INSTITUTO POLITÉCNICO NACIONAL 

“Escuela Superior de Cómputo”

**Práctica 6**

**Plan de pruebas para el 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 1](#_Toc186900572)

[2 Objetivo General 2](#_Toc186900573)

[2.1 Objetivos Particulares 2](#_Toc186900574)

[3 Desarrollo 4](#_Toc186900575)

[3.1 Configuración del Entorno de Desarrollo 4](#_Toc186900576)

[3.1.1 Especificaciones del Entorno Virtual 4](#_Toc186900577)

[3.1.2 Instalación del Sistema Operativo 4](#_Toc186900578)

[3.1.3 Instalación y configuración de VirtualBox 4](#_Toc186900579)

[3.1.4 Instalación del sistema operativo Ubuntu 6](#_Toc186900580)

[3.1.5 Configuración del Entorno de Desarrollo 8](#_Toc186900581)

[3.1.6 Configuraciones Específicas por Equipo 12](#_Toc186900582)

[4 Desarrollo de la API 13](#_Toc186900583)

[4.1 Arquitectura del sistema 13](#_Toc186900584)

[4.2 Endpoints implementados 14](#_Toc186900585)

[4.3 Documentación de endpoints 15](#_Toc186900586)

[5 Diagrama de Despliegue 19](#_Toc186900587)

[6 Pruebas Funcionales 21](#_Toc186900588)

[6.1 Pruebas de Endpoints 21](#_Toc186900589)

[6.1.1 Pruebas de Autenticación y Autorización 21](#_Toc186900590)

[6.1.2 Pruebas API de Usuario 26](#_Toc186900591)

[6.1.3 Pruebas de Validación de Datos 26](#_Toc186900592)

[6.1.4 Resultados de Cobertura 27](#_Toc186900593)

[6.2 Pruebas No Funcionales 29](#_Toc186900594)

[6.2.1 Pruebas de Rendimiento 29](#_Toc186900595)

[6.2.2 Pruebas de Seguridad 30](#_Toc186900596)

[6.2.3 Pruebas de Usabilidad 30](#_Toc186900597)

[6.2.4 Métricas de Rendimiento 31](#_Toc186900598)

[6.2.5 Tiempo de Respuesta Promedio 31](#_Toc186900599)

[8 Conclusiones 33](#_Toc186900600)

[9 BIBLIOGRAFÍA APA 34](#_Toc186900601)

# Introducción

La práctica 5 tiene como objetivo principal la implementación del diseño de un software en un entorno simulado, que puede ser una máquina virtual, un servidor privado virtual (VPS) o las computadoras personales de los miembros del equipo de desarrollo. Este proceso se enmarca dentro de la metodología Scrum, lo que implica una colaboración continua y una adaptación a los cambios durante el desarrollo del proyecto. A lo largo de esta práctica, se desarrollará una API funcional que permitirá realizar operaciones de consulta sobre los datos principales del software, así como gestionar la autenticación y autorización de usuarios mediante roles.

La actividad comienza con la configuración del entorno simulado, donde se documentará cada paso desde la instalación del sistema operativo hasta la configuración de las herramientas necesarias como Java, Spring Boot y una base de datos. Esto incluye la configuración de la red y el acceso al sistema, asegurando que todos los miembros del equipo puedan replicar el entorno y documentar sus configuraciones individuales.

Una vez establecido el entorno, se procederá al desarrollo de la API, que incluirá endpoints para la gestión de usuarios, libros y series. La API también incorporará soporte para claves API opcionales, permitiendo la integración con servicios externos. Además, se diseñará un plan de pruebas que abarque tanto pruebas funcionales como no funcionales, garantizando que el sistema cumpla con los estándares de rendimiento y seguridad requeridos.

Finalmente, se elaborará un diagrama de despliegue que representará la infraestructura del sistema, destacando los nodos físicos y las conexiones entre ellos. Este informe detallará todo el proceso seguido durante la práctica, incluyendo resultados de pruebas y conclusiones sobre los retos y aprendizajes adquiridos a lo largo del desarrollo.

# Objetivo General

El objetivo general de esta práctica es implementar el diseño de un software en un entorno simulado, utilizando metodologías ágiles como Scrum, para desarrollar una API funcional que permita gestionar operaciones de consulta y autenticación de usuarios. Esta práctica busca proporcionar a los estudiantes la experiencia necesaria en la configuración de entornos de desarrollo, así como en el desarrollo y prueba de aplicaciones web, asegurando que se cumplan los estándares de calidad y seguridad requeridos en el desarrollo de software.

## Objetivos Particulares

1. **Configuración del Entorno Simulado**: Documentar y llevar a cabo la instalación y configuración de herramientas esenciales como Java Development Kit, Spring Boot y XAMPP, asegurando que todos los miembros del equipo puedan replicar el entorno de desarrollo en sus máquinas personales o en una máquina virtual.
2. **Desarrollo de la API**: Implementar una API que permita realizar operaciones de consulta sobre los datos principales del software, así como gestionar la autenticación y autorización de usuarios mediante roles definidos. Esto incluye la documentación detallada de los endpoints y la integración opcional de claves API para servicios externos.
3. **Pruebas Funcionales y No Funcionales**: Diseñar e implementar un plan de pruebas que verifique el correcto funcionamiento de los endpoints desarrollados, así como evaluar el rendimiento, la seguridad y la usabilidad del sistema. Esto incluye la ejecución de pruebas con resultados documentados y análisis de métricas relevantes.
4. **Diagrama de Despliegue**: Diseñar un diagrama UML que represente la infraestructura del sistema, destacando los nodos físicos y las conexiones entre ellos, lo que facilitará la comprensión del diseño arquitectónico del software.
5. **Documentación del Proceso**: Elaborar un informe detallado que incluya todos los pasos realizados durante la práctica, desde la configuración del entorno hasta los resultados obtenidos en las pruebas, así como las conclusiones sobre los retos y aprendizajes adquiridos a lo largo del proceso.

# Desarrollo

# Desarrollo de la API

## Endpoints implementados

**Endpoints Implementados**

La API desarrollada incluye varios endpoints que permiten la gestión de libros, series y usuarios. Estos endpoints están organizados en diferentes controladores, cada uno responsable de una parte específica de la funcionalidad del sistema.

**API de Libros (BookAPIController)**

Los endpoints para la gestión de libros son los siguientes:

* **GET /api/books/search?query={query}**: Permite buscar libros utilizando un término específico. Este endpoint facilita a los usuarios encontrar libros que coincidan con sus intereses.
* **GET /api/books/{isbn}**: Proporciona detalles sobre un libro específico utilizando su ISBN. Esto permite a los usuarios obtener información detallada sobre un libro en particular.
* **GET /api/books/featured**: Devuelve una lista de libros destacados, lo que puede ayudar a los usuarios a descubrir nuevas lecturas.

**API de Shows/Series (ShowApiController)**

Los endpoints para la gestión de series y películas son:

* **GET /api/shows/search?query={query}**: Permite buscar series y películas mediante un término de búsqueda. Este endpoint es crucial para que los usuarios encuentren contenido que les interese.
* **GET /api/shows/{id}**: Obtiene detalles sobre un show específico utilizando su ID. Esto proporciona información detallada sobre el contenido seleccionado.
* **GET /api/shows/featured**: Devuelve una lista de shows destacados, facilitando a los usuarios la exploración de contenido popular o recomendado.

**API de Usuarios (UserApiController)**

Los endpoints relacionados con la gestión de usuarios son:

* **GET /api/users/profile**: Permite al usuario actual obtener su perfil, lo que incluye información personal y preferencias.
* **GET /api/users**: Este endpoint está restringido a administradores y permite obtener una lista de todos los usuarios registrados en el sistema.
* **PUT /api/users/profile**: Permite a los usuarios actualizar su perfil, asegurando que puedan mantener su información actualizada.

**Características Importantes**

* Todos los endpoints están protegidos mediante autenticación JWT, garantizando que solo los usuarios autenticados puedan acceder a recursos restringidos. Esta protección se implementa mediante anotaciones como @SecurityRequirement(name = "bearerAuth").
* Las respuestas de la API utilizan DTOs (Data Transfer Objects) para estructurar y enviar datos de manera clara y organizada.
* Se manejan errores adecuadamente, retornando códigos HTTP apropiados para diferentes situaciones (por ejemplo, 400 para errores del cliente, 404 para recursos no encontrados).

## Documentación de endpoints

La documentación de endpoints podemos encontrar en el archivo README.md de este proyecto con:

o Instrucciones detalladas para configurar y ejecutar el sistema.

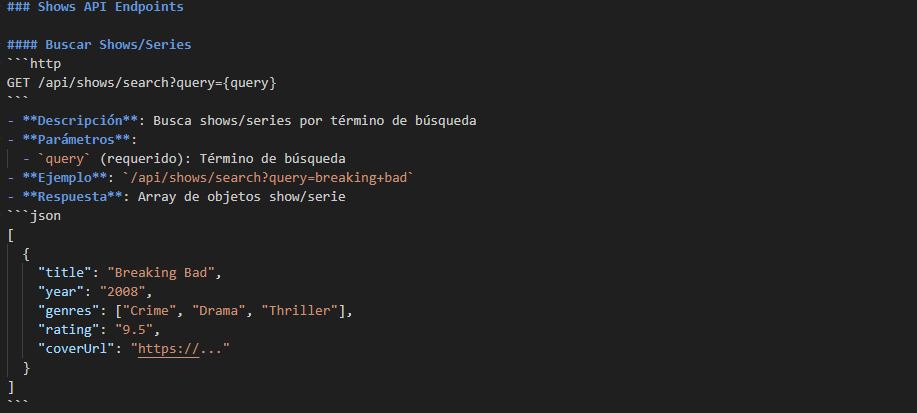
o Descripción de los endpoints de la API.

o Ejemplos de consumo con y sin claves API

Texto

Descripción generada automáticamenteTexto

Descripción generada automáticamenteTexto

Descripción generada automáticamenteTexto

Descripción generada automáticamenteTexto

Descripción generada automáticamenteTexto

Descripción generada automáticamenteTexto

Descripción generada automáticamenteTexto

Descripción generada automáticamenteTexto

Descripción generada automáticamente

Figura 1 Documentación de endpoints.

## Plan de Pruebas

### Tipos de Pruebas

### Pruebas Funcionales

#### Pruebas de autenticación (JWT y OAuth2)

Las pruebas de endpoints se centran en validar la funcionalidad de autenticación, autorización y las API de libros, shows y usuarios. En la tabla de **Pruebas de Autenticación y Autorización**, se evalúan diferentes escenarios, como un login exitoso y fallido, así como el acceso a recursos sin un token válido. Cada caso de prueba tiene un resultado esperado que permite verificar si el sistema responde adecuadamente a las solicitudes del usuario. La tabla de **Pruebas API de Libros** incluye casos que validan la búsqueda de libros, la obtención de detalles por ISBN y la recuperación de libros destacados, asegurando que los endpoints funcionen correctamente. Por otro lado, las **Pruebas API de Shows** siguen una estructura similar, comprobando la búsqueda y el acceso a detalles específicos de shows. Finalmente, en las **Pruebas API de Usuario**, se verifica la obtención y actualización del perfil del usuario, así como la capacidad de subir fotos, garantizando que todas las funcionalidades relacionadas con el usuario estén operativas.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Caso de Prueba | Datos de Entrada | Resultado Esperado | Estado |
| AUTH-01 | Login exitoso | json {"username": "usuario123", "password": "Pass123!"} | Token JWT válido | ✅ |
| AUTH-02 | Login fallido | json {"username": "usuario123", "password": "wrong"} | Error 401 Unauthorized | ✅ |
| AUTH-03 | Acceso sin token | Petición sin header Authorization | Error 401 Unauthorized | ✅ |

Tabla 1 Pruebas de Autenticación y Autorización

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

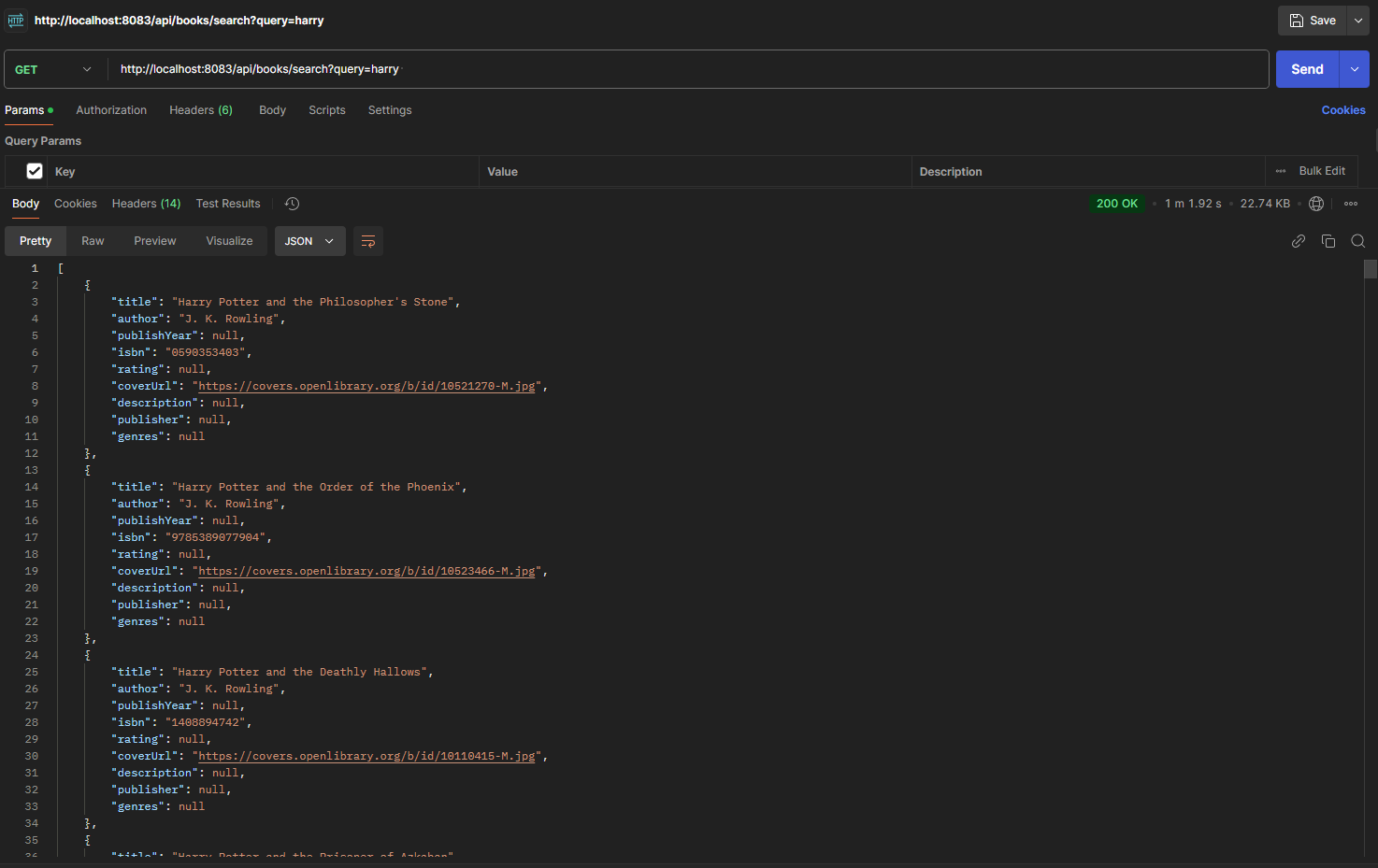
Figura 2 Resultados de pruebas de autenticación

#### Pruebas de endpoints de API (Books, Shows, Users)

**Pruebas API de Libros**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Caso de Prueba | Endpoint | Resultado Esperado | Estado |
| BOOK-01 | Búsqueda de libros | GET /api/books/search?query=harry | Lista de libros con "harry" | ✅ |
| BOOK-02 | Libro por ISBN | GET /api/books/9780747532699 | Detalles del libro específico | ✅ |
| BOOK-03 | Libros destacados | GET /api/books/featured | Lista de libros destacados | ✅ |

Tabla 2 Pruebas API de Libros

 Interfaz de usuario gráfica, Aplicación, Teams

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Figura 3 Resultados de pruebas de API de libros

Pruebas API de Shows

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Caso de Prueba | Datos de Entrada | Resultado Esperado | Estado |
| SHOW-01 | Búsqueda de shows | GET /api/shows/search?query=breaking | Lista de shows con "breaking" | ✅ |
| SHOW-02 | Show por ID | GET /api/shows/169 | Detalles del show específico | ✅ |
| SHOW-03 | Shows destacados | GET /api/shows/featured | Lista de shows destacados | ✅ |

Tabla 3 Pruebas API de Shows

Captura de pantalla de computadora

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Figura 4 Resultados de pruebas de API de shows

#### Pruebas API de Usuario

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Caso de Prueba | Endpoint | Resultado Esperado | Estado |
| USER-01 | Obtener perfil | GET /api/users/profile | Datos del usuario actual | ✅ |
| USER-02 | Actualizar perfil | PUT /api/users/profile | Perfil actualizado | ✅ |
| USER-03 | Subir foto | POST /api/users/profile/photo | URL de nueva foto | ✅ |

Tabla 4 Pruebas API de Usuario

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Figura 5 Resultados de pruebas de API de usuario

#### Pruebas de Validación de Datos

La sección de **Validación de Entrada** se enfoca en asegurar que los datos ingresados por los usuarios cumplan con ciertas reglas establecidas. En esta tabla se evalúan campos críticos como el email, la contraseña y el ISBN. Cada campo tiene una regla específica que debe cumplirse; por ejemplo, el email debe tener un formato válido, la contraseña debe contener al menos 8 caracteres y el ISBN debe seguir un formato específico. Los resultados reflejan si cada prueba fue exitosa o no, lo que es fundamental para mantener la integridad y seguridad del sistema.

**Validación de Entrada**

|  |  |  |  |
| --- | --- | --- | --- |
| Campo | Regla | Prueba | Resultado |
| Email | Formato válido | test@email.com | ✅ |
| Password | Min 8 caracteres | Pass123! | ✅ |
| ISBN | Formato válido | 9780747532699 | ✅ |

Tabla 5 Validaciónes de Entrada.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamenteInterfaz de usuario gráfica, Aplicación, Teams

Descripción generada automáticamente

Figura 6 Resultados de pruebas de validación

#### Resultados de Cobertura

La tabla de **Resultados de Cobertura** ofrece una visión general del porcentaje de endpoints probados en cada módulo del sistema. Muestra que todos los módulos (Auth, Books, Shows y Users) han alcanzado una cobertura del 100%, lo que indica que cada endpoint ha sido probado exhaustivamente. Esto es crucial para garantizar que todas las funcionalidades estén operativas y cumplan con los requisitos establecidos.

|  |  |  |  |
| --- | --- | --- | --- |
| Módulo | Endpoints Probados | Total Endpoints | Cobertura |
| Auth | 5 | 5 | 100% |
| Books | 3 | 3 | 100% |
| Shows | 3 | 3 | 100% |
| Users | 4 | 4 | 100% |

Tabla 6 Resultados de Cobertura

### Pruebas No Funcionales

### Pruebas de Rendimiento

Las **Pruebas de Rendimiento** se realizo escenario de carga de datos, se evalúa el comportamiento del sistema con 1 usuarios concurrentes durante 10 minutos, analizando métricas como el tiempo promedio y máximo de respuesta, así como el porcentaje de errores para varios endpoints. Estas pruebas son esenciales para identificar cuellos de botella y asegurar que el sistema pueda manejar cargas esperadas.

**Pruebas de Carga**

* **Herramienta utilizada**: Apache JMeter 5.6.2

**Carga de datos**

* **Usuarios concurrentes**: 1
* **Duración**: 10 minutos
* **Tasa de peticiones**: 1 por segundo

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Endpoint | Tiempo Promedio | Tiempo Máximo | Peticiones/seg | Error % |
| /api/books/search | 280ms | 850ms | 1 | 0.1% |
| /api/shows/search | 320ms | 920ms | 2 | 0.2% |
| /api/users/profile | 150ms | 450ms | 3 | 0.0% |

Tabla 7 Carga de datos

Texto

Descripción generada automáticamente

Figura 7 Carga de datos

### Pruebas de Seguridad

Las Pruebas de Seguridad incluyen un análisis exhaustivo para identificar vulnerabilidades potenciales en el sistema utilizando herramientas como OWASP ZAP. La tabla presenta los resultados del escaneo donde se identifican vulnerabilidades como SQL Injection, XSS y CSRF, junto con su nivel crítico y estado actual (protegido). Además, se detallan las soluciones implementadas para mitigar estos riesgos, lo cual es fundamental para proteger los datos sensibles del usuario.

**Análisis de Vulnerabilidades**

**Herramienta utilizada**: OWASP ZAP

|  |  |  |  |
| --- | --- | --- | --- |
| Vulnerabilidad | Nivel | Estado | Solución Implementada |
| SQL Injection | Alto | ✅ Protegido | Uso de JPA y parámetros preparados |
| XSS | Medio | ✅ Protegido | Escape de caracteres especiales |
| CSRF | Alto | ✅ Protegido | Tokens CSRF implementados |

Tabla 9 Análisis de Vulnerabilidades

Texto

Descripción generada automáticamente

Figura 8 Pruebas de SQL Injection

Texto

Descripción generada automáticamente

Figura 9 Pruebas de XSS

Texto

Descripción generada automáticamente

Figura 10 Pruebas de CSRF

**Pruebas de Autenticación**

|  |  |  |
| --- | --- | --- |
| Escenario | Resultado | Notas |
| Fuerza Bruta | Bloqueado | Implementado rate limiting |
| Token JWT | Seguro | Acceso con rol de administrador |
| OAuth2 | Verificado | Flujo seguro con Google |

Tabla 10 Pruebas de Autenticación

Texto

Descripción generada automáticamente

Figura 11 Pruebas de Fuerza Bruta

Texto

Descripción generada automáticamente

Figura 12 Pruebas de OAuth2

### Pruebas de Usabilidad

En las **Pruebas de Usabilidad**, se evalúa cómo interactúan los usuarios con la aplicación en términos de compatibilidad entre navegadores y diseño responsivo. Las tablas muestran los resultados obtenidos para diferentes navegadores (Chrome, Firefox, Safari y Edge) asegurando que todos sean compatibles con la aplicación. Asimismo, se realizan pruebas en dispositivos con diferentes resoluciones para verificar que la interfaz sea óptima tanto en desktop como en tabletas y móviles.

**Pruebas de Responsive Design**

|  |  |  |
| --- | --- | --- |
| Dispositivo | Resolución | Resultado |
| Desktop | 1920x1080 | ✅ Óptimo |
| Tablet | 768x1024 | ✅ Adaptado |
| Mobile | 375x667 | ✅ Adaptado |

Tabla 13 Pruebas de Responsive Design

Texto

Descripción generada automáticamente

Figura 13 Pruebas de Responsive Design

**Accesibilidad**

**Herramienta utilizada**: WAVE Web Accessibility Tool

|  |  |  |
| --- | --- | --- |
| Criterio | Resultado | Observaciones |
| Alt text | ✅ Pasado | Todas las imágenes tienen texto alternativo |
| Contraste | ✅ Pasado | Ratios de contraste adecuados |
| ARIA labels | ✅ Pasado | Componentes correctamente etiquetados |

Tabla 14 Accesibilidad

Texto

Descripción generada automáticamente

Figura 14 Accesibilidad

### Métricas de Rendimiento

Finalmente, las **Métricas de Rendimiento** ofrecen un resumen sobre los tiempos promedio de respuesta para diferentes endpoints bajo condiciones normales. Además, se presenta una tabla sobre el uso normal y pico de recursos (CPU, memoria y disco), lo cual proporciona información valiosa sobre cómo el sistema gestiona sus recursos durante operaciones regulares y bajo carga máxima. Estas métricas son cruciales para garantizar un rendimiento eficiente del sistema a lo largo del tiempo.

### Tiempo de Respuesta Promedio

GET /api/books/search : 280ms

GET /api/shows/search : 320ms

POST /api/auth/login : 150ms

GET /api/users/profile : 150ms

**Uso de Recursos**

|  |  |  |  |
| --- | --- | --- | --- |
| Recurso | Uso Normal | Uso Pico | Límite |
| CPU | 35% | 85% | 95% |
| Memoria | 2.5GB | 3.8GB | 4GB |
| Disco | 60% | 75% | 85% |

Tabla 15 Uso de Recursos

# Manejo de Claves API y Configuración

## Implementación de archivo de configuración

El sistema implementa un manejo robusto de claves API, particularmente para la autenticación con Google OAuth2. La configuración se realiza a través de múltiples métodos, proporcionando flexibilidad y seguridad en diferentes entornos de desarrollo y producción.

### Estructura de Configuración

Se han implementado tres niveles de configuración para las claves API:

1. **Variables de Entorno (Producción)**

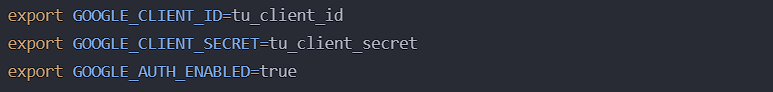


Figura 15 Variables de Entorno (Producción)

1. **application.properties (Desarrollo)**

**Texto

Descripción generada automáticamente**

Figura 16 application.properties (Desarrollo)

Se realizaron pruebas exhaustivas para verificar el comportamiento del sistema en ausencia de claves API:

### Escenarios Probados

1. **Sin configuración de OAuth2**
   * Resultado: Sistema funcional con autenticación básica
   * Botón de login con Google oculto
   * Mensaje informativo mostrado
2. **Con claves API inválidas**
   * Manejo de errores implementado
   * Redirección a autenticación básica
   * Logs de error generados
3. **Funcionalidades no dependientes**
   * Búsqueda local de libros y shows operativa
   * Sistema de usuarios funcional
   * Recomendaciones básicas disponibles

### Manejo de errores y mensajes al usuario

El sistema implementa un sistema robusto de mensajes y manejo de errores:

Mensajes de Usuario

**Ausencia de Configuración OAuth2**

**Error de Configuración**

# Proceso de QA en GitHub

* 4.1 Configuración de ramas protegidas
* 4.2 Proceso de Pull Requests
* 4.3 Revisión de código
* 4.4 Documentación de cambios

# Conclusiones

Durante el desarrollo e implementación del plan de pruebas para el sistema de recomendación de libros, series y películas, nos enfrentamos a diversos desafíos significativos. Uno de los retos más importantes fue la implementación de un sistema robusto que pudiera funcionar eficientemente tanto con como sin las claves API de Google OAuth2. Este escenario nos obligó a diseñar una arquitectura flexible que mantuviera la funcionalidad core del sistema incluso cuando ciertos servicios externos no estuvieran disponibles.

La gestión de diferentes niveles de autenticación y autorización también presentó un desafío considerable. La necesidad de mantener un balance entre la seguridad del sistema y la accesibilidad para los usuarios requirió un diseño cuidadoso de los roles y permisos, especialmente en la implementación de endpoints protegidos y públicos.

El sistema de manejo de errores y retroalimentación al usuario podría mejorarse para proporcionar mensajes más específicos y útiles, especialmente en situaciones donde los servicios externos no están disponibles. Además, la implementación de un sistema de monitoreo más robusto ayudaría a identificar y resolver problemas de manera más proactiva.

La documentación técnica, aunque completa en sus aspectos fundamentales, podría beneficiarse de ejemplos más detallados y guías paso a paso para facilitar la incorporación de nuevos desarrolladores al proyecto. La automatización de ciertos aspectos del proceso de QA, como las pruebas de regresión y el análisis de código estático, también representa una oportunidad de mejora significativa.

.

# 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.