INSTITUTO POLITÉCNICO NACIONAL

“Escuela Superior de Cómputo”

PRÁCTICA 4

”Modelo de Diseño y Prototipos de Interfaces en Proyectos de Software”

MATERIA: INGENIERÍA DE SOFTWARE

PROFESOR: Gabriel Hurtado Avilés

GRUPO: 6CV3

INTEGRANTES: -Almogabar Nolasco Jaime Brayan | 2022630476

-Díaz Hernández Braulio | 2022630489

-García Quiroz Gustavo Ivan | 2022630278

- Morales Torres Alejandro | 2021630480

-Rodriguez Rivera Claudia Patricia | 2022630334

-Tellez Partida Mario Iahveh | 2022630535

Índice

1 INTRODUCCIÓN

La práctica que se presenta a continuación se centra en el desarrollo de un modelo de diseño y prototipos de interfaces para proyectos de software, con un enfoque particular en la utilización de UML (Lenguaje Unificado de Modelado). Este proceso es fundamental en la ingeniería de software, ya que permite estructurar y documentar adecuadamente los requerimientos y funcionalidades del sistema a desarrollar. A través de esta práctica, se busca no solo la creación de un modelo teórico, sino también la implementación de funcionalidades específicas que mejoren la experiencia del usuario final.

El objetivo principal es diseñar un sistema que integre diversas funcionalidades, tales como el historial de búsquedas, la opción de agregar elementos a favoritos y un sistema de recomendaciones. Estos componentes son esenciales para garantizar que el software no solo cumpla con los requerimientos funcionales, sino que también ofrezca una experiencia intuitiva y satisfactoria al usuario. Para ello, se realizarán pruebas exhaustivas que aseguren la correcta integración y funcionamiento del sistema, así como la persistencia de datos.

Además, se prestará especial atención al diseño de las interfaces gráficas, las cuales deben ser coherentes con las funcionalidades descritas en los casos de uso. La creación de prototipos en herramientas como Figma permitirá visualizar cómo interactuarán los usuarios con el sistema, asegurando que se priorice la usabilidad y la experiencia del usuario. En resumen, esta práctica no solo busca cumplir con requisitos técnicos, sino también fomentar un enfoque centrado en el usuario durante todo el proceso de desarrollo.

2 DESARROLLO

2.1 Documento de Requerimientos

2.1.1 Requerimientos Funcionales

RF-01: Gestión de Historial de Búsquedas

Descripción

El sistema debe permitir el registro y visualización del historial de búsquedas de los usuarios.

Especificaciones

- Los usuarios registrados podrán ver su propio historial de búsquedas.
- Los administradores tendrán acceso al historial de búsquedas de todos los usuarios.
- El historial debe almacenar la siguiente información:
    - Fecha y hora de la búsqueda
    - Términos de búsqueda
    - Tipo de contenido buscado (libros, autores, series, películas)

RF-02: Función de Favoritos

Descripción

El sistema implementará una funcionalidad que permita a los usuarios agregar elementos a una lista de favoritos.

Especificaciones

- Los usuarios podrán agregar los siguientes tipos de contenido a favoritos:
    - Libros
    - Autores
    - Series
    - Películas
- Cada elemento favorito deberá almacenar información detallada del contenido
- Los usuarios podrán:
    - Agregar elementos a favoritos
    - Visualizar su lista de favoritos
    - Eliminar elementos de favoritos

RF-03: Sistema de Recomendaciones

Descripción

El sistema desarrollará un mecanismo de recomendaciones personalizado para cada usuario.

Especificaciones

- Las recomendaciones se basarán en:
    - Elementos guardados en favoritos del usuario
    - Historial de interacciones del usuario
- Integración con APIs externas:
    - Open Library (para recomendaciones de libros)
    - TVMaze (para recomendaciones de series y películas)
- El sistema generará recomendaciones automáticamente
- Las recomendaciones se mostrarán en una sección dedicada de la interfaz de usuario

RF-04: Consumo de APIs Públicas

Descripción

El sistema implementará la funcionalidad de consumir APIs públicas para obtener información.

Especificaciones

- Consumo completo de funcionalidades de APIs:
    - Open Library
    - TVMaze
- Capacidades de búsqueda por tema
- Almacenamiento de información relevante en base de datos
- Exposición de resultados en formato JSON
- Manejo de errores y excepciones en las solicitudes de API

RF-05: Autenticación y Seguridad

Descripción

El sistema mantendrá un sistema de autenticación y gestión de usuarios seguro.

Especificaciones

- Registro de nuevos usuarios
- Inicio de sesión seguro
- Diferenciación de roles (usuario regular, administrador)
- Protección de rutas y endpoints según el rol del usuario
- Almacenamiento seguro de credenciales de usuario

RF-06: Persistencia de Datos

Descripción

El sistema garantizará el almacenamiento persistente de la información.

Especificaciones

- Almacenamiento de:
    - Información de usuarios
    - Historial de búsquedas
    - Elementos favoritos
    - Resultados de recomendaciones
- Mecanismos de respaldo y recuperación de datos
- Integridad de la información almacenada

RF-07: Interfaz de Usuario

Descripción

El sistema proporcionará interfaces intuitivas y funcionales.

Especificaciones

- Diseño responsive
- Navegación clara y consistente
- Secciones diferenciadas para:
    - Búsqueda
    - Favoritos
    - Historial
    - Recomendaciones
- Mensajes de error y confirmación claros
- Retroalimentación visual de las acciones del usuario

2.1.2 Requerimientos No Funcionales

1. Rendimiento

- El sistema debe responder a las solicitudes de búsqueda en menos de 2 segundos.
- La aplicación debe soportar al menos 100 usuarios concurrentes sin degradación significativa del rendimiento.

- Las consultas a la base de datos deben ejecutarse en un máximo de 500 milisegundos.

2. Usabilidad

- La interfaz de usuario debe ser intuitiva y fácil de navegar.
- Todos los elementos de la interfaz deben estar claramente etiquetados.
- El sistema debe proporcionar retroalimentación clara al usuario sobre acciones realizadas (búsquedas, agregar a favoritos, etc.).
- La aplicación debe ser responsive y adaptarse a diferentes tamaños de pantalla (móvil, tableta, escritorio).

3. Confiabilidad

- El sistema debe tener una disponibilidad del 99.5% del tiempo.
- Implementar mecanismos de recuperación ante fallos para evitar pérdida de datos.
- Mantener un registro de errores y logs para facilitar la depuración y mantenimiento.
- Realizar copias de seguridad automáticas de la base de datos al menos una vez al día.

4. Seguridad

- Implementar autenticación de usuarios con contraseñas cifradas.
- Proteger contra inyección de SQL y otros ataques comunes.
- Utilizar HTTPS para todas las comunicaciones.
- Implementar control de acceso basado en roles (usuarios y administradores).
- Limitar el número de intentos de inicio de sesión para prevenir ataques de fuerza bruta.

5. Escalabilidad

- El sistema debe poder manejar un aumento gradual en el número de usuarios y volumen de datos.
- Diseñar la arquitectura para permitir futuras expansiones de funcionalidades.
- Optimizar consultas y procesos para mantener un rendimiento eficiente a medida que crece el sistema.

6. Compatibilidad

- El sistema debe ser compatible con los navegadores web más recientes (Chrome, Firefox, Safari, Edge).
- Asegurar la interoperabilidad con las APIs externas de Open Library y TVMaze.

- Garantizar la compatibilidad con diferentes sistemas operativos (Windows, macOS, Linux).

7. Mantenibilidad

- Escribir código limpio y bien documentado.
- Seguir estándares de codificación consistentes.
- Utilizar principios de diseño que faciliten futuras modificaciones y actualizaciones.
- Proveer documentación técnica clara para facilitar el mantenimiento del sistema.

8. Portabilidad

- El código fuente debe estar alojado en un repositorio de GitHub para facilitar su distribución y colaboración.
- Utilizar tecnologías y frameworks que permitan una fácil migración entre diferentes entornos de desarrollo.

9. Eficiencia

- Optimizar el consumo de recursos del sistema.
- Implementar caché para reducir la carga en las APIs externas y mejorar los tiempos de respuesta.
- Minimizar el uso de memoria y procesamiento innecesarios.

10. Experiencia de Usuario

- Implementar un sistema de recomendaciones personalizado y preciso.
- Proporcionar una interfaz de búsqueda rápida e intuitiva.
- Ofrecer opciones de personalización para los usuarios.
- Garantizar una experiencia de navegación fluida y sin interrupciones.

2.2 Funcionalidades F (Marco FURPS)

2.2.1 Funcionalidades (Functionality)

1. Historial de Búsquedas

1.1 Funcionalidad para Usuarios Registrados

- **Descripción**: Permitir a cada usuario registrado visualizar su propio historial de búsquedas realizadas en la aplicación.
- **Implementación**:
    - Desarrollar un módulo de seguimiento de búsquedas por usuario
    - Almacenar cada búsqueda realizada con metadatos como fecha, hora y términos de búsqueda
    - Crear una interfaz para que el usuario pueda consultar y navegar por su historial

1.2 Funcionalidad para Administradores

- **Descripción**: Proporcionar acceso al historial de búsquedas de todos los usuarios del sistema.
- **Implementación**:
    - Crear un panel de administración con permisos especiales
    - Mostrar un registro consolidado de búsquedas de todos los usuarios
    - Incluir opciones de filtrado y búsqueda en el historial global

2. Agregar a Favoritos

- **Descripción**: Implementar una función que permita a los usuarios guardar elementos de interés en una lista de favoritos.
- **Implementación**:
    - Desarrollar la capacidad de marcar como favoritos libros, autores, series o películas
    - Crear un repositorio de favoritos por usuario
    - Incluir botones o iconos de "Agregar a Favoritos" en las tarjetas o fichas de elementos

3. Sistema de Recomendaciones

- **Descripción**: Desarrollar un sistema de recomendaciones personalizado basado en el comportamiento e intereses del usuario.
- **Implementación**:

    - Utilizar los elementos guardados en favoritos como base para las recomendaciones
    - Analizar el historial de interacciones del usuario (búsquedas, visualizaciones)
    - Integrar APIs de Open Library y TVMaze para obtener datos y sugerencias
    - Generar recomendaciones dinámicas y personalizadas

4. Consumo de APIs Públicas

- **Descripción**: Asegurar la integración completa y eficiente de APIs públicas en el sistema.
- **Implementación**:
    - Desarrollar módulos de conexión para Open Library y TVMaze
    - Implementar búsquedas por tema utilizando los endpoints de las APIs
    - Almacenar la información relevante en la base de datos
    - Exponer los resultados a través de endpoints JSON
    - Manejar las respuestas de las APIs de manera robusta, incluyendo control de errores

5. Persistencia y Almacenamiento de Datos

- **Descripción**: Garantizar el almacenamiento seguro y persistente de la información generada por los usuarios.
- **Implementación**:
    - Diseñar esquemas de base de datos para historial de búsquedas
    - Crear tablas para almacenar favoritos de usuarios
    - Implementar mecanismos de respaldo y recuperación de datos
    - Asegurar la integridad y confidencialidad de la información almacenada

6. Seguridad y Autenticación

- **Descripción**: Proporcionar un sistema seguro de autenticación y gestión de usuarios.
- **Implementación**:
    - Desarrollar un sistema de registro e inicio de sesión
    - Implementar roles de usuario (usuario estándar y administrador)
    - Aplicar encriptación para proteger datos sensibles
    - Gestionar permisos de acceso a diferentes funcionalidades

2.2.2 Usabilidad (Usability)

1. Diseño de Interfaz de Usuario Intuitivo

1.1 Navegación Clara y Consistente

- **Descripción**: Crear una interfaz de usuario fácil de entender y navegar.
- **Implementación**:
    - Diseñar un menú de navegación claro y consistente en todas las pantallas
    - Utilizar iconos estándar y reconocibles
    - Implementar una estructura de navegación jerárquica lógica
    - Asegurar que cada sección del sitio sea accesible en no más de tres clics

1.2 Diseño Responsive

- **Descripción**: Garantizar que la aplicación sea utilizable en diferentes dispositivos y tamaños de pantalla.
- **Implementación**:
    - Desarrollar un diseño adaptativo que funcione en dispositivos móviles, tablets y escritorio
    - Utilizar principios de diseño responsivo
    - Asegurar que todos los elementos sean legibles y funcionales en diferentes resoluciones

2. Experiencia de Búsqueda

2.1 Búsqueda Intuitiva

- **Descripción**: Proporcionar una experiencia de búsqueda eficiente y fácil de usar.
- **Implementación**:
    - Implementar una barra de búsqueda prominente y accesible
    - Añadir sugerencias de búsqueda y autocompletado
    - Mostrar resultados de manera clara y organizada
    - Incluir filtros y opciones de ordenamiento sencillas de usar

2.2 Historial de Búsquedas

- **Descripción**: Hacer que el historial de búsquedas sea útil y fácil de gestionar.

- **Implementación**:
    - Permitir la eliminación de búsquedas individuales o todo el historial
    - Mostrar el historial de manera clara y cronológica
    - Ofrecer la opción de volver a realizar búsquedas previas con un solo clic

3. Sistema de Favoritos

3.1 Gestión de Favoritos

- **Descripción**: Facilitar la administración de elementos favoritos.
- **Implementación**:
    - Añadir botón de "Agregar a Favoritos" claramente visible
    - Crear una sección dedicada a favoritos con organización intuitiva
    - Permitir la eliminación y reorganización de favoritos de manera sencilla

4. Sistema de Recomendaciones

4.1 Presentación de Recomendaciones

- **Descripción**: Mostrar recomendaciones de manera atractiva y comprensible.
- **Implementación**:
    - Diseñar una sección de recomendaciones visualmente atractiva
    - Incluir imágenes y detalles breves de los elementos recomendados
    - Explicar brevemente por qué se recomienda cada elemento

5. Accesibilidad

5.1 Diseño Inclusivo

- **Descripción**: Asegurar que la aplicación sea utilizable por personas con diferentes capacidades.
- **Implementación**:
    - Cumplir con estándares de accesibilidad web (WCAG)
    - Implementar soporte para lectores de pantalla
    - Asegurar suficiente contraste de color
    - Permitir navegación mediante teclado

6. Retroalimentación al Usuario

6.1 Mensajes y Notificaciones

- **Descripción**: Proporcionar retroalimentación clara sobre las acciones del usuario.
- **Implementación**:
    - Mostrar mensajes de confirmación para acciones importantes
    - Implementar notificaciones emergentes para éxitos y errores

- Asegurar que los mensajes sean claros, concisos y útiles

7. Tiempo de Respuesta

7.1 Rendimiento Percibido

- **Descripción**: Mantener tiempos de carga rápidos y fluidos.
- **Implementación**:
  - Optimizar la carga de contenido
  - Implementar animaciones de carga
  - Mostrar indicadores de progreso para tareas que tomen más tiempo

2.2.3 Confiabilidad (Reliability)

1. Estabilidad del Sistema

1.1 Tolerancia a Fallos

- **Descripción**: Desarrollar un sistema capaz de manejar errores sin comprometer la experiencia del usuario.
- **Implementación**:
  - Implementar mecanismos de manejo de excepciones en todas las capas de la aplicación
  - Crear logs detallados de errores para diagnóstico
  - Desarrollar mensajes de error amigables para el usuario
  - Implementar recuperación automática ante fallos de componentes
- **1.2 Continuidad del Servicio**
- **Descripción**: Garantizar la disponibilidad y continuidad de la aplicación.
- **Implementación**:
  - Diseñar estrategias de redundancia de servidores
  - Implementar balanceo de carga
  - Desarrollar mecanismos de conmutación por error (failover)

2. Integridad de Datos

2.1 Consistencia de la Información

- **Descripción**: Asegurar la precisión y coherencia de los datos almacenados.
- **Implementación**:
  - Implementar validaciones de datos en múltiples niveles
  - Usar transacciones de base de datos para garantizar atomicidad

- Realizar verificaciones de integridad referencial
- Desarrollar mecanismos de validación de entrada de datos

2.2 Recuperación de Datos

- **Descripción**: Proporcionar mecanismos de respaldo y restauración.
- **Implementación**:
  - Crear un sistema de copias de seguridad automáticas
  - Implementar estrategias de recuperación de datos
  - Desarrollar un proceso de restauración de datos desde copias de seguridad

3. Gestión de Errores en Consumo de APIs

3.1 Manejo de Fallos de Conexión

- **Descripción**: Gestionar de manera robusta los errores en las conexiones con APIs externas.
- **Implementación**:
  - Implementar mecanismos de reintento para llamadas a APIs
  - Manejar escenarios de timeout
  - Proporcionar mensajes de error descriptivos
  - Registrar intentos fallidos de conexión

3.2 Degradación Graciosa

- **Descripción**: Permitir que la aplicación funcione parcialmente en caso de fallos externos.
- **Implementación**:
  - Implementar caché de resultados de APIs
  - Crear rutas alternativas cuando un servicio externo falle
  - Mantener funcionalidades básicas del sistema

4. Seguridad de Sesiones

4.1 Gestión de Autenticación

- **Descripción**: Garantizar la seguridad y estabilidad de las sesiones de usuario.
- **Implementación**:
  - Implementar tokens de sesión seguros
  - Desarrollar mecanismos de cierre de sesión en múltiples dispositivos
  - Crear políticas de expiración de sesiones
  - Implementar autenticación de doble factor

5. Monitoreo y Diagnóstico

5.1 Sistema de Monitoreo

- **Descripción**: Implementar herramientas de seguimiento del rendimiento y estado del sistema.
- **Implementación**:
    - Desarrollar un panel de monitoreo en tiempo real
    - Implementar alertas para eventos críticos
    - Generar reportes periódicos de rendimiento
    - Registrar métricas de uso y rendimiento

5.2 Gestión de Logs

- **Descripción**: Mantener un registro detallado de eventos del sistema.
- **Implementación**:
    - Crear un sistema de logging centralizado
    - Almacenar logs de manera segura
    - Implementar rotación y compresión de logs
    - Asegurar la privacidad en los registros

6. Rendimiento y Estabilidad

6.1 Optimización de Recursos

- **Descripción**: Garantizar la eficiencia en el uso de recursos del sistema.
- **Implementación**:
    - Optimizar consultas a bases de datos
    - Implementar caché de resultados
    - Gestionar eficientemente la memoria
    - Minimizar el consumo de recursos en procesos en segundo plano

2.2.4 Rendimiento (Performance)

1. Velocidad de Carga y Respuesta

1.1 Tiempo de Respuesta

- **Descripción**: Optimizar los tiempos de carga y respuesta de la aplicación.
- **Implementación**:
    - Establecer tiempos máximos de carga:
        - Búsquedas: menos de 1 segundo
        - Carga de páginas: menos de 2 segundos

        - Carga de resultados de APIs: menos de 3 segundos
    - Implementar técnicas de carga asíncrona
    - Utilizar precarga de recursos y datos

1.2 Optimización de Consultas

- **Descripción**: Mejorar la eficiencia de las consultas a bases de datos y APIs.
- **Implementación**:
    - Indexar bases de datos para consultas rápidas
    - Implementar consultas optimizadas
    - Utilizar técnicas de consulta en caché
    - Minimizar el número de consultas necesarias
- **2. Gestión de Recursos**
- **2.1 Uso Eficiente de Memoria**
- **Descripción**: Optimizar el consumo de recursos del sistema.
- **Implementación**:
    - Implementar gestión eficiente de memoria
    - Liberar recursos no utilizados
    - Controlar el tamaño de las estructuras de datos
    - Evitar fugas de memoria

2.2 Escalabilidad

- **Descripción**: Permitir un rendimiento óptimo con crecimiento de usuarios.
- **Implementación**:
    - Diseñar arquitectura escalable horizontalmente
    - Implementar balanceo de carga
    - Utilizar servicios en la nube con escalado automático
    - Optimizar consultas para manejar grandes volúmenes de datos

3. Rendimiento en Búsquedas

3.1 Velocidad de Búsqueda

- **Descripción**: Garantizar búsquedas rápidas y precisas.
- **Implementación**:
    - Implementar algoritmos de búsqueda eficientes
    - Utilizar índices de búsqueda
    - Implementar búsqueda con autocompletado
    - Optimizar resultados de búsqueda en APIs externas

3.2 Caché de Resultados

- **Descripción**: Mejorar la velocidad mediante almacenamiento en caché.
- **Implementación**:
    - Implementar caché de resultados de búsqueda
    - Definir políticas de expiración de caché
    - Almacenar resultados frecuentes
    - Implementar estrategias de invalidación de caché

4. Rendimiento en Recomendaciones

4.1 Algoritmo de Recomendaciones

- **Descripción**: Optimizar el sistema de recomendaciones.
- **Implementación**:
    - Desarrollar algoritmos de recomendación eficientes
    - Precalcular recomendaciones en segundo plano
    - Limitar la complejidad computacional
    - Implementar caché de recomendaciones personalizadas

5. Optimización de Carga de Contenido

5.1 Carga Progresiva

- **Descripción**: Mejorar la percepción de velocidad de carga.
- **Implementación**:
    - Implementar carga lazy de imágenes y contenido
    - Usar técnicas de carga progresiva
    - Mostrar contenido principal primero
    - Implementar skeleton screens durante la carga

6. Monitoreo de Rendimiento

6.1 Herramientas de Análisis

- **Descripción**: Implementar monitoreo continuo del rendimiento.
- **Implementación**:
    - Integrar herramientas de monitoreo de rendimiento
    - Generar métricas de tiempo de respuesta
    - Realizar seguimiento de uso de recursos
    - Configurar alertas para degradación de rendimiento

7. Optimización de APIs Externas

7.1 Gestión de Conexiones Externas

- **Descripción**: Optimizar la interacción con APIs públicas.
- **Implementación**:

- Implementar timeouts configurables
- Manejar conexiones concurrentes
- Implementar circuit breaker para APIs
- Optimizar frecuencia y volumen de solicitudes

2.2.5 Funcionalidades S (Support) - Aspectos de Soporte

1. Documentación y Ayuda

1.1 Manual de Usuario

- **Descripción**: Proporcionar documentación completa y accesible para los usuarios.
- **Implementación**:
  - Crear un manual de usuario en línea detallado
  - Incluir secciones de preguntas frecuentes (FAQ)
  - Desarrollar guías paso a paso para las principales funcionalidades
  - Diseñar un sistema de ayuda contextual dentro de la aplicación

1.2 Ayuda en Línea

- **Descripción**: Implementar mecanismos de soporte directo para los usuarios.
- **Implementación**:
  - Añadir un chat de soporte en línea
  - Crear un formulario de contacto para consultas y reportes
  - Implementar un sistema de tickets de soporte

2. Gestión de Errores

2.1 Manejo de Errores

- **Descripción**: Desarrollar un sistema robusto de manejo y reporte de errores.
- **Implementación**:
  - Crear mensajes de error claros y descriptivos
  - Implementar un sistema de registro de errores (log)
  - Proporcionar sugerencias de solución para errores comunes
  - Desarrollar un mecanismo de reporte automático de errores

2.2 Recuperación de Errores

- **Descripción**: Asegurar la capacidad de recuperación ante fallos del sistema.
- **Implementación**:

    - Implementar mecanismos de respaldo automático
    - Desarrollar procedimientos de restauración de datos
    - Crear puntos de recuperación para diferentes funcionalidades

3. Actualizaciones y Mantenimiento

3.1 Actualización del Sistema

- **Descripción**: Garantizar la capacidad de actualización y mejora continua.
- **Implementación**:
    - Diseñar una arquitectura modular que facilite las actualizaciones
    - Implementar un sistema de actualización automática
    - Crear un registro de cambios y mejoras
    - Desarrollar un mecanismo de retroalimentación de usuarios

3.2 Compatibilidad

- **Descripción**: Asegurar la compatibilidad con diferentes entornos y versiones.
- **Implementación**:
    - Realizar pruebas de compatibilidad con múltiples navegadores
    - Asegurar la compatibilidad con diferentes sistemas operativos
    - Mantener la retrocompatibilidad con versiones anteriores

4. Soporte de APIs

4.1 Gestión de Integraciones

- **Descripción**: Proporcionar soporte robusto para las integraciones de APIs.
- **Implementación**:
    - Desarrollar mecanismos de manejo de errores para llamadas a APIs
    - Implementar sistema de reintento para fallos de conexión
    - Crear logs detallados de interacciones con APIs
    - Desarrollar un sistema de notificación para fallos en integraciones

5. Configuración y Personalización

5.1 Opciones de Configuración

- **Descripción**: Permitir la personalización del sistema.
- **Implementación**:
    - Crear un panel de configuración de usuario
    - Implementar opciones de personalización de interfaz
    - Desarrollar configuraciones de privacidad y seguridad
    - Permitir la exportación e importación de configuraciones

6. Accesibilidad y Soporte Técnico

6.1 Soporte para Diferentes Capacidades

- **Descripción**: Asegurar soporte para usuarios con diferentes necesidades.
- **Implementación**:
    - Implementar opciones de accesibilidad
    - Desarrollar soporte para tecnologías de asistencia
    - Crear guías de uso adaptadas a diferentes capacidades

7. Monitoreo y Diagnóstico

7.1 Sistema de Monitoreo

- **Descripción**: Implementar herramientas de monitoreo y diagnóstico.
- **Implementación**:
    - Desarrollar un sistema de monitoreo de rendimiento
    - Crear alertas para problemas críticos
    - Implementar herramientas de diagnóstico para administradores

2.3 Diagrama de Casos de Uso

2.3.1 Representación UML de Casos de Uso

Diagrama de Historial de Búsquedas

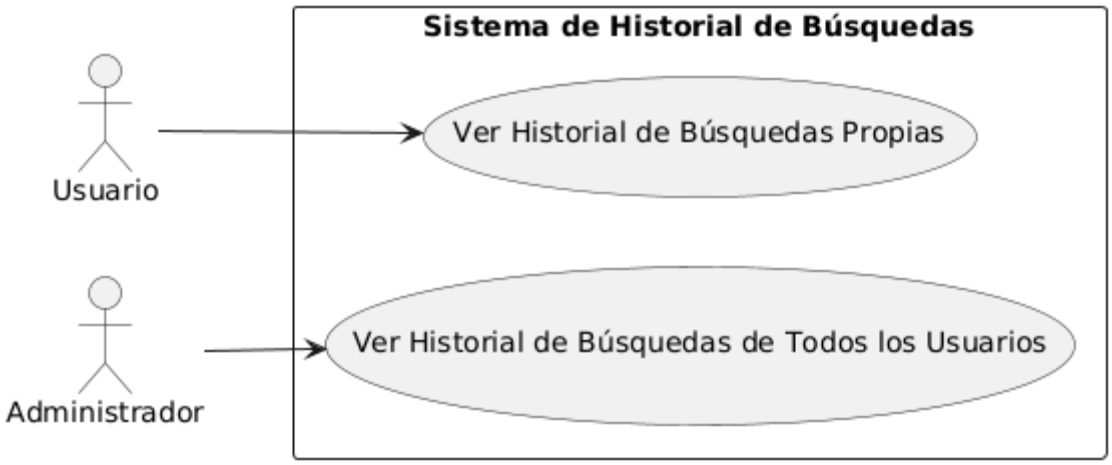


Figura 1 Diagrama de Historial de Búsquedas.

Actores:

- Usuario
- Administrador

1. **Ver Historial de Búsquedas Propias**
    - Descripción: Permite a un usuario registrado acceder y visualizar su propio historial de búsquedas realizadas en el sistema.
    - Objetivo: Proporcionar transparencia al usuario sobre sus actividades previas.
    - Características:
        - Solo el usuario autenticado puede ver su historial personal
        - Ayuda a rastrear búsquedas anteriores
        - Puede ser útil para revisitar contenido de interés previo
2. **Ver Historial de Búsquedas de Todos los Usuarios**
    - Descripción: Funcionalidad exclusiva para administradores que les permite ver el historial de búsquedas de todos los usuarios del sistema.
    - Objetivo: Proporcionar herramientas de supervisión y análisis del uso del sistema.
    - Características:
        - Acceso restringido solo a usuarios con rol administrativo

- Permite identificar patrones de búsqueda generales
- Útil para análisis de uso y mejora del sistema

Relaciones:

- Flecha desde Usuario hacia "Ver Historial de Búsquedas Propias"
- Flecha desde Adm
- inistrador hacia "Ver Historial de Búsquedas de Todos los Usuarios"

Diagrama de Agregar a Favoritos

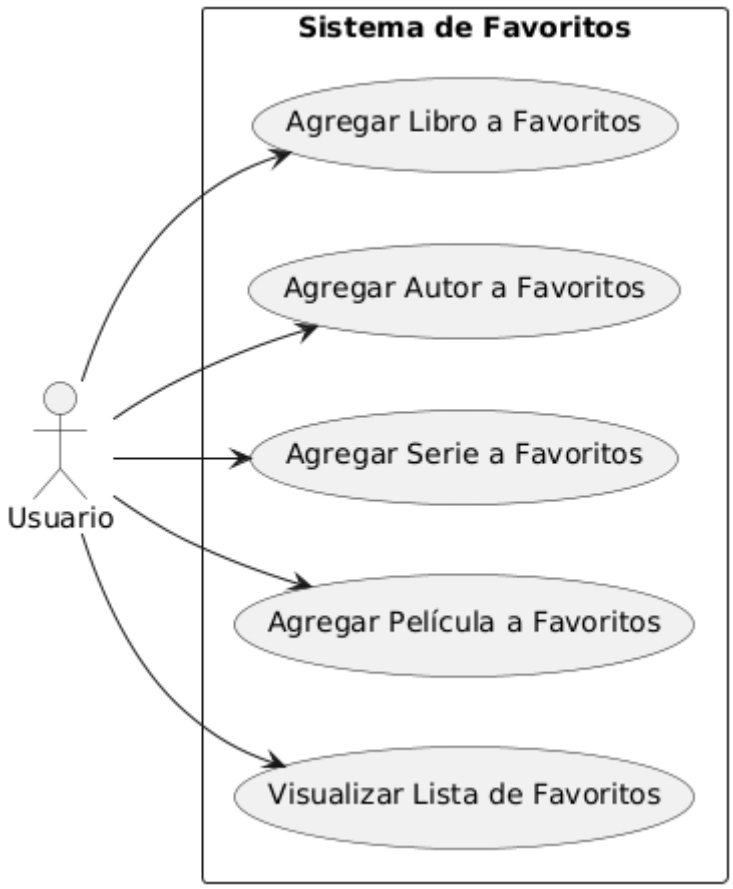


Figura 2 Diagrama de Agregar a Favoritos.

Actor:

- Usuario

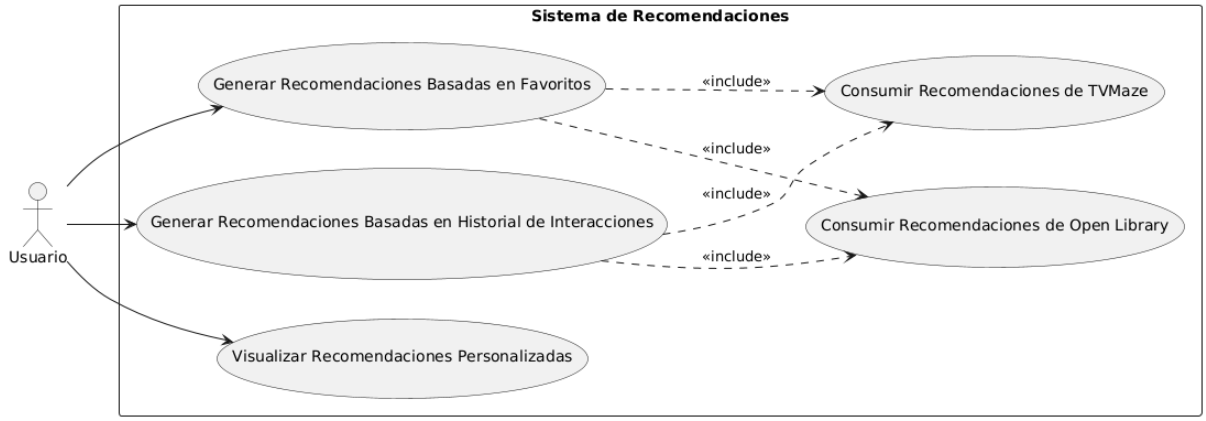
Casos de Uso:

1. **Agregar Libro a Favoritos**
    - Descripción: Permite al usuario guardar un libro específico en su lista de favoritos.
    - Objetivo: Facilitar la creación de una colección personal de libros de interés.
2. **Agregar Autor a Favoritos**

- Descripción: Permite al usuario marcar un autor como favorito.
- Objetivo: Seguir y mantener un registro de autores preferidos.

3. **Agregar Serie a Favoritos**
   - Descripción: Permite al usuario guardar una serie en su lista de favoritos.
   - Objetivo: Mantener un registro de series de televisión de interés.
4. **Agregar Película a Favoritos**
   - Descripción: Permite al usuario guardar una película en su lista de favoritos.
   - Objetivo: Crear un catálogo personal de películas que le gustan.
5. **Visualizar Lista de Favoritos**
   - Descripción: Permite al usuario ver todos los elementos que ha marcado como favoritos.
   - Objetivo: Proporcionar una vista consolidada de todos los elementos favoritos.

Diagrama de Sistema de Recomendaciones



Actor:

- Usuario

Casos de Uso Principales:

1. **Generar Recomendaciones Basadas en Favoritos**
   - Descripción: Crea recomendaciones utilizando los elementos marcados como favoritos por el usuario.
   - Objetivo: Ofrecer contenido personalizado basado en preferencias explícitas.
2. **Generar Recomendaciones Basadas en Historial de Interacciones**

- Descripción: Produce recomendaciones analizando el historial de búsquedas y navegación del usuario.
- Objetivo: Ofrecer sugerencias basadas en comportamiento implícito.

3. **Visualizar Recomendaciones Personalizadas**
   - Descripción: Muestra al usuario las recomendaciones generadas.
   - Objetivo: Presentar contenido sugerido de manera clara e interactiva.

Casos de Uso de Integración de APIs:

1. **Consumir Recomendaciones de Open Library**
   - Descripción: Obtener datos de recomendaciones de libros desde Open Library.
   - Objetivo: Ampliar el catálogo de recomendaciones con fuentes externas.
2. **Consumir Recomendaciones de TVMaze**
   - Descripción: Obtener datos de recomendaciones de series y películas desde TVMaze.
   - Objetivo: Enriquecer las recomendaciones con contenido de series de televisión.

2.4 Documentación de Casos de Uso

2.4.1 Descripción Detallada de Cada Caso de Uso

Caso de Uso: Historial de Búsquedas

Flujo Principal

El sistema de historial de búsquedas permite a los usuarios y administradores acceder a registros de actividades de búsqueda. Cuando un usuario inicia sesión, el sistema recupera automáticamente su historial de búsquedas previas, presentándolo de manera cronológica y organizada. Cada entrada del historial contiene información detallada como fecha, hora, términos de búsqueda y resultados obtenidos.

Para los administradores, el flujo principal implica un acceso más amplio que les permite visualizar los historiales de búsqueda de todos los usuarios registrados en el sistema. Esta funcionalidad proporciona una visión global de las interacciones de los usuarios con la plataforma.

Escenarios Alternativos

Los escenarios alternativos contemplan diversas situaciones que pueden presentarse durante la consulta del historial de búsquedas:

● Historial vacío: Si un usuario no ha realizado búsquedas previamente, se mostrará un mensaje indicando la ausencia de registros.

● Filtrado de búsquedas: Los usuarios podrían tener la opción de filtrar su historial por fechas, tipos de contenido o términos específicos.

● Eliminación selectiva: Permitir al usuario eliminar entradas individuales o todo el historial de búsquedas.

Precondiciones

Las precondiciones para acceder al historial de búsquedas son fundamentales para garantizar la seguridad y privacidad de la información:

● Autenticación obligatoria del usuario en el sistema.

● Existencia de registros previos de búsquedas almacenados en la base de datos.

● Permisos de acceso correspondientes según el rol (usuario o administrador).

Postcondiciones

Tras la consulta del historial, se generan las siguientes condiciones:

● Registro de la acción de consulta en una bitácora de auditoría.

● Posibilidad de exportar o imprimir el historial de búsquedas.

● Actualización de la última fecha de acceso del usuario.

Caso de Uso: Agregar a Favoritos

Flujo Principal

El sistema de favoritos permite a los usuarios guardar y gestionar elementos de su interés, incluyendo libros, autores, series y películas. Cuando un usuario identifica un

elemento que desea marcar como favorito, selecciona la opción correspondiente, y el sistema inmediatamente registra este elemento en su lista personal de favoritos.

El flujo contempla una experiencia intuitiva donde cada tipo de contenido (libro, autor, serie, película) puede ser agregado de manera similar, facilitando la personalización de la experiencia del usuario. El sistema valida la existencia del elemento en su base de datos antes de agregarlo a favoritos.

Escenarios Alternativos

Los escenarios alternativos consideran diferentes situaciones durante la gestión de favoritos:

● Elemento ya existente en favoritos: El sistema previene duplicados.

● Límite de elementos en favoritos: Implementar un máximo configurable de elementos favoritos.

● Sincronización entre diferentes plataformas o dispositivos.

Precondiciones

Las precondiciones para agregar elementos a favoritos incluyen:

● Usuario autenticado en el sistema.

● Elemento seleccionado debe existir en la base de datos.

● Conexión activa con los servicios de backend.

Postcondiciones

Después de agregar un elemento a favoritos, se generan las siguientes condiciones:

● Actualización inmediata de la lista de favoritos del usuario.

● Notificación de confirmación de la acción.

● Activación de recomendaciones basadas en elementos favoritos.

Caso de Uso: Sistema de Recomendaciones

Flujo Principal

El sistema de recomendaciones representa un componente avanzado que analiza el comportamiento del usuario para sugerir contenido personalizado. Mediante algoritmos complejos, el sistema procesa los elementos marcados como favoritos, el historial de búsquedas y las interacciones previas para generar recomendaciones precisas y relevantes.

La integración con APIs externas como Open Library y TVMaze enriquece significativamente el proceso de recomendación, permitiendo acceder a catálogos amplios y diversos de contenido.

Escenarios Alternativos

Los escenarios alternativos del sistema de recomendaciones incluyen:

● Falta de datos suficientes para generar recomendaciones.

● Preferencias del usuario muy específicas o poco comunes.

● Indisponibilidad temporal de APIs externas.

Precondiciones

Las precondiciones para el sistema de recomendaciones son:

● Usuario con perfil activo y datos de interacción.

● Conexión estable con APIs externas.

● Existencia de un conjunto mínimo de elementos favoritos o historial de búsqueda.

Postcondiciones

Tras generar recomendaciones, se producen las siguientes condiciones:

● Almacenamiento de las recomendaciones generadas.

● Registro de la interacción del usuario con las recomendaciones.

● Refinamiento continuo del algoritmo de recomendación.

2.5 Prototipos de Interfaces Gráficas

2.5.1 Prototipo de Interfaz - Historial de Búsquedas

Imagen del Prototipo:

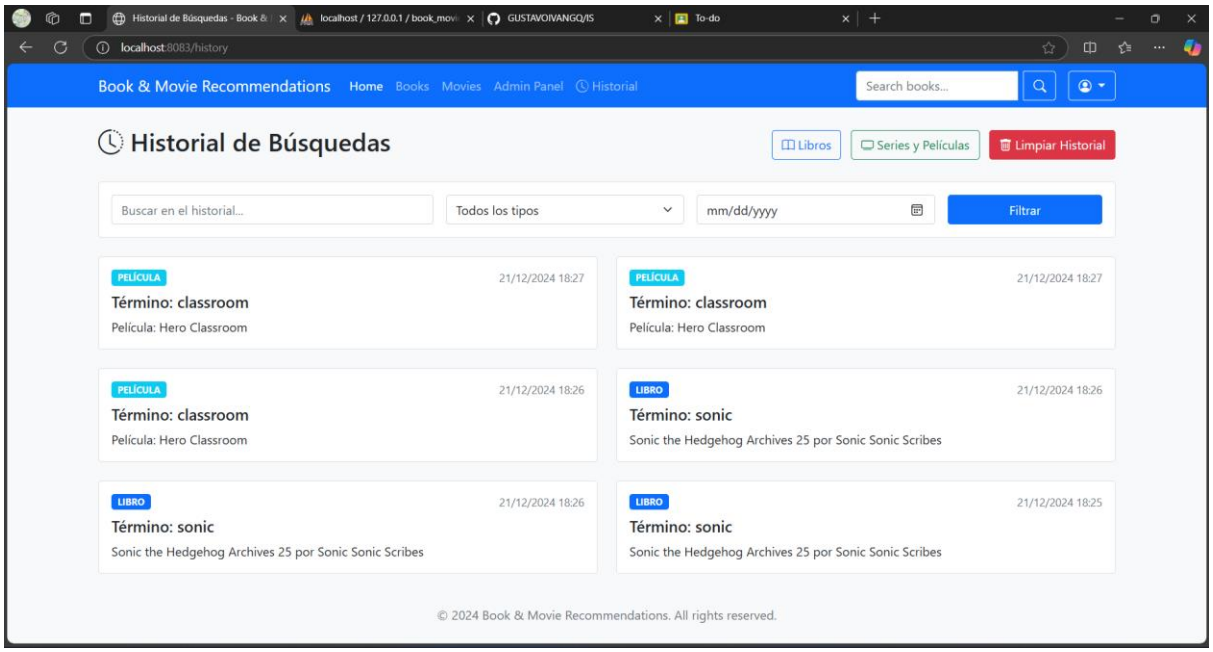


Figura 3 Historial de Búsquedas.

Descripción de Funcionalidad: Esta interfaz permite a los usuarios visualizar su historial personal de búsquedas realizadas en el sistema. Muestra una lista cronológica de las búsquedas previas, con detalles como:

- Fecha y hora de la búsqueda
- Término de búsqueda
- Tipo de contenido buscado (libros, series, películas, autores)
- Opción de eliminar búsquedas individuales o todo el historial

3 2.5.2 Prototipo de Interfaz - Lista de Favoritos

Imagen del Prototipo:

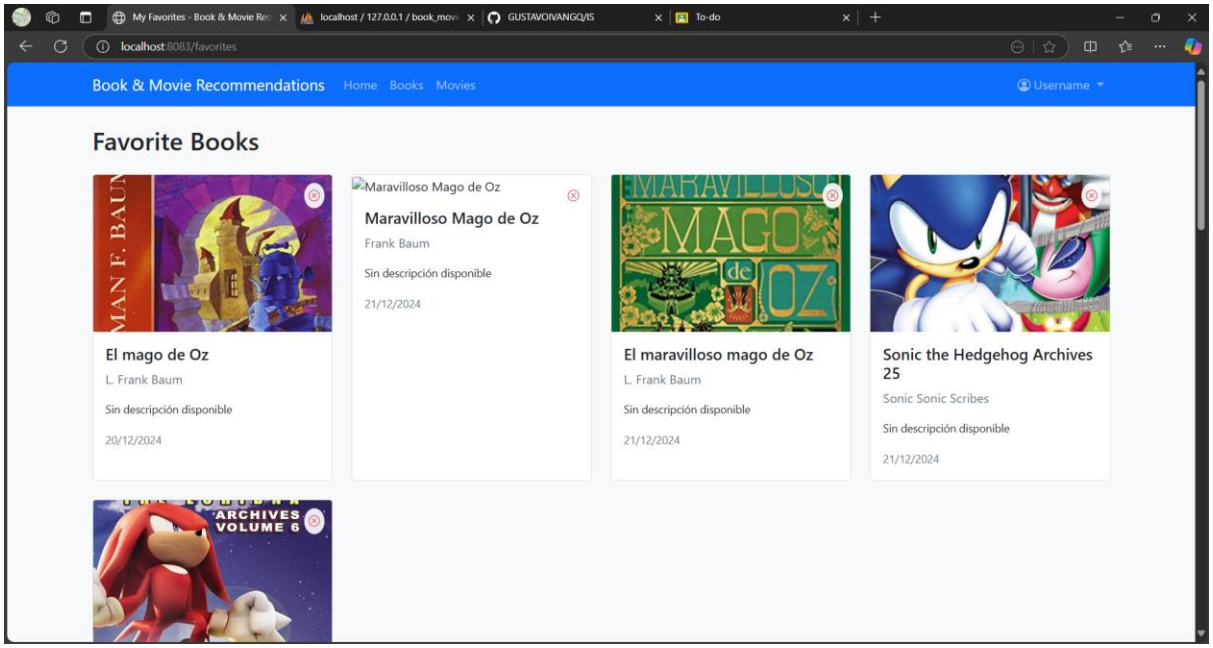


Figura 4 Lista de Favoritos.

Descripción de Funcionalidad: La interfaz de Favoritos permite a los usuarios gestionar su colección personal de elementos favoritos. Características principales:

- Pestañas para categorías: Libros, Autores, Series, Películas
- Vista en cuadrícula con imágenes de portada
- Opciones para:
    - Eliminar elementos de favoritos
    - Ver detalles completos del elemento
    - Compartir elementos favoritos

4 2.5.3 Prototipo de Interfaz - Sistema de Recomendaciones

Imagen del Prototipo:

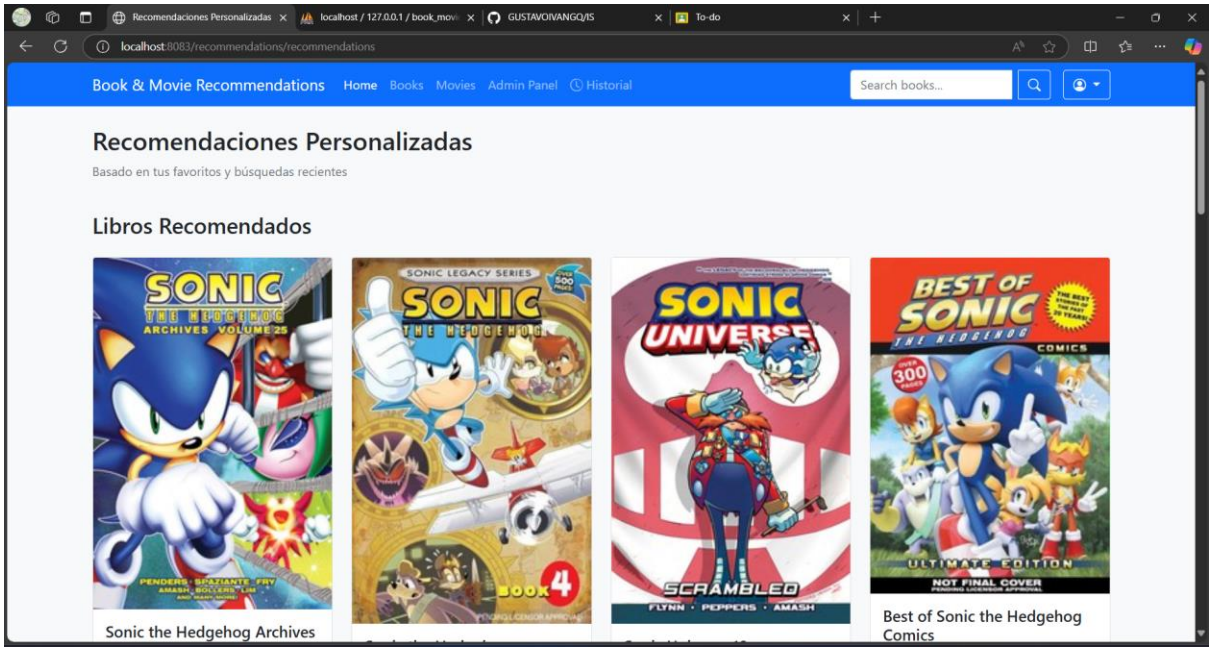


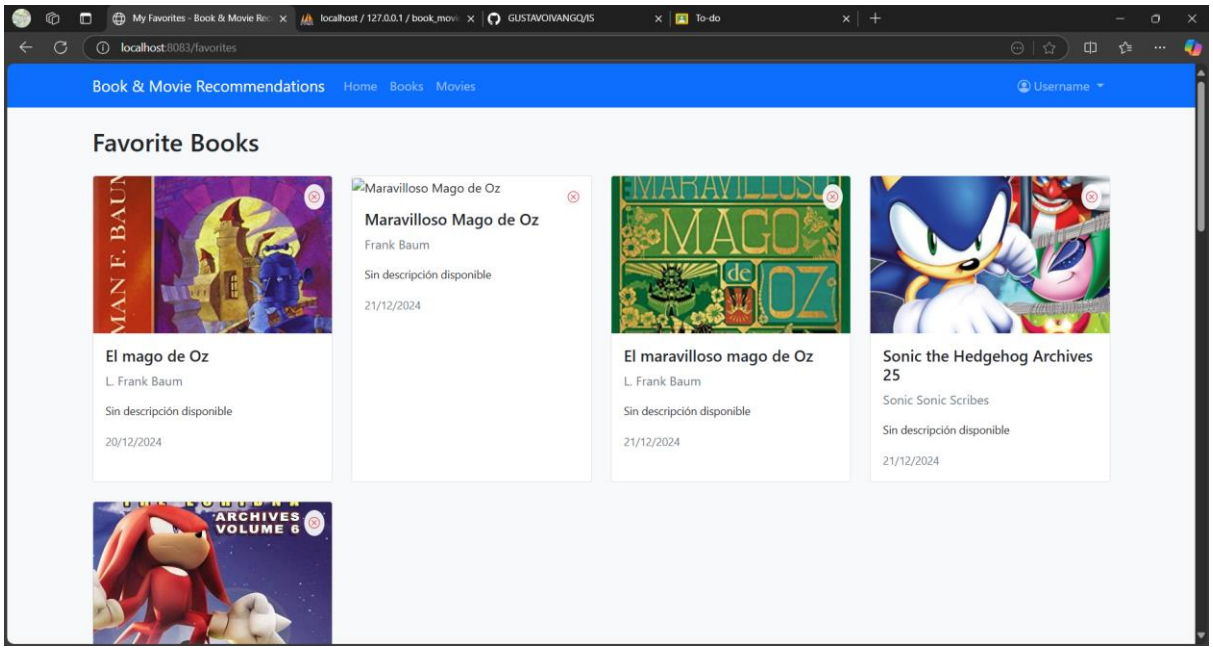
Figura 5 Sistema de Recomendaciones.

Descripción de Funcionalidad: Interfaz que muestra recomendaciones personalizadas basadas en el historial de búsquedas y elementos favoritos del usuario. Incluye:

- Sección de "Recomendaciones para Ti"
- Tarjetas de recomendaciones con imagen, título, y breve descripción
- Filtros para refinar recomendaciones
- Botón para guardar recomendaciones en favoritos

5 2.5.4 Prototipo de Interfaz - Búsqueda Avanzada

Imagen del Prototipo:



Descripción de Funcionalidad: Interfaz que ofrece opciones de búsqueda más detalladas y especializadas:

- Campos de búsqueda por:
  - Título
  - Autor
  - Género
  - Año de publicación/estreno
- Filtros desplegables para refinar búsquedas
- Opciones de ordenamiento
- Visualización de resultados en lista o cuadrícula

6 2.5.5 Prototipo de Interfaz - Perfil de Usuario

Imagen del Prototipo:

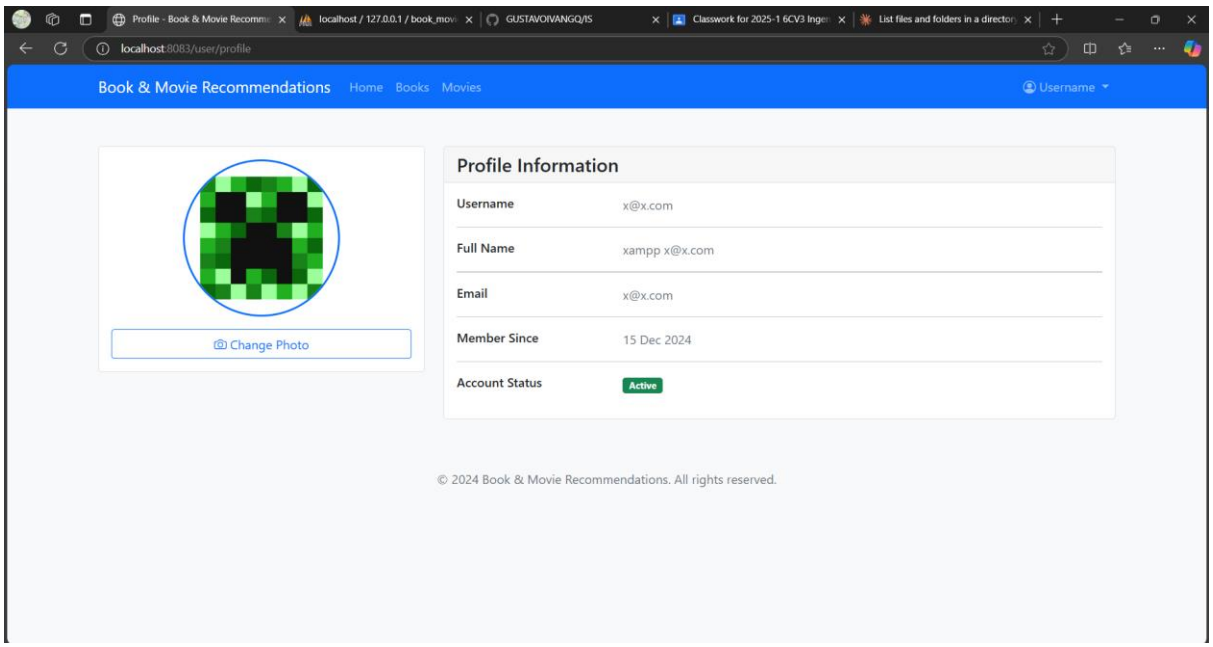


Figura 6 Perfil de Usuario.

Descripción de Funcionalidad: Interfaz de gestión de perfil de usuario que permite:

- Editar información personal
- Configurar preferencias de privacidad
- Gestionar suscripciones o configuraciones de recomendaciones
- Vista de estadísticas de uso (libros leídos, series vistas, etc.)
- Opción de cambiar contraseña y configuraciones de cuenta

6.1 Nuevas Funcionalidades Implementadas

6.1.1 Historial de Búsquedas

La funcionalidad para usuarios se implementó siguiendo una arquitectura multicapa centrada en el patrón MVC. El componente principal SearchHistoryController maneja las peticiones de los usuarios y coordina con SearchHistoryService para procesar la lógica de negocio. Los datos se persisten mediante SearchHistoryRepository, que interactúa con la base de datos utilizando JPA. La transferencia de datos se realiza a través de SearchHistoryDTO, asegurando una separación limpia entre las capas.

```java
public List<SearchHistory> getUserHistoryWithFilters(User user, String search, String type, String date
    try {
        if (search == null && type == null && date == null) {
            return getUserHistory(user);
        }

        LocalDateTime dateFilter = date != null ? LocalDate.parse(date).atStartOfDay() : null;
        return searchHistoryRepository.findByFilters(user, search, type, dateFilter);
    } catch (Exception e) {
        logger.error(format:"Error filtering user history: {}", e.getMessage());
        throw new RuntimeException(message:"Error filtering search history", e);
    }
}

public List<SearchHistory> getAllHistoryWithFilters(String search, String type, String date) {
    try {
        if (search == null && type == null && date == null) {
            return getAllHistory();
        }

        LocalDateTime dateFilter = date != null ? LocalDate.parse(date).atStartOfDay() : null;
        return searchHistoryRepository.findByFiltersAdmin(search, type, dateFilter);
    } catch (Exception e) {
        logger.error(format:"Error filtering all history: {}", e.getMessage());
        throw new RuntimeException(message:"Error filtering search history", e);
    }
}
```

Figura 7 SearchHistoryService.java.

La funcionalidad para administradores se integró en el sistema de seguridad existente, aprovechando WebSecurityConfig para el control de acceso basado en roles. Los administradores tienen acceso a vistas especializadas que permiten:

- Visualización completa del historial de todos los usuarios
- Análisis de patrones de búsqueda
- Gestión y mantenimiento de registros históricos

6.1.2 Agregar a Favoritos

El sistema de favoritos se implementó como una funcionalidad transversal que permite a los usuarios guardar tanto libros como contenido multimedia. La implementación utiliza FavoriteController como punto de entrada principal, que coordina con FavoriteService para manejar la lógica de negocio. La persistencia se maneja a través de FavoriteRepository, permitiendo operaciones CRUD eficientes.

La interactividad del lado del cliente se logró mediante:

- Integración de JavaScript para actualizaciones dinámicas
- Manejo asíncrono de favoritos sin recarga de página
- Feedback visual inmediato al usuario

```java
public List<Book> getUserFavoriteBooks(User user) {
    // Implement the logic to get the user's favorite books
    List<Favorite> favorites = favoriteRepository.findBookFavoritesByUser(user);
    List<Book> favoriteBooks = new ArrayList<>();
    for (Favorite favorite : favorites) {
        Book book = new Book();
        book.setTitle(favorite.getLibro());
        book.setAuthor(favorite.getAutor());
        book.setDescription(favorite.getDescripcion());
        book.setCoverUrl(favorite.getImagenUrl());
        book.setIsbn(favorite.getContenidoId());
        favoriteBooks.add(book);
    }
    return favoriteBooks;
}
public List<Show> getUserFavoriteShows(User user) {
    // Implement the logic to get the user's favorite shows
    List<Favorite> favorites = favoriteRepository.findSeriesFavoritesByUser(user);
    List<Show> favoriteShows = new ArrayList<>();
    for (Favorite favorite : favorites) {
        Show show = new Show();
        show.setName(favorite.getLibro());
        show.setSummary(favorite.getDescripcion());
        show.setId(Long.parseLong(favorite.getContenidoId()));
        favoriteShows.add(show);
    }
    return favoriteShows;
}
```

Figura 8 FavoriteService. java.

6.1.3 Sistema de Recomendaciones

El motor de recomendaciones se desarrolló utilizando un enfoque basado en criterios múltiples, implementado principalmente en RecommendationService. El sistema analiza el historial de usuario, preferencias explícitas y patrones de consumo para generar recomendaciones personalizadas. La integración con APIs externas enriquece las recomendaciones con datos actualizados de OpenLibrary y TVMaze.

Aspectos destacados de la implementación incluyen:

- Algoritmos de filtrado basados en preferencias de usuario
- Sistema de puntuación dinámico
- Actualización periódica de recomendaciones

```java
private RecommendationCriteria buildCriteriaFromUserActivity(
        List<SearchHistory> searchHistory,
        List<Favorite> favorites) {
    RecommendationCriteria criteria = new RecommendationCriteria();

    // Analizar historial de búsquedas
    for (SearchHistory search : searchHistory) {
        if (!search.getLibro().isEmpty()) {
            criteria.incrementGenreWeight(genre:"book", weight:0.3);
        }
        if (!search.getSerie().isEmpty() || !search.getPelicula().isEmpty()) {
            criteria.incrementGenreWeight(genre:"show", weight:0.3);
        }
    }

    // Analizar favoritos
    for (Favorite favorite : favorites) {
        if (!favorite.getLibro().isEmpty()) {
            criteria.incrementGenreWeight(genre:"book", weight:0.5);
        }
        if (!favorite.getSerie().isEmpty() || !favorite.getPelicula().isEmpty()) {
            criteria.incrementGenreWeight(genre:"show", weight:0.5);
        }
    }

    return criteria;
}
```

Figura 9 RecommendationService.java

6.1.4 Consumo de APIs Públicas

La integración con APIs públicas se implementó mediante servicios especializados que encapsulan la lógica de comunicación con cada proveedor. OpenLibraryService y TVMazeService manejan las respectivas integraciones, implementando cache para optimizar el rendimiento y manejo de errores robusto. Los controladores específicos de dominio (BookController, MoviesController, TVMazeController) utilizan estos servicios para proporcionar funcionalidades específicas.

6.1.5 Persistencia de Datos

La implementación de la persistencia de datos se realizó utilizando Spring Data JPA con MySQL como sistema gestor de base de datos, según se evidencia en las dependencias del pom.xml. El modelo de datos para las recomendaciones se define en la entidad Recommendation.java, que se mapea directamente a la tabla de recomendaciones en la base de datos. Los métodos de almacenamiento se implementan a través del RecommendationRepository.java, que extiende JpaRepository, proporcionando operaciones CRUD básicas y consultas personalizadas mediante query methods. La verificación de la persistencia se gestiona a través de transacciones declarativas de Spring, asegurando la integridad de los datos mediante el uso de la anotación @Transactional en el servicio RecommendationService.java. El sistema implementa un esquema de validación en dos niveles: validaciones a nivel de entidad usando anotaciones de Jakarta Validation (anteriormente Java Bean Validation) como @NotNull, @Size, y validaciones a nivel de servicio para reglas de negocio más complejas. Para optimizar el rendimiento, se implementó un sistema de caché utilizando las anotaciones @Cacheable de Spring, reduciendo así la carga en la base de datos para consultas frecuentes.

Métodos de almacenamiento:

- Inserción atómica de nuevas recomendaciones
- Actualización batch para múltiples registros
- Borrado lógico para mantener historial

Verificación de la persistencia:

- Logs de auditoría para cambios críticos
- Validación de integridad referencial

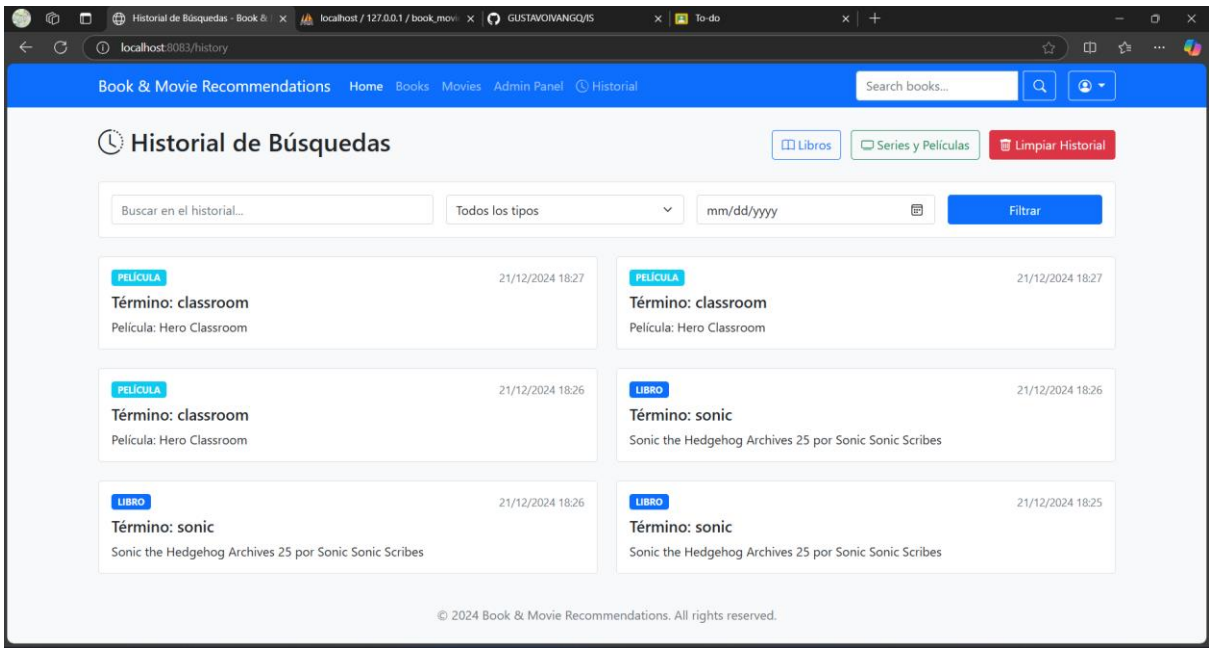
- Monitoreo de transacciones exitosas

6.2 Pruebas y Verificación

6.2.1 Pruebas de API REST

6.2.2 Historial de Búsquedas

- Funcionalidad para usuarios



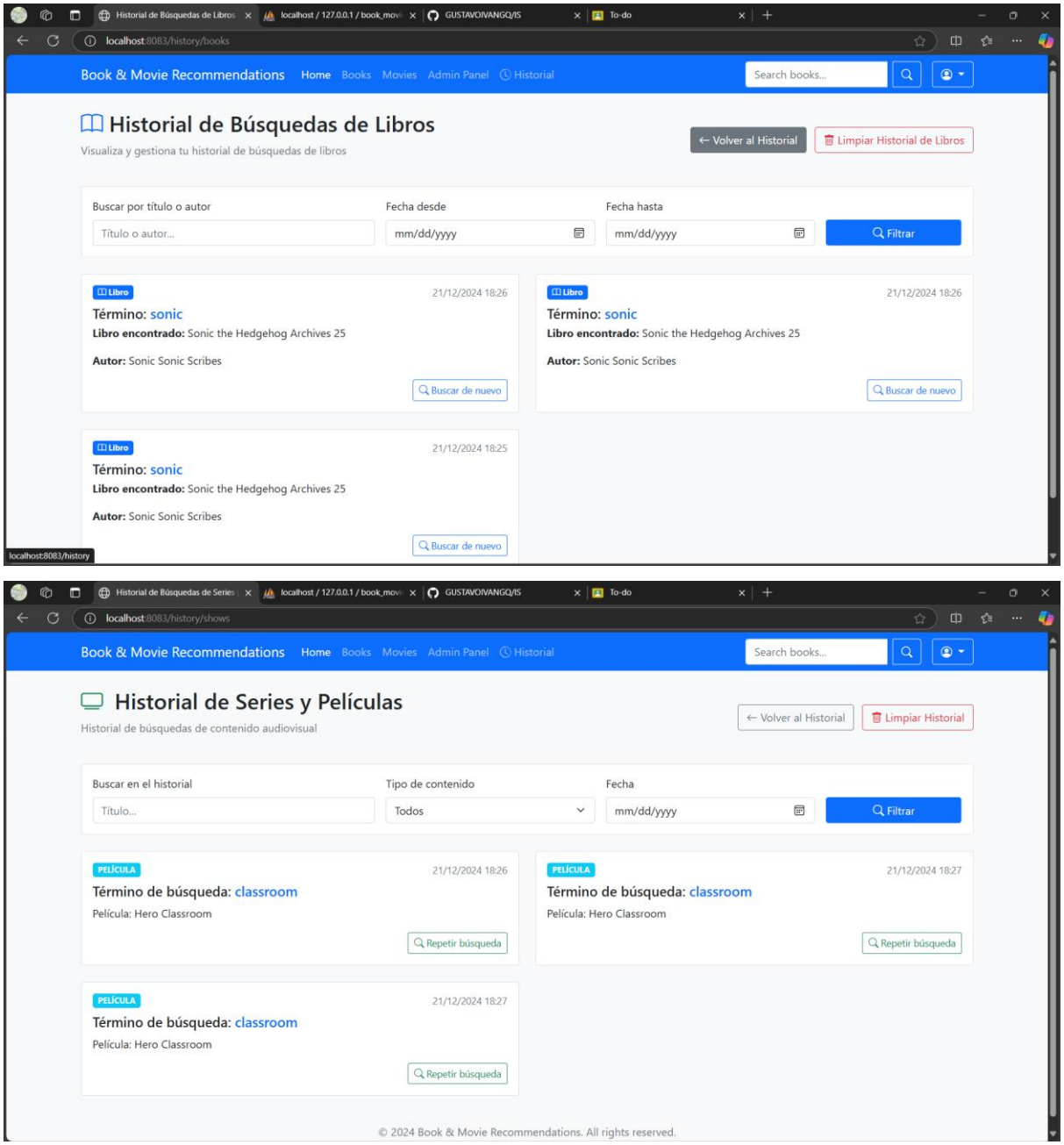


Figura 10 Funcionalidad para usuarios.

- Funcionalidad para administradores

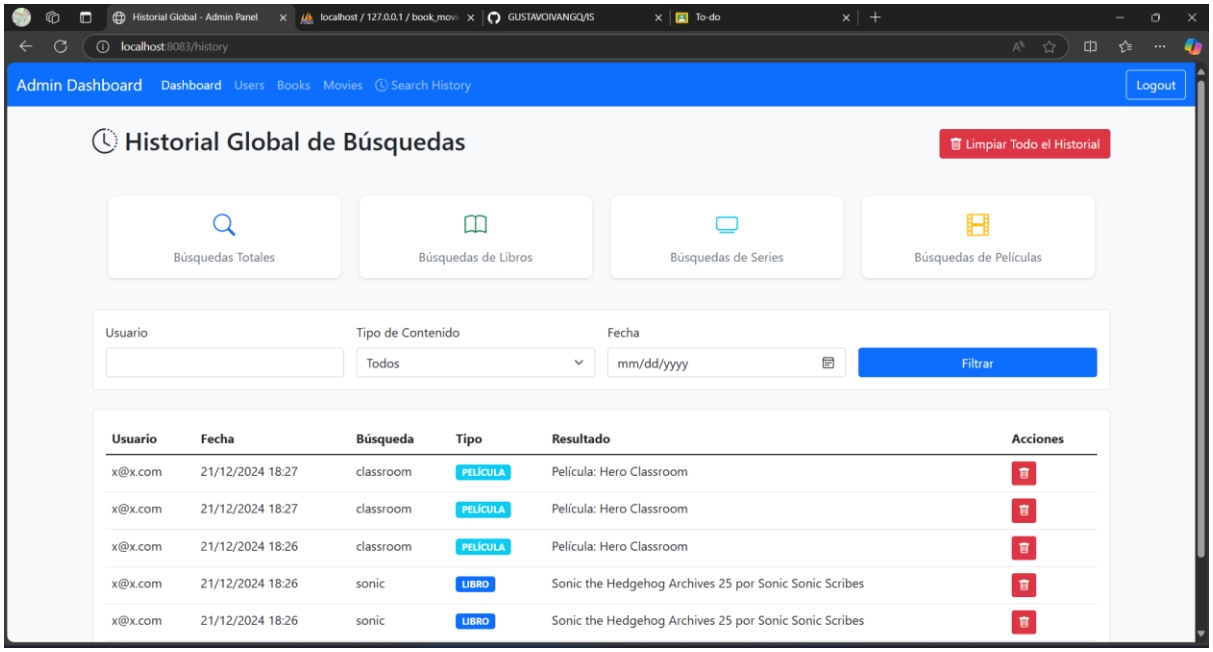


Figura 11 Funcionalidad para administradores.

6.2.3 Agregar a Favoritos

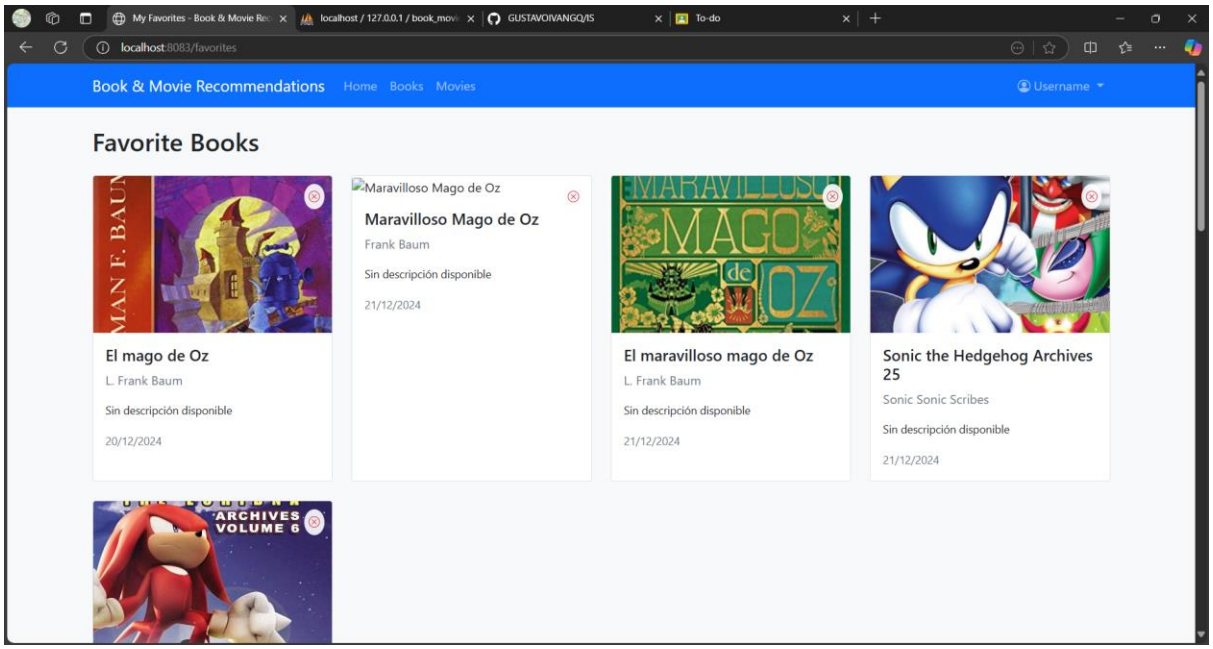


Figura 12 Funcionalidad para Favoritos.

6.2.4 Sistema de Recomendaciones

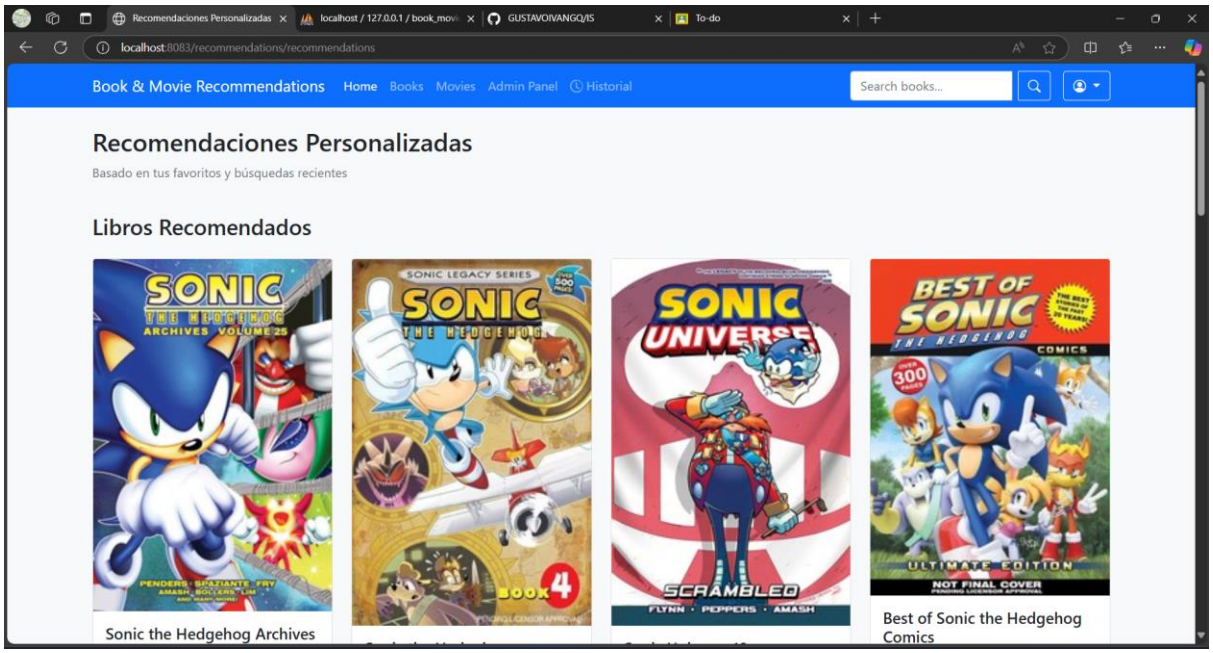


Figura 13 Sistema de Recomendaciones.

8 Conclusiones

Durante el desarrollo de la Práctica 4 de Modelo de Diseño y Prototipos de Interfaces, se experimentaron diversos desafíos y se lograron avances significativos en el proceso de diseño y desarrollo de software. La implementación de nuevas funcionalidades representó una oportunidad para profundizar la comprensión de los principios de ingeniería de software y la importancia de un diseño estructurado y centrado en el usuario.

El proceso de documentación de requerimientos fue fundamental para establecer una base sólida para el proyecto. Mediante el uso de diagramas UML y la descripción detallada de casos de uso, se logró una representación clara de la arquitectura del sistema y las interacciones entre diferentes componentes. Esta etapa de modelado permitió identificar posibles mejoras y optimizaciones en la estructura del software antes de su implementación completa.

La creación de prototipos de interfaces gráficas fue un componente crucial que ayudó a visualizar la experiencia del usuario y garantizar una interacción intuitiva. Al diseñar estos prototipos, se pudo reflexionar sobre aspectos de usabilidad y diseño centrado en el usuario, lo que contribuyó a una comprensión más profunda de las necesidades de los usuarios finales del sistema.

La implementación de funcionalidades adicionales, como el historial de búsquedas, la función de agregar favoritos y el sistema de recomendaciones, demostró la complejidad de integrar múltiples servicios y APIs. Se adquirió experiencia práctica en la gestión de consumo de APIs públicas, persistencia de datos y desarrollo de funcionalidades que mejoran la experiencia del usuario.

Los desafíos técnicos encontrados durante la práctica incluyeron la correcta integración de diferentes componentes, el manejo de datos entre diferentes servicios y la implementación de un sistema de recomendaciones coherente. Cada obstáculo superado representó una oportunidad de aprendizaje y mejora en las habilidades de desarrollo de software.

Finalmente, esta práctica reforzó la importancia de un enfoque metódico en el desarrollo de software, destacando la necesidad de una documentación detallada, un diseño y una implementación que priorice tanto los aspectos técnicos como la experiencia del usuario. El proceso ha proporcionado una visión de las etapas de diseño y prototipado en un proyecto de software.

9 BIBLIOGRAFÍA APA

- Pressman RS. INGENIERIA DE SOFTWARE.; 2010.
- Sommerville I, Velázquez SF. Ingeniería de software.; 2011.
- Sommerville, I. (2015). Ingeniería de Software. Pearson Educación.
- Booch, G., Rumbaugh, J., & Jacobson, I. (2005). The Unified Modeling Language User Guide. Addison-Wesley.