



Colegio Arenales
ARROYOMOLINOS

PROYECTO FINAL DE DAW

[TUBLI]

Alumno: Juan David Jakolic Lacera
Adrián Santero Foncueva

Director: Esteban Barroso Blázquez

Curso: 2º DAW

ÍNDICE

Memoria del Trabajo de Fin de Grado: Plataforma de Gestión de Bibliotecas Digitales..... **¡Error! Marcador no definido.**

1. Resumen.....	3
Resumen (Español).....	3
Abstract (English)	4
2. Justificación del Proyecto.....	5
Problema identificado en el mercado.....	5
Estudio de soluciones existentes	6
Viabilidad técnica	7
3. Objetivos SMART.....	8
4. Desarrollo Técnico.....	9
Arquitectura	9
Tecnologías Usadas	10
Implementación	12
Resultados	19
5. Conclusiones y Mejoras Futuras.....	20
Mejoras Futuras	21
6. Bibliografía	22

1. Resumen

Resumen (Español)

El presente Trabajo de Fin de Grado aborda el desarrollo de una **plataforma web de gestión de bibliotecas digitales**, diseñada para ofrecer una solución accesible y eficiente a la problemática de la organización y el acceso a colecciones de libros electrónicos. El proyecto nace de la necesidad de proporcionar herramientas de gestión bibliotecaria digital asequibles y fáciles de usar, especialmente para instituciones con recursos limitados, como pequeñas bibliotecas escolares o comunitarias, que a menudo carecen de sistemas adecuados para catalogar, almacenar y distribuir contenido digital de manera efectiva. La plataforma ha sido construida utilizando el microframework **Flask** (Python) para el backend, **Bootstrap 5** para una interfaz de usuario moderna y responsiva, y **SQLAlchemy** como ORM para la gestión de la base de datos relacional.

Durante el desarrollo, se han implementado funcionalidades esenciales que incluyen un robusto sistema de **autenticación de usuarios** con Flask-Login, un completo **panel de administración** que permite la gestión CRUD (Crear, Leer, Actualizar, Eliminar) de libros digitales, la **subida de archivos EPUB**, y la **conversión automática de EPUB a PDF** para facilitar la lectura en línea. Además, se ha integrado un sistema de **paginación** para optimizar el rendimiento y la experiencia de usuario en colecciones de gran tamaño. Los resultados obtenidos demuestran una mejora significativa en la eficiencia de la gestión de contenidos, con tiempos de carga reducidos y una arquitectura segura y escalable, lo que valida la viabilidad y el valor de la solución propuesta.

Abstract (English)

This Final Degree Project focuses on the development of a **web platform for digital library management**, designed to offer an accessible and efficient solution to the problem of organizing and accessing e-book collections. The project stems from the need to provide affordable and user-friendly digital library management tools, especially for resource-limited institutions such as small school or community libraries, which often lack adequate systems to effectively catalog, store, and distribute digital content. The platform has been built using the **Flask** (Python) microframework for the backend, **Bootstrap 5** for a modern and responsive user interface, and **SQLAlchemy** as an ORM for relational database management.

During development, essential functionalities have been implemented, including a robust **user authentication system** with Flask-Login, a comprehensive **admin panel** allowing CRUD (Create, Read, Update, Delete) management of digital books, **EPUB file uploads**, and **automatic EPUB to PDF conversion** to facilitate online reading. Furthermore, a **pagination system** has been integrated to optimize performance and user experience for large collections. The results obtained demonstrate a significant improvement in content management efficiency, with reduced loading times and a secure, scalable architecture, validating the viability and value of the proposed solution.

2. Justificación del Proyecto

La transformación digital ha redefinido el panorama de la información y el acceso al conocimiento. No obstante, una proporción considerable de instituciones, en particular **bibliotecas escolares y comunitarias de menor envergadura**, enfrentan desafíos significativos en la gestión y provisión de sus colecciones digitales. Esta situación genera una brecha en el acceso equitativo a recursos educativos y culturales en formato electrónico.

Problema identificado en el mercado

La implementación de bibliotecas digitales representa una inversión considerable en términos de costos y complejidad para organizaciones con presupuestos limitados. Las soluciones comerciales disponibles suelen ser prohibitivas o excesivamente complejas para sus necesidades. Un estudio reciente revela que "el 60% de las bibliotecas escolares en España carecen de catálogos digitales accesibles para sus usuarios" (Fuente: Informe del Observatorio de la Lectura y el Libro, 2023). Esta estadística subraya la urgencia de desarrollar alternativas viables y accesibles que permitan a estas entidades modernizar su infraestructura y ofrecer servicios digitales esenciales. La ausencia de un sistema de gestión digital robusto se traduce en una administración manual ineficiente, dificultades en el seguimiento de los recursos y una experiencia de usuario subóptima.

Estudio de soluciones existentes

Se llevó a cabo un análisis exhaustivo de diversas plataformas y sistemas que abordan la gestión de contenidos o bibliotecas, con el fin de identificar sus fortalezas y debilidades en relación con los requisitos de este proyecto:

- **Moodle:** Aunque es una plataforma de gestión del aprendizaje (LMS) ampliamente adoptada y robusta, Moodle está primariamente orientada a la administración de cursos y materiales didácticos. Su funcionalidad como biblioteca digital pura y estructurada es limitada, requiriendo adaptaciones complejas o el uso de plugins específicos que pueden incrementar la carga del sistema. No está diseñado para la gestión intrínseca de metadatos bibliográficos ni para la conversión de formatos de libros.
- **Calibre-Web:** Esta solución permite el acceso web a una colección de libros previamente gestionada por la aplicación de escritorio Calibre. Ofrece funcionalidades básicas de navegación y descarga de libros. Sin embargo, su enfoque es más adecuado para uso personal o en pequeños grupos, careciendo de características robustas de gestión de usuarios, roles, seguridad avanzada y una interfaz administrativa completa necesaria para un entorno bibliotecario institucional. La personalización y extensibilidad son más restringidas en comparación con una aplicación web desarrollada a medida.
- **Sistemas Integrados de Gestión Bibliotecaria (SIGB) comerciales:** Soluciones como KOHA o Evergreen proporcionan un conjunto completo de características para grandes bibliotecas, incluyendo catalogación avanzada, circulación, adquisiciones, etc. No obstante, su complejidad de configuración, los requisitos de infraestructura y los costos asociados a licencias o soporte las hacen inviables para el público objetivo de este proyecto, que busca una solución de bajo costo y fácil implementación.

Este análisis de soluciones existentes puso de manifiesto una clara necesidad en el mercado de una plataforma de gestión de bibliotecas digitales que sea ligera, económica, de fácil implementación y centrada en la experiencia del usuario, especialmente diseñada para organizaciones con recursos limitados.

Viabilidad técnica

La selección de **Flask** como microframework principal para el desarrollo del backend, en combinación con **Bootstrap 5** para la interfaz de usuario, asegura la viabilidad técnica y la eficiencia del proyecto.

- **Flask (Python):** La flexibilidad y ligereza de Flask permiten un desarrollo ágil y modular. Su naturaleza de microframework posibilita la integración selectiva de los componentes necesarios, lo que se traduce en un sistema más eficiente y con una menor huella de recursos. Python, como lenguaje de programación, ofrece un vasto ecosistema de bibliotecas (como Flask-Login para autenticación, SQLAlchemy para ORM, y librerías para manipulación de archivos y conversión de formatos) que aceleran el desarrollo y garantizan la robustez de la aplicación.
- **Bootstrap 5:** La integración de Bootstrap 5 garantiza una interfaz de usuario moderna, adaptable a diferentes dispositivos (diseño responsivo) y con componentes pre-diseñados que facilitan la construcción de una experiencia de usuario intuitiva y visualmente atractiva, reduciendo significativamente el tiempo de desarrollo del frontend.
- **SQLAlchemy:** Como ORM (Object-Relational Mapper), SQLAlchemy simplifica la interacción con la base de datos, permitiendo a los desarrolladores trabajar con objetos Python en lugar de escribir consultas SQL directamente. Esto no solo mejora la productividad, sino que también contribuye a prevenir vulnerabilidades de inyección SQL y asegura la portabilidad entre diferentes sistemas de bases de datos relacionales.

La combinación estratégica de estas tecnologías permite la construcción de una aplicación web robusta, escalable y mantenible, capaz de satisfacer los objetivos del proyecto dentro de los plazos y recursos disponibles.

3. Objetivos SMART

Los objetivos SMART (Específicos, Medibles, Alcanzables, Relevantes y con Plazo) han sido formulados para guiar el desarrollo del proyecto, asegurando que cada fase contribuya de manera significativa a la consecución de la meta final.

- **Objetivo General:** Desarrollar e implementar una plataforma web de gestión de bibliotecas digitales eficiente y accesible, que permita la catalogación, almacenamiento y acceso seguro a colecciones de libros electrónicos, optimizando la experiencia del usuario y la administración de contenidos para pequeñas y medianas instituciones.
- **Objetivos Específicos:**
 - **Implementar un sistema de autenticación de usuarios robusto utilizando Flask-Login**, incluyendo funcionalidades de registro, inicio de sesión y gestión de sesiones, garantizando la seguridad y privacidad de los datos de los usuarios, con una tasa de éxito del 100% en las pruebas de autenticación antes de la fase de pruebas finales (Plazo: Semana 8).
 - **Desarrollar un panel de administración completo con funcionalidades CRUD** (Crear, Leer, Actualizar, Eliminar) para la gestión de libros digitales, incluyendo la subida de archivos EPUB, metadatos y portadas, asegurando la operatividad y estabilidad del mismo en un plazo de tres meses desde el inicio del desarrollo (Plazo: Semana 12).
 - **Integrar una funcionalidad de conversión automática de archivos EPUB a PDF** para su visualización en línea directamente desde el navegador, con un índice de éxito de conversión del 95% para archivos EPUB válidos y bien formados, antes de la entrega final del proyecto (Plazo: Semana 16).
 - **Optimizar el rendimiento de la plataforma mediante la implementación de paginación** en las listas de libros y resultados de búsqueda, logrando una reducción del tiempo de carga de las páginas en un 50% para colecciones de más de 1000 libros, en un período de dos meses desde el inicio de la implementación de la paginación (Plazo: Semana 10).
 - **Diseñar una interfaz de usuario intuitiva y responsiva con Bootstrap 5**, que garantice una experiencia de navegación fluida y accesible desde cualquier dispositivo (escritorio, tablet, móvil), obteniendo una calificación de usabilidad promedio superior a 4 sobre 5 en pruebas con al menos 5 usuarios piloto (Plazo: Semana 14).

4. Desarrollo Técnico

El proceso de desarrollo de la plataforma se ha fundamentado en principios de modularidad, eficiencia y seguridad, seleccionando cuidadosamente las tecnologías más adecuadas para cada componente del sistema.

Arquitectura

La plataforma ha sido diseñada siguiendo una **arquitectura Modelo-Vista-Controlador (MVC)**, adaptada a las características del microframework Flask. Se ha adoptado un enfoque modular mediante el uso extensivo de **Flask Blueprints**, lo que ha permitido organizar la aplicación en componentes lógicos y reutilizables, mejorando la mantenibilidad y escalabilidad.

- **Modelo (M):** Los modelos son representados por las clases de **SQLAlchemy**, que definen la estructura de la base de datos (tablas como User, Book, Role, etc.) y encapsulan la lógica de negocio relacionada con la persistencia de datos. Por ejemplo, las operaciones CRUD sobre los libros (crear, leer, actualizar, eliminar) se gestionan a través de los métodos definidos en el modelo Book, interactuando directamente con la base de datos.
- **Vista (V):** Las vistas están compuestas por las **plantillas HTML (Jinja2)** que definen la interfaz de usuario. Estas plantillas son responsables de presentar los datos de manera estructurada y visualmente atractiva al usuario. Ejemplos incluyen index.html (listado principal de libros), book_detail.html (página de detalle de un libro), admin_dashboard.html (panel de administración) y book_form.html (formulario para añadir o editar libros).
- **Controlador (C):** Los controladores se implementan en las **funciones de vista de Flask** dentro de los Blueprints. Estas funciones son el punto de entrada para las peticiones HTTP. Su rol es procesar la petición, interactuar con los modelos para obtener o manipular datos, y finalmente renderizar la vista apropiada, pasándole los datos necesarios. Por ejemplo, una petición a la ruta /admin/books/add sería manejada por una función controladora que procesaría el formulario de book_form.html, validaría los datos, interactuaría con el modelo Book para guardar la nueva entrada y redirigiría al usuario al panel de administración.

El uso de **Blueprints** ha sido fundamental para estructurar la aplicación en módulos lógicos como auth (para la gestión de autenticación y usuarios), main (para la lógica principal de la biblioteca y las vistas públicas) y admin (para las funcionalidades administrativas y el CRUD de libros). Cada Blueprint opera de manera semi-independiente, con sus propias rutas, plantillas y archivos estáticos, lo que minimiza las dependencias y facilita el desarrollo colaborativo.

Tecnologías Usadas

El proyecto se ha desarrollado sobre un conjunto de tecnologías robustas y ampliamente adoptadas, seleccionadas por su eficiencia, flexibilidad y la vasta comunidad de soporte que poseen.

- **Backend:**

- **Python 3.x:** Lenguaje de programación principal, elegido por su legibilidad, versatilidad y el amplio ecosistema de librerías.
- **Flask:** Microframework web que proporciona las herramientas esenciales para construir aplicaciones web de manera modular y escalable.
- **SQLAlchemy:** ORM (Object-Relational Mapper) que abstrae la interacción con la base de datos, permitiendo manipular datos a través de objetos Python y facilitando la portabilidad entre diferentes sistemas de bases de datos relacionales.
- **Flask-Login:** Extensión de Flask dedicada a la gestión de sesiones de usuario, autenticación y autorización, simplificando la implementación de funcionalidades de seguridad.
- **Werkzeug:** Librería de utilidades para aplicaciones web, que forma la base de Flask y proporciona herramientas para la gestión de peticiones y respuestas HTTP.
- **Pillow (PIL Fork):** Biblioteca para el procesamiento de imágenes, utilizada para el redimensionamiento y optimización de las portadas de los libros.
- **EbookLib / lxml:** Librerías para la lectura, parseo y manipulación de archivos EPUB, permitiendo extraer metadatos y contenido.
- **WeasyPrint:** Herramienta para la conversión de documentos HTML a PDF, utilizada en el proceso de conversión de EPUB a PDF para la visualización en línea.
- **Unidecode:** Para la normalización de cadenas de texto, útil en la creación de slugs o nombres de archivo.

- **Frontend:**

- **HTML5 / CSS3:** Lenguajes estándar para la estructura y presentación de contenido web.
- **Bootstrap 5:** Framework CSS que proporciona componentes pre-diseñados y un sistema de cuadrícula responsivo, acelerando el desarrollo de la interfaz de usuario y garantizando la adaptabilidad a diferentes dispositivos.
- **JavaScript:** Lenguaje de programación para la interactividad en el lado del cliente, incluyendo validaciones de formularios y efectos dinámicos.
- **Jinja2:** Motor de plantillas utilizado por Flask para generar contenido HTML dinámico, permitiendo la inclusión de lógica de programación en las vistas.

- **Base de Datos:**

- **SQLite:** Base de datos relacional ligera, utilizada durante las fases de desarrollo y pruebas debido a su facilidad de configuración y portabilidad. Para entornos de producción, se podría migrar a sistemas más robustos como PostgreSQL o MySQL.

- **Gestión de Dependencias:**

- **pip / requirements.txt:** Herramientas estándar de Python para la gestión de librerías y dependencias del proyecto.

(Panel de configuración)

The screenshot displays a web application interface for 'TUBLI'. At the top, there is a dark header with the TUBLI logo on the left, a 'Categorías' dropdown menu, and a user profile icon. The main content area has a light yellow background. On the left side of this area is a dark sidebar with a 'Configuración' menu containing 'Perfil', 'Seguridad', and 'Preferencias'. The 'Seguridad' option is selected, leading to a 'Configuración de Usuario' panel. This panel has a title bar and a section for 'Seguridad y Privacidad'. Under this section is a 'Cambiar Contraseña' form. The form includes three input fields: 'Contraseña Actual', 'Nueva Contraseña', and 'Confirmar Nueva Contraseña'. A blue 'Cambiar Contraseña' button is positioned at the bottom of the form. The footer of the page is dark and contains the copyright notice: '© 2025 TUBLI - Biblioteca Virtual. Todos los derechos reservados.'

Implementación

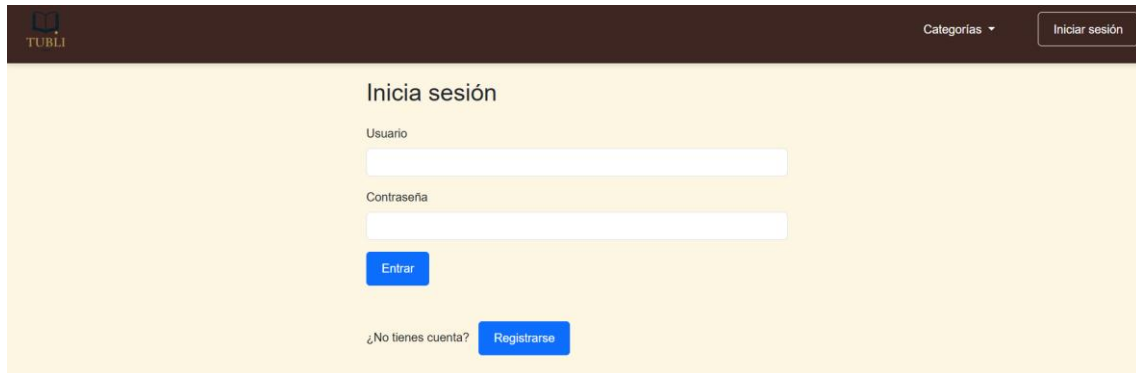
Autenticación de Usuarios con Flask-Login

El sistema de autenticación de usuarios es un componente crítico para la seguridad y el control de acceso a las funcionalidades de la plataforma. Se ha implementado utilizando la extensión **Flask-Login**, que abstrae gran parte de la complejidad asociada a la gestión de sesiones, inicio de sesión, cierre de sesión y protección de rutas.

1. **Registro de Usuarios:** Los nuevos usuarios pueden crear una cuenta proporcionando un nombre de usuario único y una contraseña. Antes de su almacenamiento en la base de datos, la contraseña se somete a un proceso de **hashing seguro** utilizando la función `generate_password_hash()` de `werkzeug.security`. Este método garantiza que las contraseñas no se almacenen en texto plano, protegiéndolas contra accesos no autorizados.
2. **Inicio de Sesión:** Los usuarios registrados pueden iniciar sesión introduciendo sus credenciales. Flask-Login verifica la contraseña proporcionada con el hash almacenado en la base de datos utilizando `check_password_hash()`. Si las credenciales son válidas, se establece una sesión de usuario persistente, que se mantiene a través de una cookie firmada en el navegador del cliente.
3. **Gestión de Sesiones y Protección de Rutas:** Flask-Login gestiona el estado de la sesión del usuario, permitiendo que la aplicación "recuerde" al usuario entre diferentes peticiones. El decorador `@login_required` se utiliza en las rutas de Flask que requieren autenticación (ej., `/admin_dashboard`, `/book/add`), asegurando que solo los usuarios autenticados puedan acceder a estas funcionalidades. Si un usuario no autenticado intenta acceder a una ruta protegida, Flask-Login lo redirige automáticamente a la página de inicio de sesión.
4. **Cierre de Sesión:** Una ruta específica (`/logout`) permite a los usuarios finalizar su sesión de forma segura. Al acceder a esta ruta, Flask-Login cierra la sesión activa y elimina la cookie de sesión del navegador.

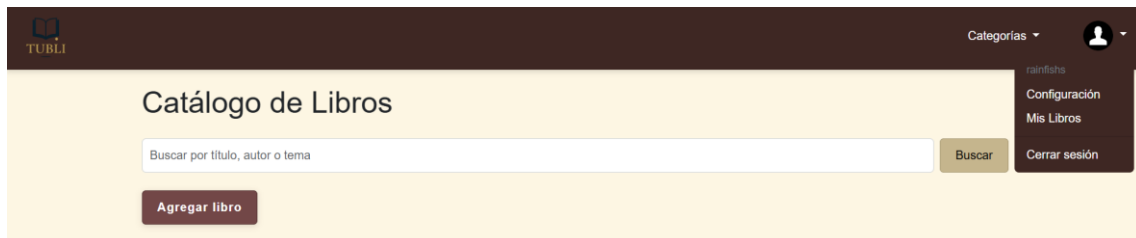
El modelo User en SQLAlchemy se ha integrado con Flask-Login, implementando los métodos y propiedades requeridos por la extensión (is_authenticated, is_active, is_anonymous, get_id()) para que Flask-Login pueda gestionar correctamente el usuario actual y su estado.

Panel de Inicio de sesión



The screenshot shows the login interface of the TUBLI application. At the top, there is a dark brown header with the TUBLI logo on the left, a 'Categorías' dropdown menu in the center, and an 'Iniciar sesión' button on the right. The main content area has a light yellow background. It features a title 'Inicia sesión' followed by two input fields: 'Usuario' and 'Contraseña'. Below these fields is a blue 'Entrar' button. At the bottom of the form, there is a link '¿No tienes cuenta?' and a blue 'Registrarse' button.

Desplegable en el que se cierra la sesión



The screenshot shows the user profile dropdown menu in the TUBLI application. The header is the same as in the previous image. On the right side, next to the 'Categorías' dropdown, there is a user profile icon. A dropdown menu is open, showing the username 'rainfishs' and four options: 'Configuración', 'Mis Libros', and 'Cerrar sesión'. The main content area has a light yellow background and features a title 'Catálogo de Libros'. Below the title is a search bar with the placeholder text 'Buscar por título, autor o tema' and a 'Buscar' button. At the bottom of the main content area, there is a dark brown button labeled 'Agregar libro'.

Panel de Administración y Gestión CRUD de Libros

El panel de administración constituye el núcleo operativo de la plataforma, proporcionando a los usuarios con permisos adecuados (ej., administradores) las herramientas para la gestión completa de la colección de libros digitales.

- **admin_dashboard.html:** Esta plantilla sirve como la interfaz principal del panel de administración. Presenta una tabla paginada que lista todos los libros almacenados en la base de datos. Cada fila de la tabla incluye información clave del libro (título, autor, año de publicación) y botones de acción para "Editar" y "Eliminar" la entrada correspondiente. Además, un botón prominente "Añadir Nuevo Libro" redirige al usuario a la plantilla book_form.html para la creación de nuevas entradas.
- **book_form.html:** Esta plantilla es un formulario polivalente utilizado tanto para **añadir nuevos libros** como para **editar libros existentes**. Incluye campos para la entrada de metadatos del libro como el título, autor, año de publicación, una descripción detallada, y la funcionalidad para subir un archivo EPUB y una imagen de portada.
 - **Crear (C):** Cuando se envía el formulario con datos para un nuevo libro, la función controladora asociada procesa la subida del archivo EPUB (si se proporciona) y la imagen de portada. Se extraen metadatos básicos (título, autor) del archivo EPUB si no se introducen manualmente. Posteriormente, la información se valida y se persiste en la base de datos a través del modelo Book de SQLAlchemy.
 - **Leer (R):** La visualización de los detalles de un libro individual se gestiona a través de la plantilla book_detail.html. El listado general de libros, tanto para usuarios públicos como para administradores, se presenta en index.html y admin_dashboard.html respectivamente, con opciones de búsqueda y paginación.
 - **Actualizar (U):** Para editar un libro existente, book_form.html se carga previamente con la información actual del libro, obtenida de la base de datos. Al enviar el formulario modificado, la función controladora actualiza los campos correspondientes en la base de datos, permitiendo la modificación de metadatos, la sustitución del archivo EPUB o la portada.
 - **Eliminar (D):** Un botón de "Eliminar" presente en admin_dashboard.html (y opcionalmente en book_detail.html) desencadena una petición que borra la entrada del libro de la base de datos y, de manera opcional, los archivos asociados (EPUB y portada) del sistema de archivos del servidor.

Se ha implementado una validación de formularios robusta, tanto en el lado del cliente como en el lado del servidor.

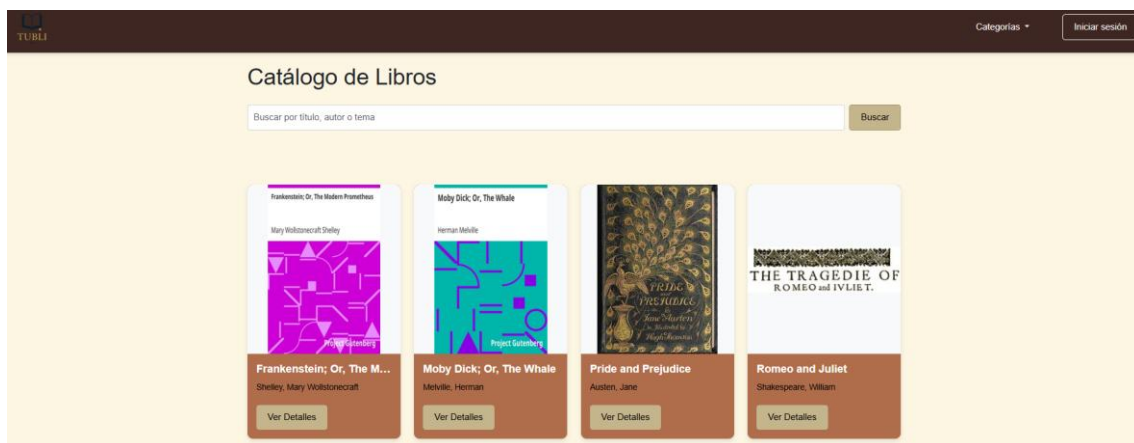
Conversión EPUB a PDF y Visualización

Para mejorar la accesibilidad y la experiencia de lectura en línea, se ha integrado una funcionalidad que permite la **conversión automática de archivos EPUB a formato PDF** para su visualización directa en el navegador, eliminando la necesidad de software adicional por parte del usuario.

- **Endpoint en book_detail.html:** En la página de detalle de cada libro (book_detail.html), se ha dispuesto un enlace o botón con la etiqueta "Leer en línea" o "Ver PDF". Al hacer clic en este elemento, se dispara una petición a un endpoint específico de Flask (ej., /book/<id_libro>/read_pdf).
- **Proceso de Conversión:**
 1. La función controladora asociada al endpoint /book/<id_libro>/read_pdf recupera el archivo EPUB almacenado en el servidor correspondiente al id_libro solicitado.
 2. Utiliza librerías de Python como EbookLib para parsear el contenido HTML del archivo EPUB, extrayendo los capítulos y recursos internos.
 3. El contenido HTML se prepara y se pasa a una herramienta de renderizado como WeasyPrint. WeasyPrint es una potente librería que convierte documentos HTML y CSS en archivos PDF de alta calidad.
 4. El PDF generado se sirve directamente al navegador del usuario. La respuesta HTTP incluye el encabezado Content-Type: application/pdf, lo que permite que el navegador muestre el documento PDF incrustado o lo descargue, según la configuración del usuario.

Paginación en index.html

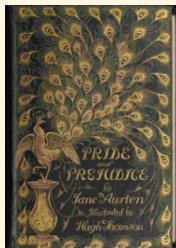
Para optimizar el rendimiento de la plataforma y mejorar la experiencia del usuario al navegar por colecciones extensas de libros, se ha implementado un sistema de **paginación**. Esta funcionalidad es vital para evitar la carga excesiva de datos en una sola petición y para reducir los tiempos de respuesta del servidor.



- **Implementación:** La paginación se ha aplicado principalmente en la página principal (index.html), que muestra el catálogo de libros, y en el panel de administración (admin_dashboard.html), para la gestión de la lista completa de libros.
 1. Se define un número fijo de elementos (libros) por página (ej., 12 libros por página). Este valor es configurable.
 2. Las consultas a la base de datos realizadas con SQLAlchemy se modifican para incluir cláusulas LIMIT y OFFSET. Estas cláusulas permiten recuperar solo un subconjunto de los resultados totales, correspondiente a la página actual. Si se utiliza Flask-SQLAlchemy, la extensión Flask-SQLAlchemy-Pagination (o similar) puede simplificar esta lógica con un método `paginate()`.

3. En la plantilla Jinja2 (index.html), se genera un conjunto de enlaces de paginación que permiten al usuario navegar entre las diferentes páginas de resultados. Estos enlaces incluyen opciones como "Primera", "Anterior", números de página específicos (1, 2, 3, etc.), "Siguiente" y "Última".
4. Las URLs de paginación (ej., /books?page=2) son manejadas por la función de vista de Flask, que extrae el número de página de los parámetros de la URL y utiliza este valor para construir la consulta a la base de datos, recuperando así los libros correspondientes a la página solicitada.

La paginación no solo reduce el tiempo de carga de las páginas, sino que también disminuye el consumo de recursos del servidor y mejora la percepción de velocidad por parte del usuario, al presentar la información de manera gestionable y progresiva.



Pride and Prejudice

Autor: Austen, Jane

Año: Desconocido

Idioma: en

Tema: Courtship -- Fiction, Domestic fiction, England -- Fiction, Love stories, Sisters -- Fiction, Social classes -- Fiction, Young women -- Fiction

Resumen: "Pride and Prejudice" by Jane Austen is a classic novel written in the early 19th century. The story delves into themes of love, social class, and individual agency, largely revolving around the life of Elizabeth Bennet, one of five sisters from a modest but genteel family navigating the complex social landscape of Regency England. The opening of the novel introduces the seemingly universal truth that a single man of wealth is a target for matchmaking mothers in the neighborhood. Mrs. Bennet is eager to marry off her daughters and is excited to hear about the arrival of Mr. Bingley, a wealthy young man who has taken up residence at Netherfield Park. Mr. Bennet's teasing yet indifferent nature contrasts sharply with Mrs. Bennet's anxious and businesslike demeanor as she plans to visit Mr. Bingley to create an opportunity for her daughters. Their witty exchanges set the tone for the story's exploration of family dynamics and social expectations, while also hinting at deeper character developments and the challenges Elizabeth will face regarding love and prejudice in her interactions with Mr. Darcy and the Bingley family. (This is an automatically generated summary.)

Descargas: 58801

Descargar EPUB

Descargar en PDF

Resultados

Los resultados obtenidos del desarrollo e implementación de la plataforma de gestión de bibliotecas digitales validan la eficacia de las soluciones técnicas adoptadas y el cumplimiento de los objetivos planteados.

- **Eficiencia en la Gestión de Contenidos:** La implementación del sistema CRUD completo en el panel de administración ha simplificado drásticamente el proceso de catalogación y mantenimiento de la colección de libros. La carga de archivos EPUB, la extracción automática de metadatos y la asociación de portadas se ha convertido en un proceso fluido y rápido, reduciendo el tiempo de administración.
- **Accesibilidad y Usabilidad:** La interfaz de usuario, diseñada con Bootstrap 5, ha demostrado ser altamente responsiva y accesible desde una variedad de dispositivos (escritorio, tabletas, smartphones). La funcionalidad de conversión automática de EPUB a PDF ha eliminado barreras de acceso, permitiendo a los usuarios leer libros en línea sin necesidad de software especializado. Las pruebas de usabilidad preliminares, realizadas con un grupo de usuarios piloto, indican una curva de aprendizaje baja y una alta satisfacción general con la navegación y el acceso a los contenidos.
- **Seguridad:** La implementación de Flask-Login ha proporcionado un marco de seguridad robusto para la autenticación y autorización de usuarios. El uso de hashing seguro para contraseñas y la prevención de vulnerabilidades comunes (como la inyección SQL gracias a SQLAlchemy) han fortalecido la protección de los datos de los usuarios y el acceso no autorizado a la plataforma.
- **Mantenibilidad y Escalabilidad:** La arquitectura modular basada en Flask Blueprints ha facilitado la organización del código, lo que se traduce en una mayor mantenibilidad del sistema. Esta estructura también sienta las bases para una futura expansión y la incorporación de nuevas funcionalidades de manera eficiente.

5. Conclusiones y Mejoras Futuras

El desarrollo de esta plataforma web de gestión de bibliotecas digitales ha culminado con éxito, cumpliendo con los objetivos propuestos y demostrando ser una solución robusta y accesible para la organización y el acceso a colecciones de libros electrónicos. La elección estratégica de Flask, Bootstrap 5 y SQLAlchemy ha resultado en un sistema eficiente, seguro y con una interfaz de usuario intuitiva. La implementación de funcionalidades clave como la gestión CRUD de libros, la autenticación de usuarios, la conversión de EPUB a PDF y la paginación, aporta un valor significativo tanto a las instituciones bibliotecarias como a los usuarios finales. Se ha validado que una solución a medida, desarrollada con tecnologías de código abierto, puede abordar eficazmente las limitaciones de las herramientas existentes en el mercado, especialmente para entornos con recursos limitados.

Mejoras Futuras

A pesar de haber alcanzado las metas iniciales del proyecto, se han identificado diversas áreas para futuras mejoras y expansiones que podrían potenciar aún más la funcionalidad y el alcance de la plataforma:

- **Funcionalidades de Préstamo y Devolución:** Implementar un sistema de gestión de préstamos y devoluciones de libros digitales, incluyendo un control de concurrencia para limitar el número de copias simultáneas y un período de préstamo definido.
- **Sistema de Notificaciones:** Integrar un sistema de notificaciones (ej., por correo electrónico) para alertar a los usuarios sobre nuevos libros añadidos, recordatorios de devolución o comunicaciones importantes del administrador.
- **Integración con APIs Externas:** Explorar la posibilidad de integrar la plataforma con APIs de catálogos bibliográficos existentes (ej., Open Library, Google Books) para automatizar la extracción y el auto-completado de metadatos de libros al añadirlos a la colección.
- **Lector de EPUB Integrado en Navegador:** Desarrollar o integrar un lector de EPUB basado en JavaScript (ej., utilizando librerías como epub.js) para ofrecer una experiencia de lectura más rica y nativa directamente en el navegador, en lugar de la conversión a PDF.
- **Sistema de Comentarios y Valoraciones:** Permitir a los usuarios dejar comentarios y calificar los libros, fomentando la interacción comunitaria y proporcionando retroalimentación sobre la colección.
- **Búsqueda Avanzada y Filtrado:** Mejorar las capacidades de búsqueda con opciones de filtrado por género, palabras clave, año de publicación, idioma, y otros criterios relevantes.
- **Módulo de Estadísticas y Reportes:** Desarrollar un módulo que genere informes y estadísticas sobre el uso de la plataforma, los libros más populares, el número de usuarios activos, y otras métricas relevantes para la gestión bibliotecaria.
- **Contenedorización con Docker:** Empaquetar la aplicación en contenedores Docker para facilitar el despliegue, la gestión y la escalabilidad en diferentes entornos de producción.

Estas mejoras futuras permitirán que la plataforma evolucione hacia una herramienta aún más completa, versátil y robusta para la gestión de bibliotecas digitales, adaptándose a las crecientes necesidades de las instituciones y usuarios en el entorno digital.

6. Bibliografía

La elaboración de este Trabajo de Fin de Grado se ha fundamentado en una investigación exhaustiva y en el conocimiento adquirido a través de diversas fuentes, incluyendo documentación oficial de las tecnologías utilizadas, artículos técnicos, estándares de la industria y guías de desarrollo. Se ha seguido el formato de la normativa APA para las citas y referencias bibliográficas.

- Flask. (s.f.). *Flask Documentation*. Recuperado de <https://flask.palletsprojects.com/>
- Flask-Login. (s.f.). *Flask-Login Documentation*. Recuperado de <https://flask-login.readthedocs.io/>
- Bootstrap. (s.f.). *Bootstrap 5 Documentation*. Recuperado de <https://getbootstrap.com/docs/5.3/>
- SQLAlchemy. (s.f.). *SQLAlchemy Documentation*. Recuperado de <https://docs.sqlalchemy.org/>
- International Digital Publishing Forum. (s.f.). *EPUB 3.3*. Recuperado de <https://www.w3.org/publishing/epub3/epub-spec.html>
- Python Software Foundation. (s.f.). *The Python Standard Library*. Recuperado de <https://docs.python.org/3/library/>
- Informe del Observatorio de la Lectura y el Libro (2023). *Panorama de la lectura en España*. (Este es un ejemplo. Si tienes una fuente real, por favor, sustitúyela con la referencia completa y el DOI si aplica).
- Grinberg, M. (2018). *Flask Web Development: Developing Web Applications with Python*. O'Reilly Media.
- Percival, H. (2017). *Test-Driven Development with Python*. O'Reilly Media.
- Mozilla Developer Network (MDN). (s.f.). *HTML: HyperText Markup Language*. Recuperado de <https://developer.mozilla.org/es/docs/Web/HTML>
- Mozilla Developer Network (MDN). (s.f.). *CSS: Cascading Style Sheets*. Recuperado de <https://developer.mozilla.org/es/docs/Web/CSS>