Introducción

las bibliotecas físicas enfrentan el desafío de adaptarse a las necesidades de los usuarios y administradores, quienes requieren un acceso ágil, flexible y descentralizado a los recursos de información. Un sistema de **biblioteca digital** permite centralizar la gestión de usuarios, libros y préstamos, al tiempo que facilita el acceso remoto desde múltiples dispositivos conectados a internet.

El proyecto propuesto consiste en el desarrollo de una aplicación web basada en **Python y Flask**, orientada a la gestión de préstamos de libros en un entorno digital. El sistema busca ser accesible, seguro y escalable, brindando soporte tanto a lectores como a administradores, y contemplando las buenas prácticas en administración de datos, seguridad y disponibilidad.

Justificación

Las bibliotecas tradicionales presentan limitaciones en cuanto a la gestión de préstamos y el acceso a la información. El control manual mediante registros físicos o sistemas locales restringe la eficiencia de los procesos, genera riesgos de pérdida de datos y dificulta el acceso remoto a los recursos. Además, los usuarios deben acudir físicamente a la biblioteca para realizar trámites como consultar la disponibilidad de un libro o renovar un préstamo, lo que representa una barrera en términos de comodidad y acceso inclusivo.

En este contexto, surge la necesidad de un sistema de **biblioteca digital descentralizada**, que permita:

Optimizar la gestión administrativa: reduciendo errores humanos y mejorando la trazabilidad de préstamos y devoluciones.

Facilitar el acceso remoto: brindando a los usuarios la posibilidad de consultar catálogos, reservar y gestionar préstamos desde cualquier dispositivo conectado a internet.

Centralizar y proteger la información: mediante una base de datos estructurada que resguarde los registros históricos y facilite la recuperación de información.

Ampliar el alcance del servicio bibliotecario: superando las limitaciones de una red local y favoreciendo la inclusión digital.

La implementación de este sistema no solo resuelve problemáticas de administración y acceso, sino que también contribuye a modernizar la experiencia bibliotecaria, alineándola con los hábitos digitales actuales y con la demanda creciente de servicios en línea.

Objetivos del Proyecto

Objetivo general

Diseñar e implementar un sistema de biblioteca digital descentralizada, accesible desde diferentes dispositivos a través de internet, que permita la gestión eficiente de libros, usuarios y préstamos, garantizando la integridad y disponibilidad de los datos.

Objetivos específicos

1)- Acceso multiplataforma: permitir el uso del sistema desde computadoras, tablets y smartphones, sin necesidad de instalaciones adicionales.

2)- Gestión integral de datos: implementar un modelo de base de datos que administre usuarios, libros y préstamos de forma estructurada y segura.

3)- Descentralización de servicios: posibilitar el acceso remoto, superando las limitaciones de una red local, de manera que diferentes bibliotecas o sedes puedan compartir un mismo sistema.

4)- Control de préstamos y devoluciones: ofrecer un mecanismo claro y automatizado para registrar transacciones, evitando errores humanos y mejorando la trazabilidad.

5)- Seguridad y privacidad: garantizar la protección de la información mediante autenticación de usuarios, roles diferenciados (usuario, administrador) y buenas prácticas en el manejo de datos.

6)- Disponibilidad y accesibilidad: asegurar que la plataforma sea usable en todo momento, promoviendo la inclusión digital y el acceso a la información en distintos contextos.

Justificación de la elección de Python y Flask

La selección de tecnologías en un proyecto de software es un aspecto fundamental que impacta en su desarrollo, mantenimiento y escalabilidad futura. En este caso, la elección de **Python** como lenguaje de programación y **Flask** como framework web se fundamenta en las siguientes razones:

1)- Simplicidad y legibilidad de Python: Python se caracteriza por una sintaxis clara y expresiva, lo cual reduce la complejidad del código y facilita la colaboración entre desarrolladores, incluso aquellos con menor experiencia. Esto contribuye a un desarrollo más ágil y a una curva de aprendizaje accesible.

2)- Amplio ecosistema de librerías: Python ofrece una gran variedad de paquetes y herramientas que permiten extender fácilmente las funcionalidades del proyecto, incluyendo manejo de bases de datos, autenticación, seguridad, pruebas y análisis de datos.

3)- Comunidad activa y soporte: Tanto Python como Flask cuentan con comunidades de desarrollo muy activas, lo cual asegura la disponibilidad de documentación, foros y recursos de aprendizaje, así como actualizaciones frecuentes que refuerzan la seguridad del sistema.

4)- Ligereza y flexibilidad de Flask: A diferencia de otros frameworks más pesados, Flask sigue el principio de *microframework*, ofreciendo únicamente las funcionalidades esenciales para el desarrollo web, pero permitiendo añadir extensiones según los requerimientos del proyecto. Esto evita una sobrecarga innecesaria y otorga mayor control sobre la arquitectura de la aplicación.

5)- Escalabilidad progresiva: Flask es adecuado tanto para proyectos pequeños como para sistemas de mayor envergadura. Permite iniciar con una solución mínima viable (MVP) y, posteriormente, escalar en complejidad a medida que aumentan los requerimientos o el número de usuarios.

6)- Integración con bases de datos y APIs: Flask se adapta de manera sencilla a diferentes motores de base de datos y facilita la creación de APIs RESTful, lo cual resulta indispensable para un sistema de biblioteca digital que requiere comunicación con distintos servicios y módulos.

7)- Compatibilidad multiplataforma: Al estar basado en Python, el sistema puede ejecutarse en distintos entornos (Windows, Linux, macOS), además de ser fácilmente desplegable en servicios de hosting en la nube.

En conclusión, la combinación de **Python y Flask** permite garantizar un desarrollo eficiente, flexible y escalable del proyecto de biblioteca digital, asegurando una solución robusta, moderna y de fácil mantenimiento.

Arquitectura del sistema

El sistema de **Biblioteca Digital** está desarrollado sobre una arquitectura cliente-servidor, diseñada para ofrecer acceso remoto, administración centralizada de datos y escalabilidad.

Tecnologías principales:

-**Lenguaje backend:** Python

-**Framework web:** Flask (ligero, modular, flexible, ideal para proyectos de tamaño medio).

-**Base de datos:** Para entornos pequeños o pruebas: SQLite (base embebida y simple).

-**Frontend:** HTML5, CSS3, JavaScript (interfaz simple y accesible desde cualquier navegador).

-**Servidor web:** Werkzeug (integrado en Flask, aunque en producción puede usarse Gunicorn o uWSGI).

-**Protocolo de comunicación:** HTTP/HTTPS.

Componentes principales:

Cliente (usuario final):

Acceso mediante navegador web desde distintos dispositivos (PC, tablet, móvil).

Interacción con la interfaz para realizar consultas, préstamos, devoluciones y administración de datos.

Framework (Flask):

Gestiona las solicitudes de los clientes.

Contiene la lógica de negocio:

Gestión de usuarios.

Registro de préstamos y devoluciones.

Administración del catálogo.

Se comunica con la base de datos para almacenar y recuperar información.

Base de datos:

Estructura centralizada con las entidades:

Usuarios.

Libros.

Préstamos.

Devoluciones.

Garantiza integridad de datos, trazabilidad de cada ejemplar y control centralizado del inventario.

Características clave de la arquitectura

-**Centralización:** todos los datos están en una base común, evitando duplicaciones y pérdida de información.

-**Escalabilidad:** posibilidad de añadir más usuarios y libros sin rehacer el sistema.

-**Descentralización de acceso:** cualquier cliente puede conectarse vía internet, no limitado a una red local.

-**Mantenibilidad:** Flask permite ampliar funcionalidades en módulos sin afectar el núcleo del sistema.

Requisitos del sistema

Requisitos funcionales

Son las funciones que el sistema **debe cumplir obligatoriamente** para satisfacer las necesidades del usuario.

Gestión de usuarios:

Registrar nuevos usuarios.
Autenticación mediante credenciales (login).
Diferentes roles: administrador, usuario lector.

Gestión del catálogo de libros:

Registrar, editar y eliminar títulos.
Consultar disponibilidad de ejemplares en tiempo real.
Buscar por título, autor, género o palabras clave.

Gestión de préstamos y devoluciones:

Registrar préstamos indicando usuario, libro y fecha.
Calcular la fecha de devolución automáticamente en base al tiempo definido.
Registrar devoluciones con actualización automática de disponibilidad.
Generar alertas de retrasos (opcional).

Acceso remoto:

Los usuarios pueden acceder desde cualquier dispositivo conectado a internet.

Requisitos no funcionales

Son características de calidad y restricciones que debe cumplir el sistema.

Usabilidad:

Interfaz simple, intuitiva y accesible desde distintos dispositivos (responsiva).

Rendimiento:

Consultas al catálogo deben responder en menos de 2 segundos en condiciones normales.

Seguridad:

Autenticación obligatoria para acceder a funciones de préstamo y administración.
Cifrado de contraseñas en la base de datos.

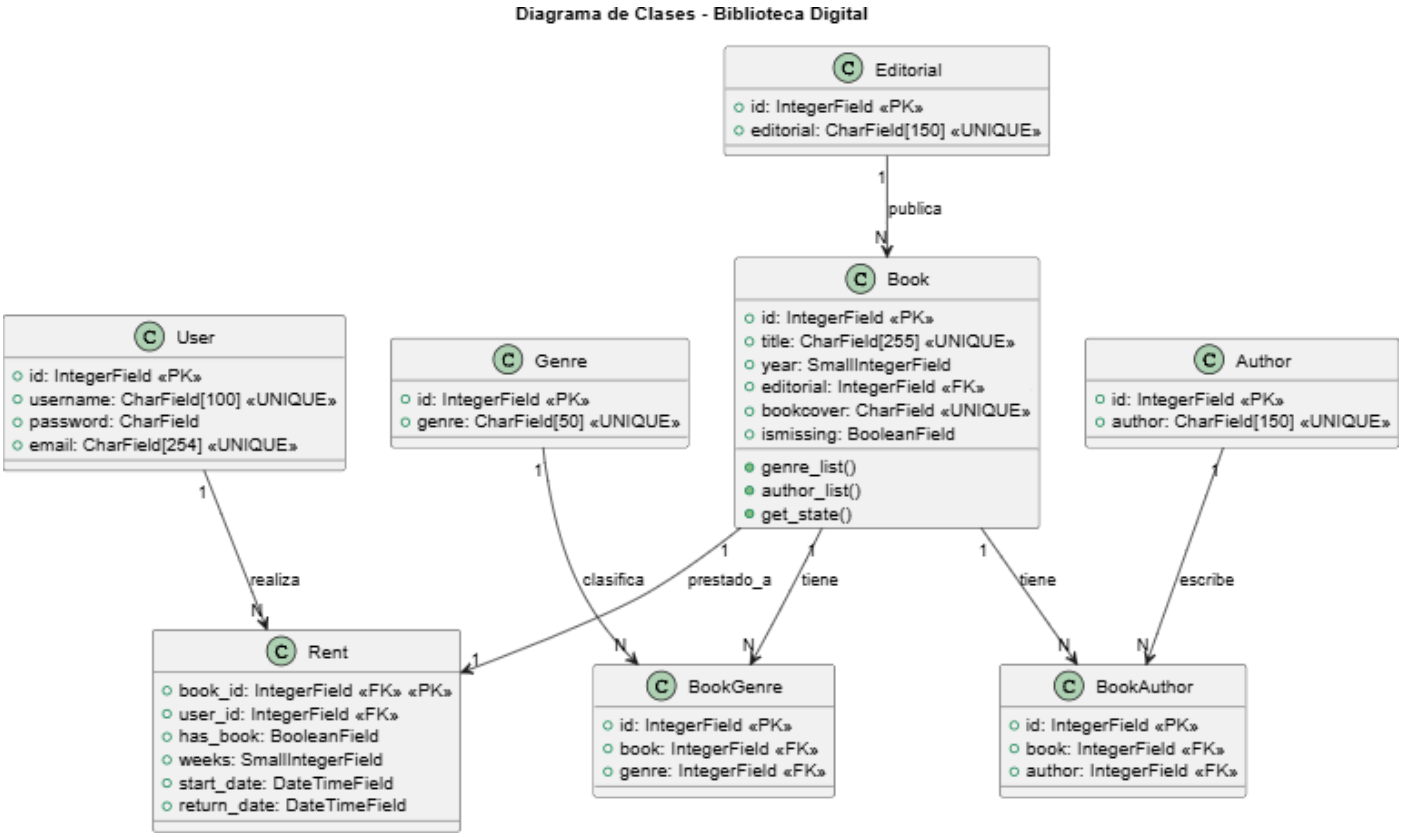
Escalabilidad:
Posibilidad de ampliar catálogo y número de usuarios sin necesidad de rehacer el sistema.

Mantenibilidad:
Código modular para facilitar actualizaciones y correcciones de errores.

Portabilidad:
Puede ejecutarse en distintos sistemas operativos (Linux, Windows, macOS).

Diseño de la base de datos

El diseño de la base de datos garantiza la **integridad, trazabilidad y control centralizado** de la información de la biblioteca. Se ha optado por un modelo relacional para representar las entidades clave y sus relaciones.



Entidades principales

User: representa a cada persona registrada en el sistema.
-ID **[Llave Primaria]**
-Nombre (username)
-Contraseña (password)
-Email (email)

Book: registro del catálogo de la biblioteca.
-ID **[Llave Primaria]**
-Titulo (title)
-Año (year)
-Id de Editorial (editorial) **[Llave Foránea]**
-Portada (bookcover)
-Libro no disponible físicamente (ismissing)

Rent: registro de cada transacción de préstamo.
-Id del Libro (book_id) **[Llave Primaria] [Llave Foránea]**
-Id del Usuario (user_id) **[Llave Foránea]**
-Usuario tiene el Libro (has_book)
-Duracion en Semanas (weeks)
-Fecha de comienzo del Préstamo (start_date)
-Fecha de retorno del Préstamo (return_date)

Author: representa a cada autor de Libros registrado en el sistema.
-ID **[Llave Primaria]**
-Nombre del Autor (author)

Editorial: representa a cada editorial de Libros registrado en el sistema.
-ID **[Llave Primaria]**
-Nombre de la Editorial (editorial)

Genre: representa a cada genero de Libros registrado en el sistema.
-ID **[Llave Primaria]**
-Nombre del Genero (genre)

BookGenre: registro de relaciones múltiples entre libros y géneros
-ID **[Llave Primaria]**
-ID del Libro (book) **[Llave Foránea]**
-ID del Genero (genre) **[Llave Foránea]**

BookAuthor: registro de relaciones múltiples entre libros y autores
-ID **[Llave Primaria]**
-ID del Libro (book) **[Llave Foránea]**
-ID del Autor (author) **[Llave Foránea]**

Relaciones

User – Rent: relación 1:N (un usuario puede tener varios préstamos).
Book – Rent: relación 1:1 (un libro especifico puede estar en un solo préstamo).
Editorial – Book: relación 1:N (una editorial puede relacionarse con más de un libro).
Book – Author: relación N:N[**BookAuthor**] (un libro puede relacionarse con múltiples autores y viceversa).
Book – Genre: relación N:N[**BookGenre**] (un libro puede relacionarse con múltiples géneros y viceversa).

Consideraciones de diseño

Integridad referencial: claves foráneas aseguran consistencia entre préstamos, usuarios y libros.

Normalización: se mantiene la redundancia mínima (ej. cantidad de libros disponibles se recalcula con base en préstamos activos).

Escalabilidad: el esquema permite agregar nuevas entidades (ej. reservas, categorías avanzadas, logs de actividad).

Flujos de uso y diagramas

Los flujos de uso representan cómo interactúan los diferentes actores (usuarios) con el sistema de la biblioteca digital.

Actores principales

Usuario lector

-Puede consultar el catálogo.
-Realizar préstamos de libros.
-Devolver libros.
-Consultar su historial de préstamos.

Administrador

-Gestiona el catálogo (alta, baja, modificaciones).
-Supervisa préstamos y devoluciones.
-Administra usuarios (crear, actualizar, asignar roles).

Casos de uso principales

Registro de un nuevo usuario

-El usuario ingresa nombre de usuario, email y contraseña.
-El sistema valida que no existan duplicados.
-Se guarda el registro en la tabla **User**.
-El usuario inicia sesión automáticamente.

Inicio de sesión

-El usuario ingresa usuario y contraseña.
-El sistema verifica en la base de datos.
-Si coincide -> se inicia sesión.
-Caso contrario -> muestra error.

Consulta del catálogo

-Usuario solicita vista /search.
-El sistema consulta libros en la tabla **Book**
-Devuelve resultados.
-Renderiza lista de libros.
-El sistema muestra disponibilidad de cada libro.

Préstamo de un libro

-El usuario selecciona un libro disponible.
-El sistema registra el préstamo, calcula la fecha de devolución y descuenta la disponibilidad.

Devolución de un libro

-El usuario devuelve el libro.
-El bibliotecario (o el sistema, en modo automatizado) registra la devolución.
-La disponibilidad del ejemplar aumenta.

Gestión del catálogo (solo administrador/bibliotecario)

-Agregar nuevos títulos al sistema.
-Modificar información de libros existentes.
-Eliminar libros (ej. ejemplares perdidos o deteriorados).

Consideraciones

El sistema está diseñado para que cada acción actualice automáticamente la base de datos.

Los roles permiten separar funciones críticas (ej. solo administradores pueden eliminar libros).

Se puede extender con funcionalidades opcionales como **reservas**, **renovaciones en línea** o **alertas automáticas por vencimiento**.