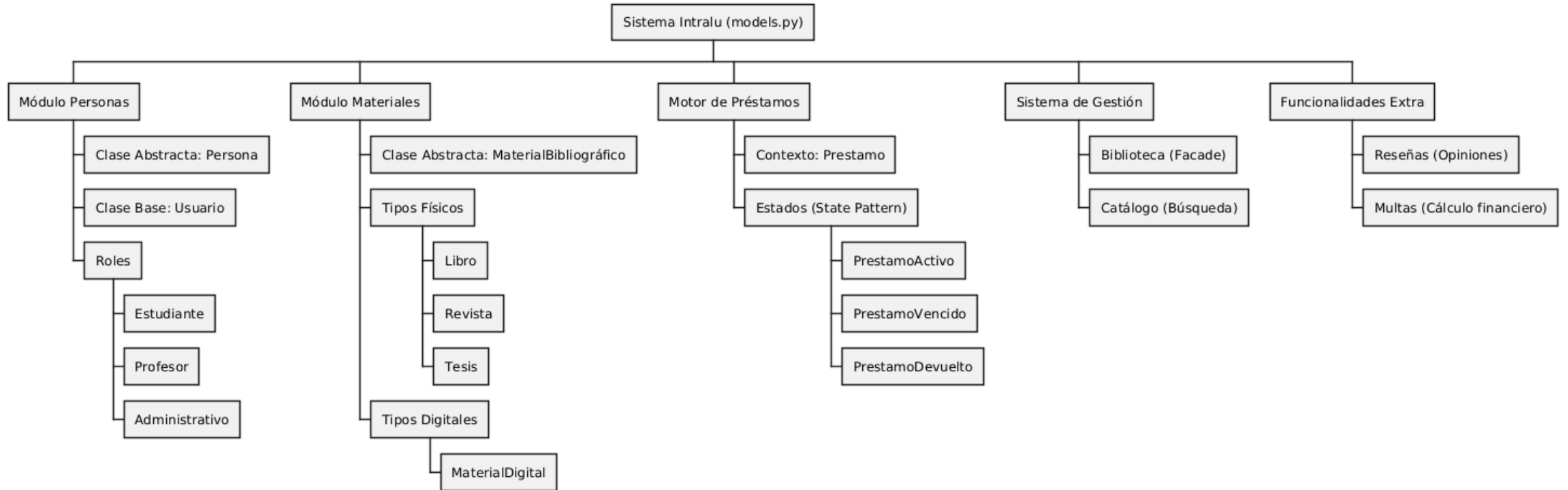


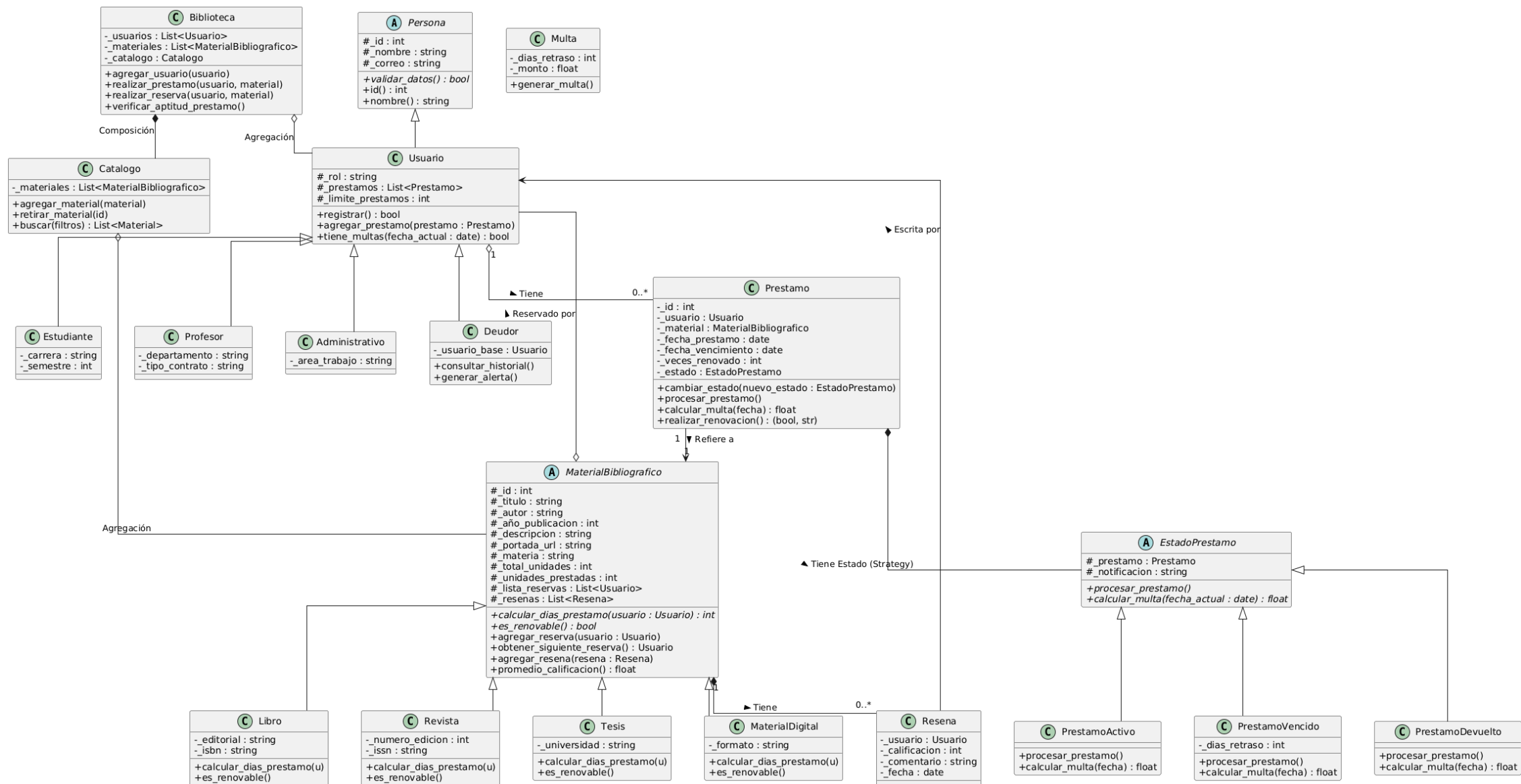
Gestor de Biblioteca Universitaria

Proyecto Final: Arquitectura de Software, Roles y Automatización

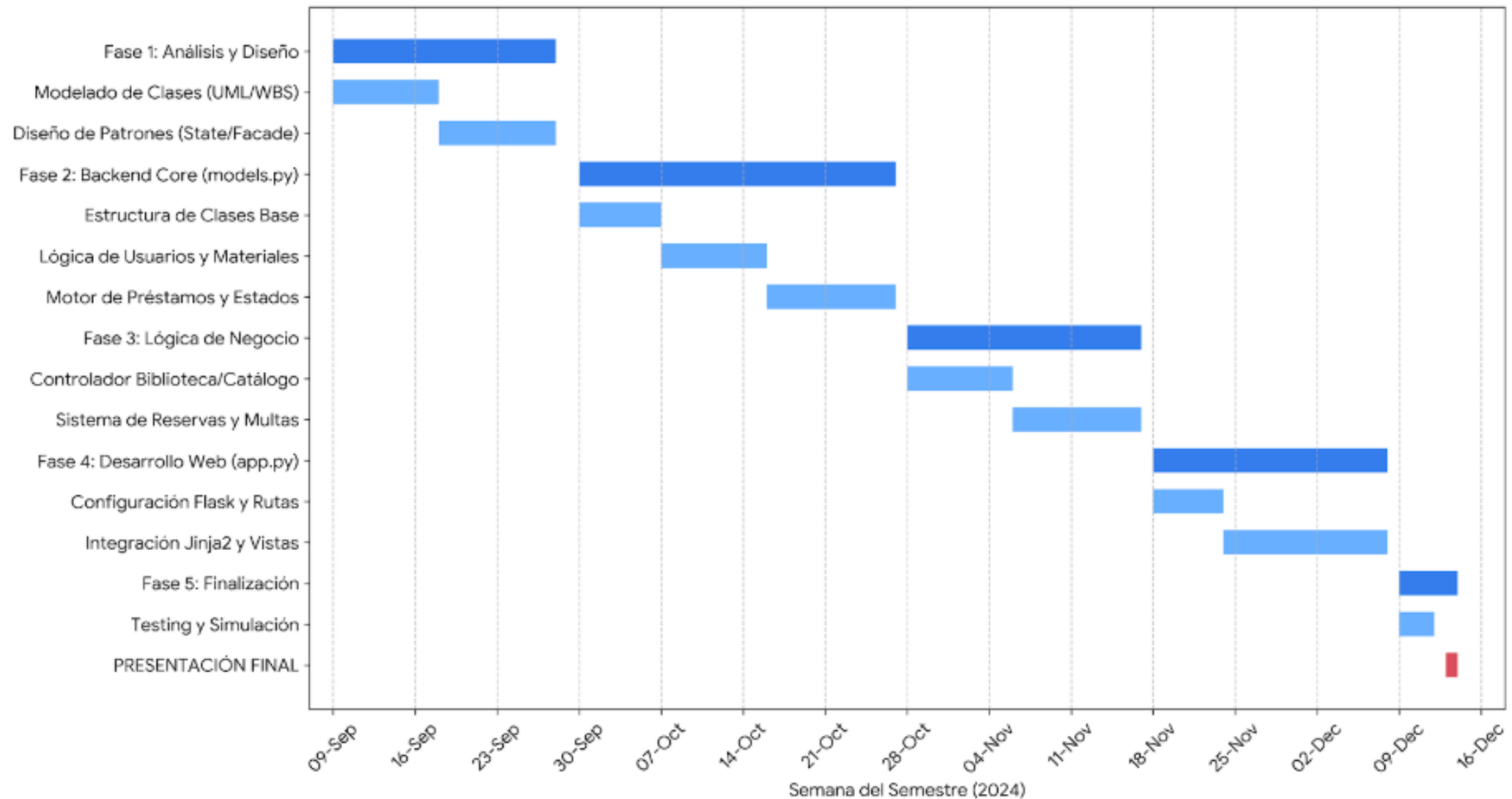
Estudiantes:

- Castillo Pacheco Arnold Fabrizzio
- Jimenez Altamirano Hervert Jeremy
- Barboza Mirez Piero Miguel

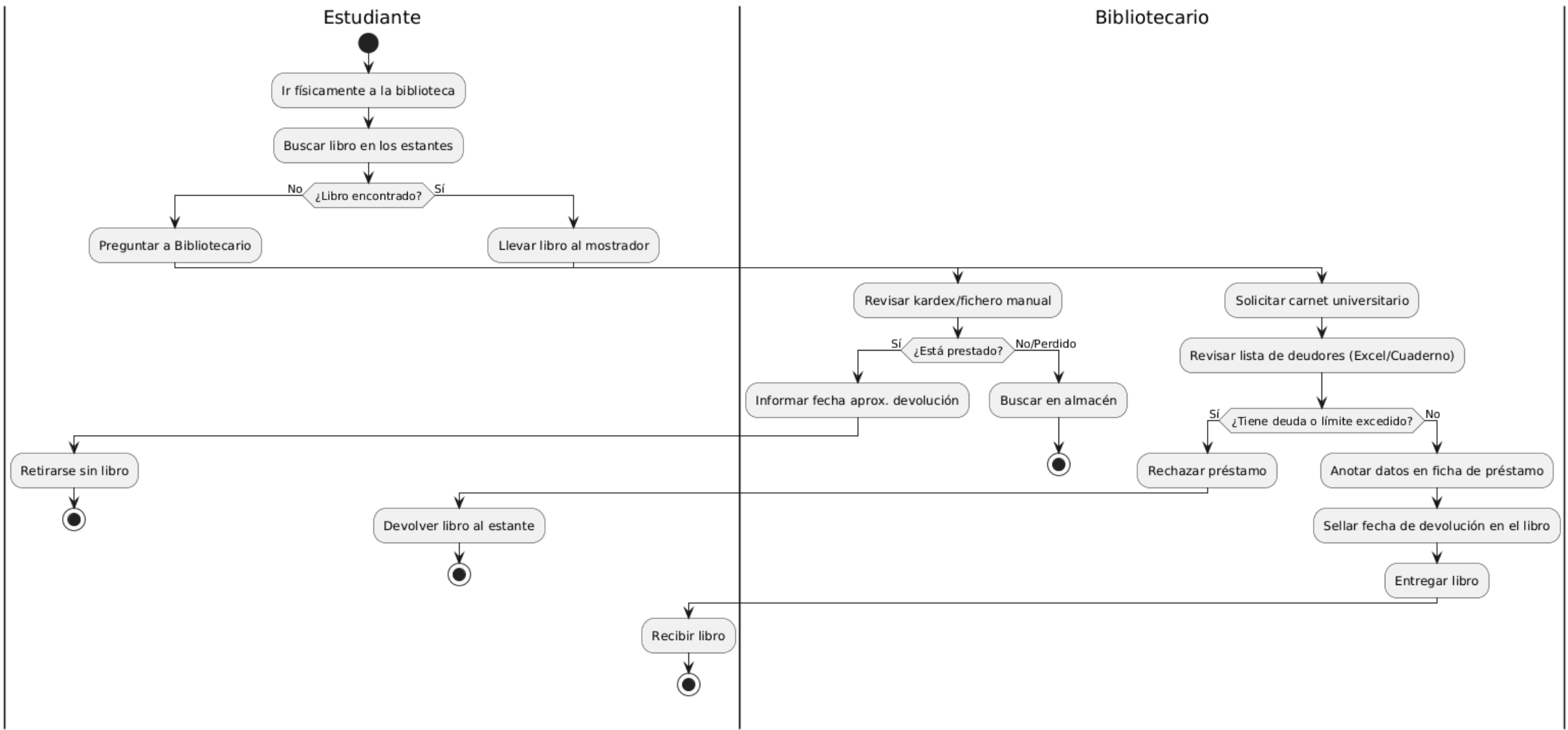




Cronograma Académico: Proyecto Intralu (Sep - Dic)



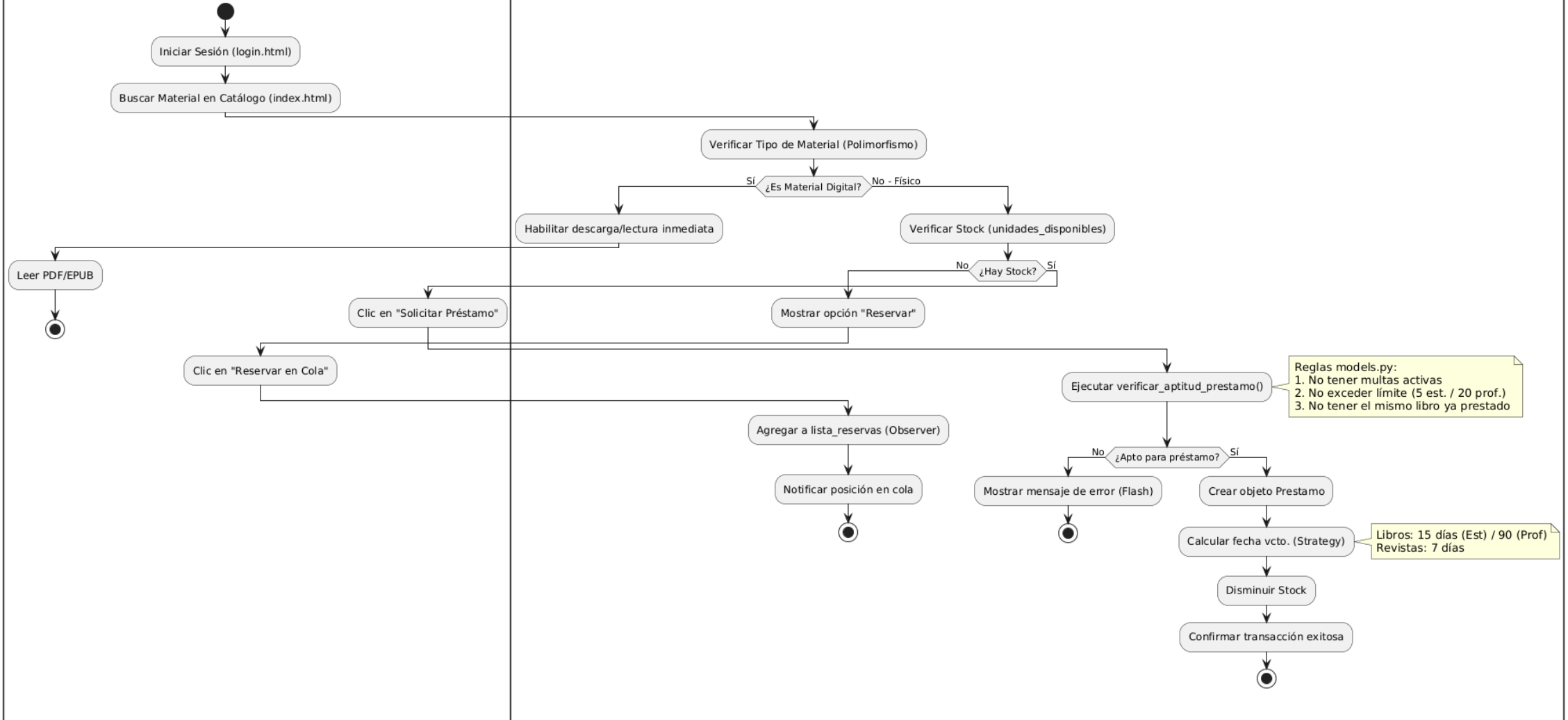
Proceso As-Is: Préstamo Manual de Libros



Proceso To-Be: Sistema Intralu (Automatizado)

Sistema (Intralu)

Usuario (Estudiante/Profesor)



Decisiones Arquitectónicas y Herramientas

- **¿Por qué Python como lenguaje base?**

- Versatilidad: Permite unir lógica de negocio robusta (POO) con librerías avanzadas de Ciencia de Datos (Scikit-learn, Matplotlib) en un solo entorno.

- **¿Por qué Flask como Framework Web?**

- Ligereza y Control: A diferencia de frameworks rígidos, Flask nos permite construir la arquitectura paso a paso, ideal para demostrar el dominio del código.
- Flexibilidad: Facilita la integración de modelos de Machine Learning personalizados sin restricciones.
- Jinja2: Motor de plantillas eficiente para renderizar datos dinámicos desde el servidor.

Diagnóstico Operativo: Puntos Críticos

- **1. Ineficiencia en Procesos Manuales:**

- Dependencia de registros físicos y hojas de cálculo desconectadas.
- Errores frecuentes en el cálculo de fechas de devolución.

- **2. Riesgo de Pérdida de Activos:**

- Falta de control automático para bloquear préstamos a usuarios morosos.

- **3. Ceguera de Datos:**

- Imposibilidad de analizar tendencias de lectura o identificar materiales escasos.

⚠ **Deuda total: S/. 25.00**

Selecciona método de pago:

BCP (Simulación) ▼

Pagar Todo

Propuesta de Valor del Gestor

- **Automatización Total:**
 - El sistema aplica reglas de negocio (multas, límites) sin intervención humana.
- **Gestión Híbrida:**
 - Capacidad de administrar inventario físico y proveer acceso a material digital (e-books) simultáneamente.
- **Centralización Web:**
 - Acceso remoto 24/7 para consultas, reservas y renovaciones.

```
class Usuario(Persona):
    def __init__(self, id: int, nombre: str, correo: str, rol: str):
        super().__init__(id, nombre, correo)
        self._rol = rol
        self._prestamos: List[Prestamo] = []
        self._limite_prestamos: int = 5
```

Modelo de Usuarios y Roles

- **Implementación mediante Herencia (POO):**
 - Clase Base 'Usuario': Contiene atributos comunes (Nombre, ID, Correo) y métodos base.
 - Subclases Especializadas: Estudiante, Profesor y Administrativo.
- **Polimorfismo en Reglas:**
 - Cada rol tiene privilegios distintos definidos en el código (Límites de préstamo y días de devolución).

Mis Préstamos y Deudas

✓ Estás al día.

Python para Principiantes

ACTIVO

Vence: 23-12-2025

Renovar

Perfil de Usuario: Estudiante

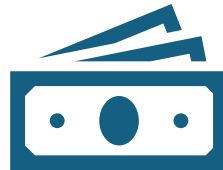
- **Características:**
 - Acceso estándar al catálogo y préstamos.
- **Reglas de Negocio (Hardcoded):**
 - Límite de Préstamos: Máximo 5 materiales simultáneos.
 - Duración de Préstamo (Libros): 15 días.
 - Duración de Préstamo (Tesis): 10 días.
- **Restricciones:**
 - Bloqueo automático si tiene multas pendientes > S/. 0.

Perfil de Usuario: Profesor



Privilegios Extendidos:

Diseñado para apoyar la investigación y docencia con plazos más holgados.



Reglas de Negocio Diferenciadas:

Límite de Préstamos: Máximo 20 materiales simultáneos.

Duración de Préstamo (Libros): 90 días (Polimorfismo).

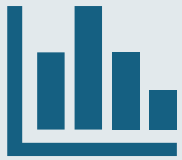
Duración de Préstamo (Tesis): 30 días.



Funcionalidad Exclusiva:

Prioridad implícita en la disponibilidad de material académico.

Perfil de Gestión: Administrador



Panel de Control (Dashboard):

Vista exclusiva no accesible para estudiantes ni profesores.



Funciones Críticas:

Gestión de Inventario: Agregar, editar o retirar libros del catálogo.

Monitoreo de Colas: Ver quién está esperando por un libro agotado.

Analítica: Visualizar gráficos de popularidad y estado de morosidad.



Permisos:

Límite de préstamos virtualmente ilimitado (para movimientos internos).

INTRALU

Catálogo

Préstamos

Admin

Simulación

Admin

Administrativo

MI Perfil

Salir

Panel de Administración

Gestión de Préstamos y Colas

Agregar Inventario

Análíticas

Monitor de Colas de Espera

Material	Total Espera	Cola
IA en Medicina	1	1. Admin Root

Préstamos Activos Globales

Usuario	Material	Vence	Estado	Multa
Ana García	Python para Principiantes	23-12-2025	OK	-
Ana García	Revista Científica	20-12-2025	OK	-
Dr. Pérez	IA en Medicina	02-01-2026	OK	-

Lo más popular

Python

Python para Principiantes

★ 4.5

Calculo

Cálculo Avanzado

★ 0.0

Ciencia

Revista Científica

★ 5.0

Tesis IA

IA en Medicina

★ 0.0

Catálogo Completo

Reseñas y Opiniones

Dejar una reseña

Calificación

★★★★★ (5)

Comentario

¿Qué te pareció este material?

Publicar Opinión

Dr. Pérez

Excelente libro introductorio.

Publicado el 13-12-2025

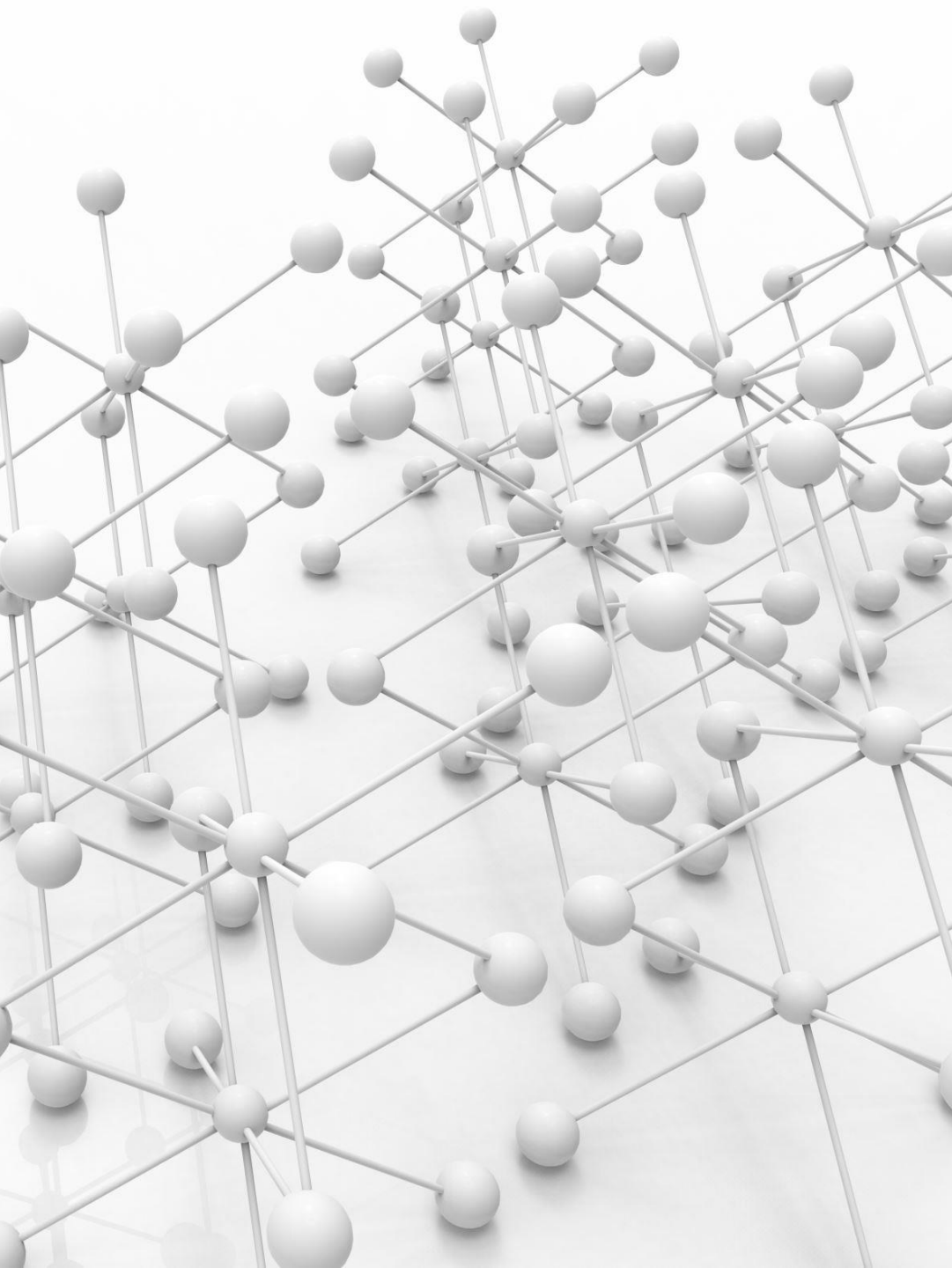
★★★★★

Ana García

Muy bueno.

Publicado el 13-12-2025

★★★★☆



Ciclo de Desarrollo

- **Fase 1: Abstracción (UML):**
 - Modelado de clases y relaciones antes de programar.
- **Fase 2: Backend (Python Core):**
 - Implementación de lógica pura (models.py) desconectada de la web.
- **Fase 3: Integración Web (Flask):**
 - Conexión de la lógica con vistas HTML/Jinja2.
- **Fase 4: Inteligencia Artificial:**
 - Agregado de valor con recomendaciones y análisis de datos.



Tecnologías Seleccionadas

- **¿Por qué Python + Flask?**
 - Control total de la arquitectura (Microframework).
 - Integración nativa con librerías de Ciencia de Datos (Scikit-Learn).
- **Frontend:**
 - HTML5, CSS3 (Diseño responsivo y Modo Oscuro).

Ingeniería: Patrón State

- **Problema:**
 - Gestionar el ciclo de vida complejo de un préstamo (Activo -> Vencido -> Devuelto).
- **Solución (State Pattern):**
 - Se delega el comportamiento a clases de estado específicas.
- **Beneficio:**
 - El objeto 'Prestamo' cambia su comportamiento (cálculo de multa) automáticamente cuando cambia su estado interno.

IA: Motor de Recomendación



Filtrado Basado en Contenido:

El sistema analiza la descripción y metadatos del libro que el usuario está viendo.



Objetivo:

Sugerir material semánticamente similar para fomentar la lectura e investigación.

Profundización en POO

Herencia para Reutilización:

- Clase Base 'MaterialBibliografico' -> Heredada por Libro, Revista, Tesis.

Polimorfismo en Acción:

- Método 'calcular_dias_prestamo()':
 - Cada tipo de material responde diferente al mismo mensaje.
 - - Libro: Reglas complejas según tipo de usuario (Profesor vs Alumno).
 - - Material Digital: Acceso corto y sin stock físico.

IA: Motor de Recomendaciones

¿Qué problema resuelve?

- Ayuda al estudiante a descubrir contenido relevante semánticamente.

Implementación Técnica (NLP):

- 1. Vectorización (TF-IDF): Convertimos el texto (título/descripción) en vectores numéricos.
- 2. Similitud del Coseno: Calculamos el ángulo entre vectores para encontrar materiales similares.
- Librería: Scikit-learn integrada directamente en el backend Flask.

Analítica y Visualización de Datos

Generación Dinámica de Gráficos:

- Uso de Matplotlib con backend 'Agg' para generar imágenes en memoria (sin guardar archivos).
- Conversión a Base64 para inyectar gráficos directamente en el HTML.

Valor para el Administrador:

- Monitoreo visual de la 'salud' de la biblioteca (Préstamos vencidos vs activos).
- Análisis de tendencias de lectura por tipo de material.

Conclusiones

Logros del Proyecto:

- Se logró automatizar un proceso manual complejo mediante una arquitectura de software escalable.
- Integración exitosa de tres mundos: Programación Orientada a Objetos, Desarrollo Web y Ciencia de Datos.

Siguientes Pasos (Roadmap):

- Persistencia en Base de Datos (SQL).
- Integración con pasarela de pagos real para multas.