INSTITUTO POLITÉCNICO NACIONAL

"Escuela Superior de Cómputo"

PRÁCTICA 5

Implementación del Diseño en un Entorno Simulado

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 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.

2 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.

2.1 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.

3 Desarrollo

3.1 Configuración del Entorno de Desarrollo

3.1.1 Especificaciones del Entorno Virtual

- **Virtualización**: Oracle VirtualBox 7.0
- **Sistema Operativo**: Ubuntu 22.04 LTS
- **Recursos Asignados**:
    - RAM: 4GB
    - Almacenamiento: 50GB
    - Procesadores: 2 cores

3.1.2 Instalación del Sistema Operativo

3.1.3 Instalación y configuración de VirtualBox

El primer paso para obtener VirtualBox es la instalación, encontraremos el programa en la página oficial.

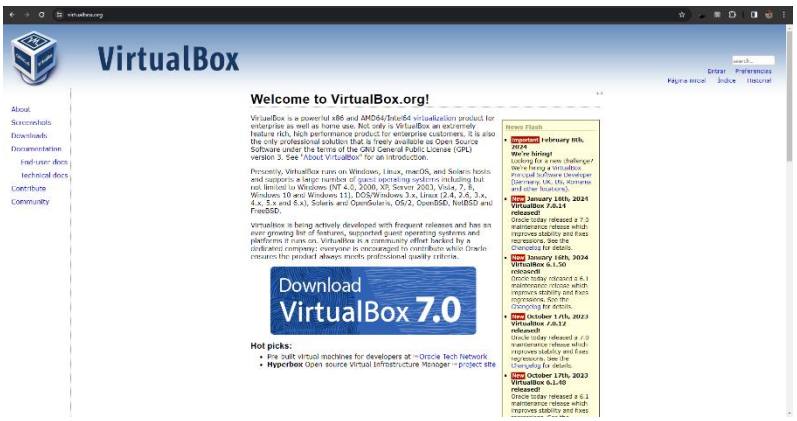


Figura 1 https://www.virtualbox.org/

Al momento en el que termina la descarga buscaremos la carpeta en nuestro dispositivo y la ejecutaremos. Una vez ejecutado elegimos la ruta de instalación, leemos los términos y condiciones y continuamos con la instalación.

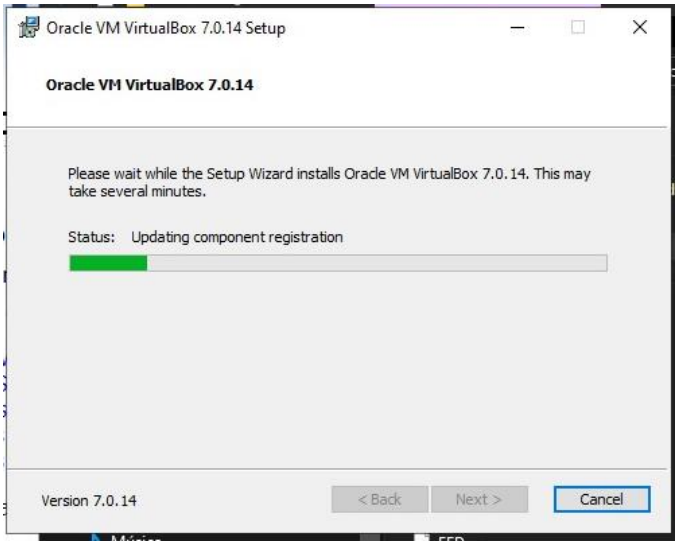


Figura 2 Captura de pantalla de instalación de VirtualBox

Una vez finalizado el proceso de instalación virtualbox estará lista para su uso y podremos apreciar su página principal en la cual encontramos la sección de añadir una máquina virtual. Usaremos esa opción para añadir una nueva máquina virtual para esto será necesario haber instalado antes la imagen del sistema operativo a usar. Una vez dentro de la sección deberemos seleccionar la carpeta donde se guardar la máquina virtual al igual que la imagen del sistema previamente instalada.

Figura 3 Administrador virtualbox

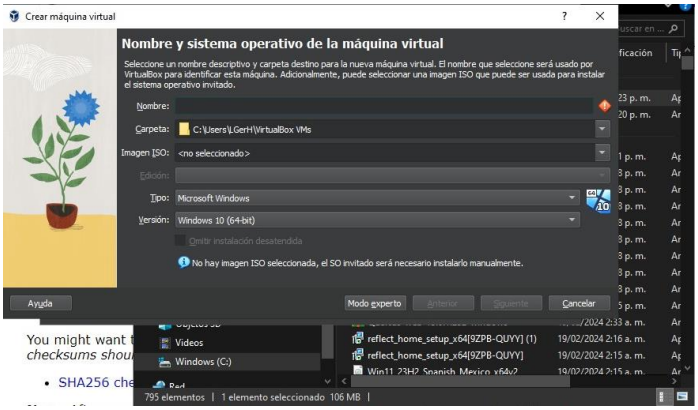
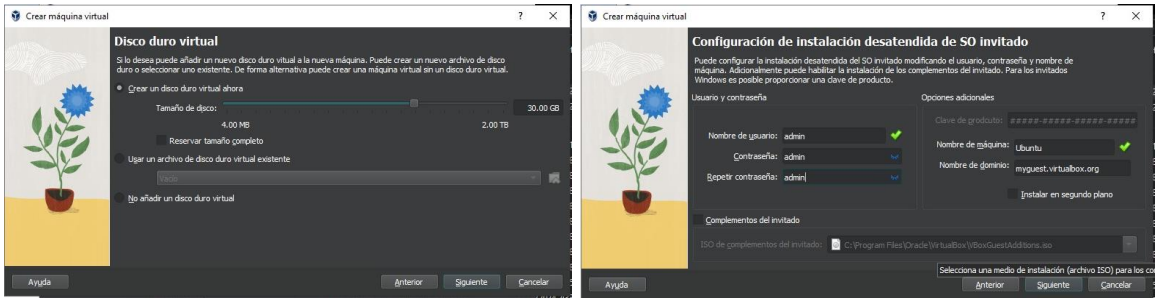


Figura 4 Crear máquina virtual virtualbox

Continuamos con la configuración de los requerimientos de nuestra máquina virtual en donde especificaremos un usuario y contraseña. Al igual que después especificaremos cuantos recursos de nuestra computadora destinaremos para el momento de ejecución de la máquina virtual en mi caso destine 4gb de ram, 2 cpus virtuales y 30gb de disco virtual.



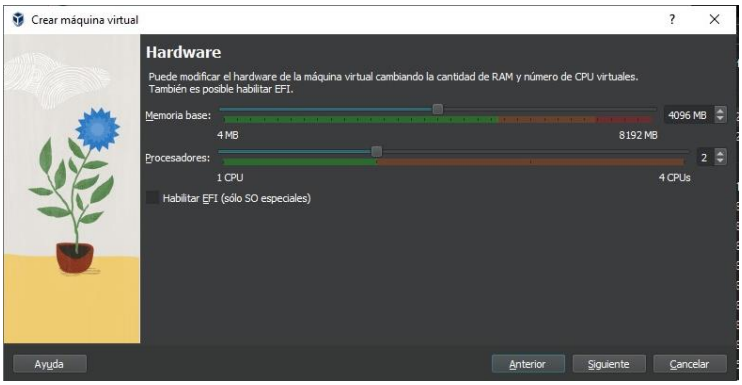


Figura 5 Configurar máquina virtual VirtualBox

3.1.4 Instalación del sistema operativo Ubuntu

Una vez finalizada la creación de nuestra máquina virtual la encenderemos, y al ser la primera vez que la encendemos nos arrojara a una ventana negra de elección de sistema operativo para después iniciar con la instalación de Ubuntu. Nos pedirá una configuración básica como poner usuario y contraseña y si queremos instalar otras

aplicaciones populares, después de esta configuración básica abrimos terminado con la instalación de nuestro nuevo sistema operativo.

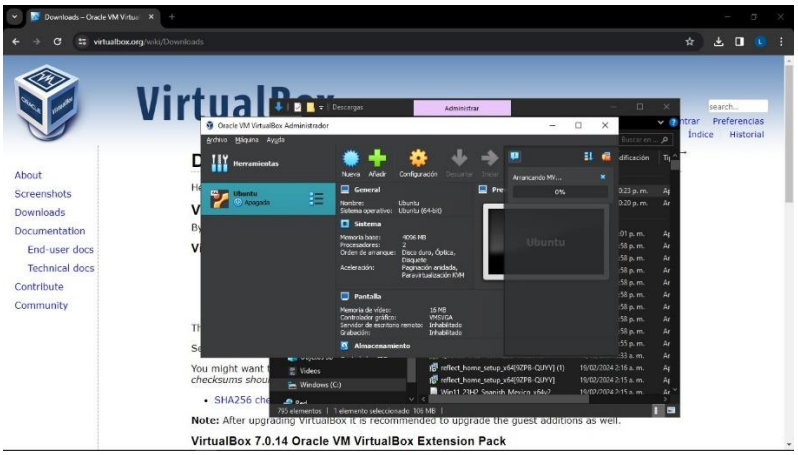


Figura 6 Administrador VirtualBox

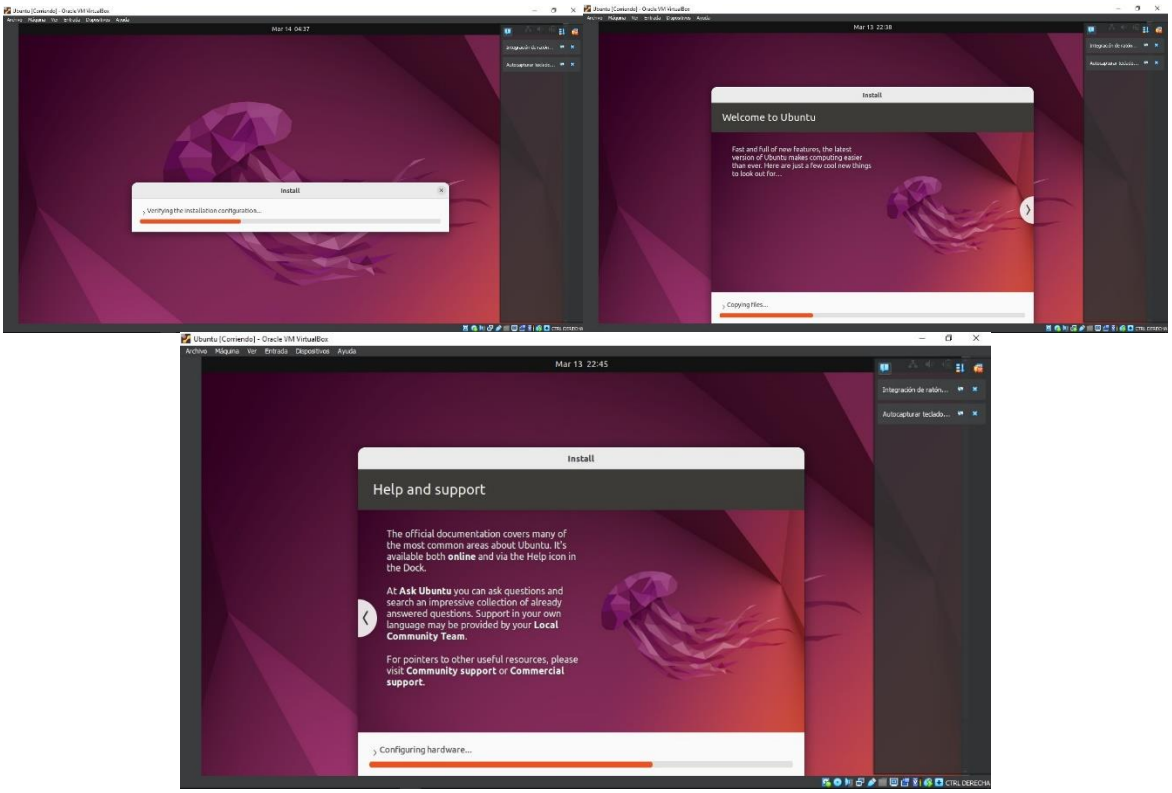


Figura 7 Máquina virtual Ubuntu

3.1.5 Configuración del Entorno de Desarrollo

La configuración del entorno de desarrollo es un paso para asegurar que todos los miembros del equipo tengan un entorno uniforme y funcional para el desarrollo del software. Este proceso incluye la instalación y configuración de diversas herramientas y tecnologías necesarias para el proyecto.

3.1.5.1 Java Development Kit

El Java Development Kit (JDK) es esencial para desarrollar aplicaciones en Java, ya que proporciona las herramientas necesarias para compilar y ejecutar programas. La instalación del JDK se realiza siguiendo un código específico que asegura que la versión correcta esté configurada en el sistema, permitiendo así el desarrollo efectivo de la aplicación utilizando Spring Boot.

```
# Instalación de OpenJDK 21
sudo apt update
sudo apt install openjdk-21-jdk
java -version
```

Figura 8 Código para instalar Java Development Kit

3.1.5.2 Visual Studio Code y Extensiones

Visual Studio Code (VS Code) es un editor de código fuente popular que se utiliza para el desarrollo en Java y otras tecnologías. La instalación de VS Code se acompaña de la adición de extensiones específicas, como el Extension Pack for Java, Spring Boot Extension Pack, Maven for Java, y Project Manager for Java. Estas extensiones mejoran la funcionalidad del editor, facilitando tareas como la gestión de proyectos, la integración con Maven y el soporte para el desarrollo de aplicaciones Spring Boot.

1. Instalar VS Code:

```
sudo snap install code --classic
```

Figura 9 Código para instalar VS Code

2. Extensiones requeridas:

- Extension Pack for Java

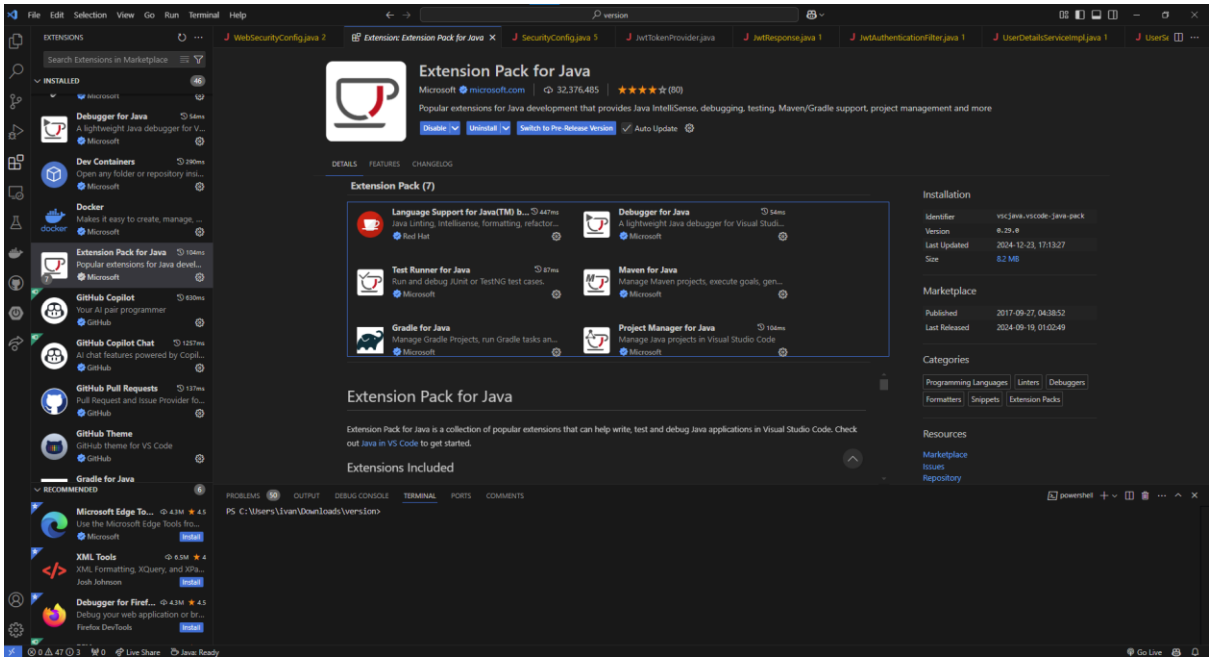


Figura 10 Instalar Extension Pack for Java en VS Code

- Spring Boot Extension Pack

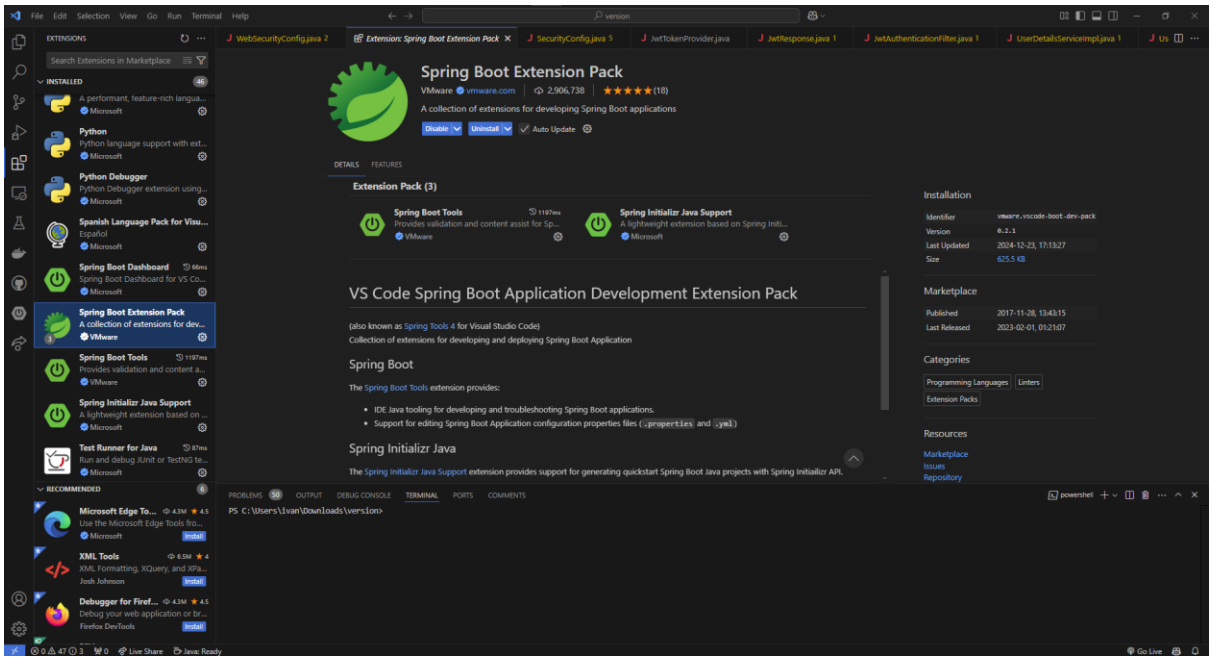


Figura 11 Instalar Extension Pack for Java en VS Code

- Maven for Java

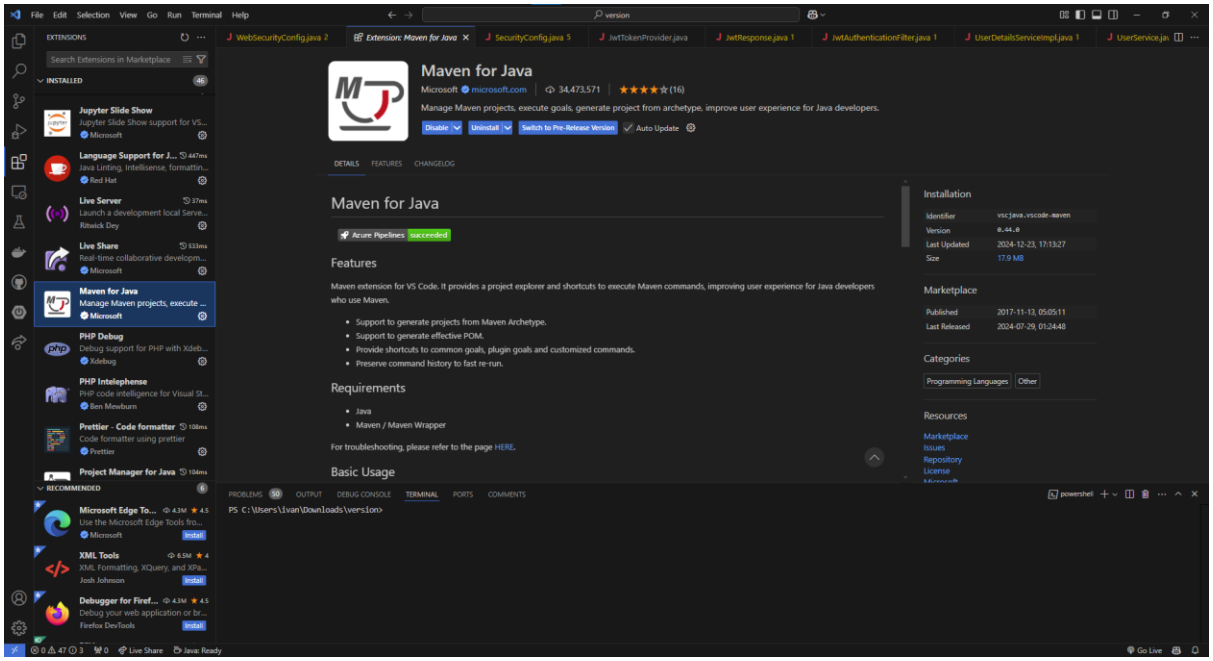


Figura 12 Instalar Maven for Java en VS Code

- Project Manager for Java

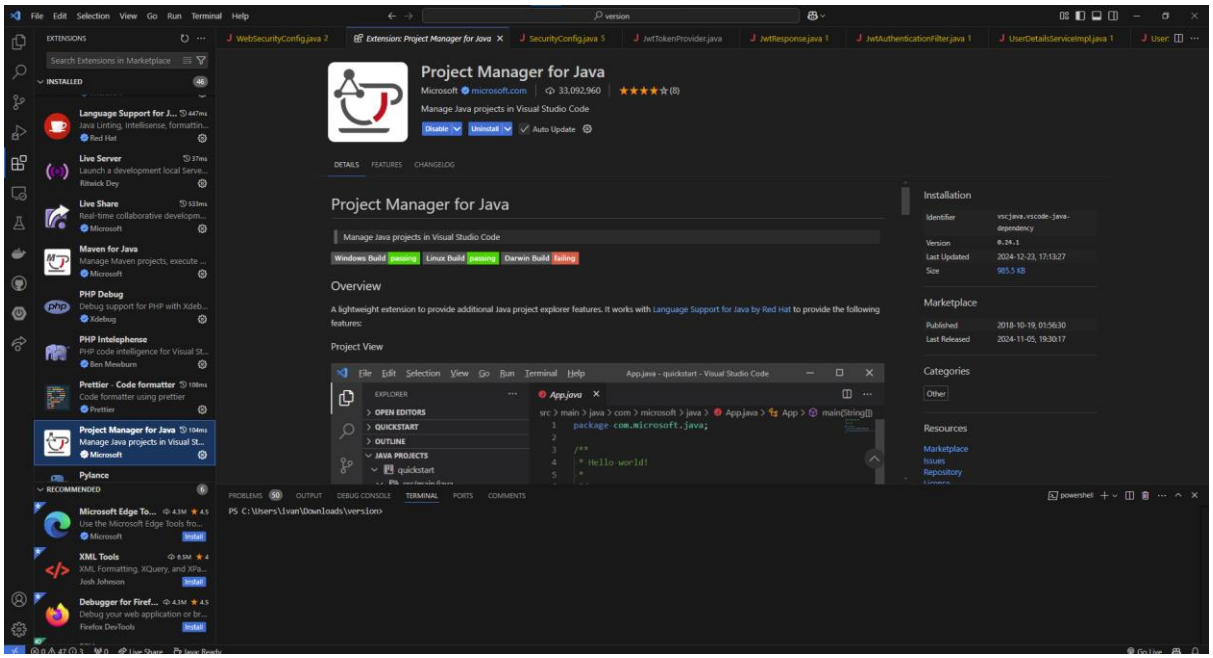


Figura 13 Instalar Project Manager for Java en VS Code

3.1.5.3 XAMPP y MySQL

XAMPP es una herramienta que permite instalar y gestionar un servidor web local junto con una base de datos MySQL. La configuración comienza con la descarga e instalación de XAMPP en Linux, seguida por el inicio de los servicios necesarios para ejecutar el servidor y la base de datos. Esta configuración es fundamental para

almacenar los datos de la aplicación, permitiendo a los desarrolladores interactuar con la base de datos localmente durante el desarrollo.

1. Descargar XAMPP para Linux desde Apache Friends

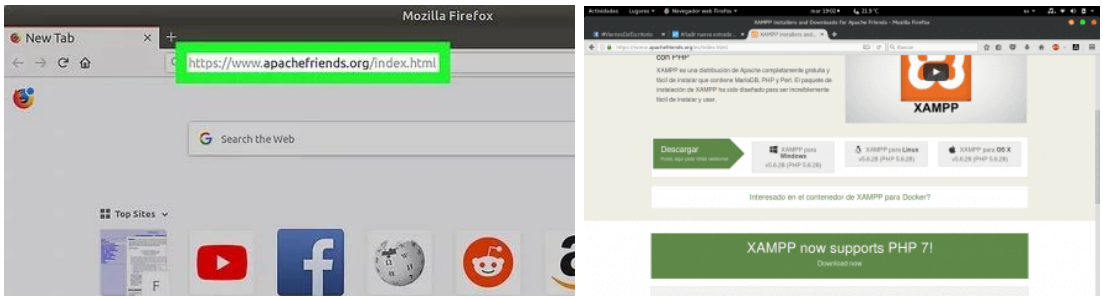


Figura 14 Descargar XAMPP para Linux

2. Instalar:

```
chmod +x xampp-linux-*-installer.run
sudo ./xampp-linux-*-installer.run
```

Figura 15 Código para instalar XAMPP

Iniciar servicios:

```
sudo /opt/lampp/lampp start
```

Figura 16 Código para iniciar servicios

Configuración de Red

1. Configuración de red en VirtualBox:
    - Adaptador en modo puente
    - Seleccionar interfaz de red física
2. Configuración en Ubuntu:
    - IP estática: 192.168.1.100
    - Máscara: 255.255.255.0
    - Gateway: 192.168.1.1

Verificar conectividad:

```
ping 8.8.8.8
```

Figura 17 Código para verificar conectividad

3.1.5.4 Verificación del Entorno

La Verificación del Entorno implica comprobar que todas las herramientas y tecnologías necesarias estén correctamente instaladas y configuradas. Esto se realiza ejecutando comandos específicos para verificar las versiones instaladas del

JDK, VS Code, XAMPP y otros componentes relevantes, asegurando así que el entorno esté listo para el desarrollo.

1. Comprobar versiones instaladas:

```
java -version        # Java 21
mysql --version      # MySQL 8.0
code --version       # VS Code
```

Figura 18 Código para comprobar versiones instaladas

3.1.6 Configuraciones Específicas por Equipo

Cada miembro del equipo debe sus configuraciones específicas, lo cual incluye detalles como la dirección IP asignada, los puertos utilizados, las credenciales de base de datos locales y las rutas de almacenamiento de proyectos. Esta documentación es vital para mantener un registro claro de las configuraciones individuales, facilitando la colaboración y resolución de problemas dentro del equipo durante el desarrollo del proyecto.

Cada miembro del equipo tiene configuraciones específicas:

- Dirección IP asignada
- Puertos utilizados
- Credenciales de base de datos locales
- Rutas de almacenamiento de proyectos

4 Desarrollo de la API

4.1 Arquitectura del sistema

El sistema de recomendación está construido sobre una arquitectura MVC, implementada con Spring Boot y siguiendo los principios de diseño REST. Este modelo sigue el patrón MVC (Modelo-Vista-Controlador), donde el controlador actúa como intermediario entre el modelo (servicio y repositorio) y la vista (frontend).

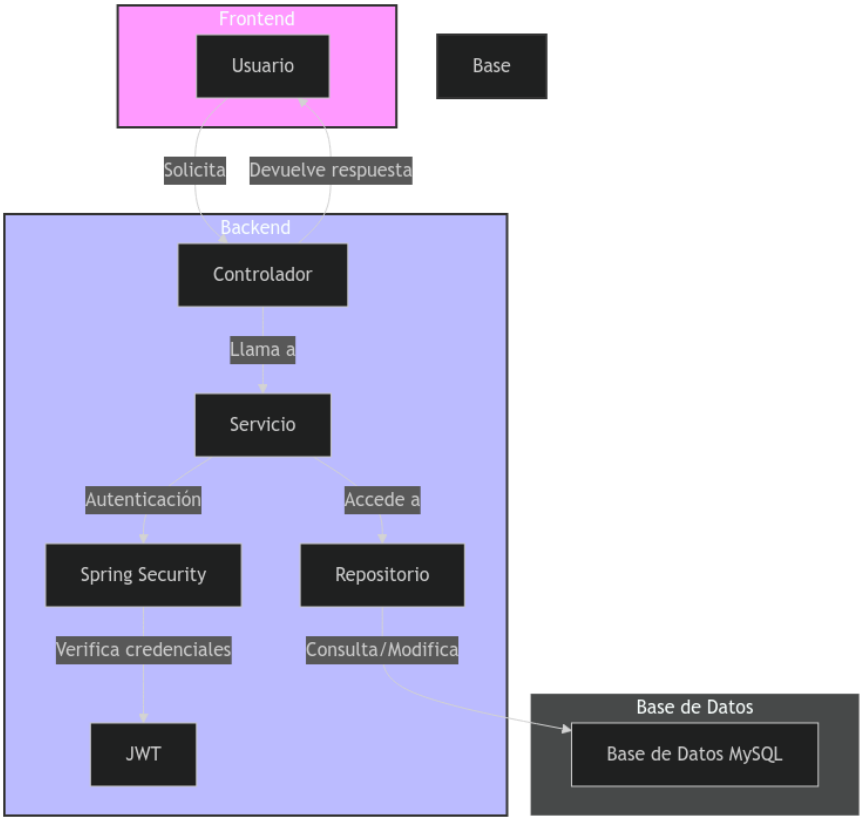


Figura 19 Arquitectura del sistema

Explicación del Diagrama

- **Usuario**: Representa al cliente que interactúa con la aplicación a través de la interfaz de usuario.
- **Controlador**: Recibe las solicitudes del usuario y las dirige al servicio correspondiente.
- **Servicio**: Contiene la lógica de negocio y coordina las operaciones entre el controlador y el repositorio.
- **Repositorio**: Interactúa directamente con la base de datos para realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar).

- **Base de Datos MySQL**: Almacena la información de usuarios, libros, series y películas.
- **Spring Security**: Maneja la autenticación y autorización de los usuarios.
- **JWT (JSON Web Tokens)**: Utilizado para mantener sesiones seguras y verificar la identidad del usuario.

4.2 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).

4.3 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

Books API Endpoints

Buscar Libros
```http
GET /api/books/search?query={query}
```
- **Descripción**: Busca libros por término de búsqueda
- **Parámetros**:
  - `query` (requerido): Término de búsqueda
- **Ejemplo**: `/api/books/search?query=harry+potter`
- **Respuesta**: Array de objetos libro
```json
[
  {
    "title": "Harry Potter and the Philosopher's Stone",
    "author": "J.K. Rowling",
    "isbn": "9780747532699",
    "publishYear": "1997",
    "coverUrl": "https://..."
  }
]
```

Obtener Libro por ISBN
```http
GET /api/books/{isbn}
```
- **Descripción**: Obtiene detalles de un libro específico
- **Parámetros**:
  - `isbn` (requerido): ISBN del libro
- **Ejemplo**: `/api/books/9780747532699`
- **Respuesta**: Objeto libro con detalles completos
```json
{
  "title": "Harry Potter and the Philosopher's Stone",
  "author": "J.K. Rowling",
  "isbn": "9780747532699",
  "publishYear": "1997",
  "description": "...",
  "coverUrl": "https://...",
  "publisher": "Bloomsbury",
  "genres": ["Fantasy", "Young Adult"]
}
```

Obtener Libros Destacados
```http
GET /api/books/featured
```
- **Descripción**: Obtiene una lista de libros destacados
- **Respuesta**: Array de objetos libro
```json
[
  {
    "title": "Book Title",
    "author": "Author Name",
    "isbn": "1234567890",
    "rating": "4.5",
    "coverUrl": "https://..."
  }
]
```

````
### Shows API Endpoints

#### Buscar Shows/Series
```http
GET /api/shows/search?query={query}
```
- **Descripción**: Busca shows/series por término de búsqueda
- **Parámetros**:
  - `query` (requerido): Término de búsqueda
- **Ejemplo**: `/api/shows/search?query=breaking+bad`
- **Respuesta**: Array de objetos show/serie
```json
[
  {
    "title": "Breaking Bad",
    "year": "2008",
    "genres": ["Crime", "Drama", "Thriller"],
    "rating": "9.5",
    "coverUrl": "https://..."
  }
]
```
````

````
#### Obtener Show/Serie por ID
```http
GET /api/shows/{id}
```
- **Descripción**: Obtiene detalles de un show/serie específico
- **Parámetros**:
  - `id` (requerido): ID del show/serie
- **Ejemplo**: `/api/shows/169`
- **Respuesta**: Objeto show/serie con detalles completos
```json
{
  "title": "Breaking Bad",
  "year": "2008",
  "genres": ["Crime", "Drama", "Thriller"],
  "rating": "9.5",
  "description": "...",
  "coverUrl": "https://...",
  "cast": ["Bryan Cranston", "Aaron Paul", "Anna Gunn"]
}
```
````

````
#### Obtener Shows/Series Destacados
```http
GET /api/shows/featured
```
- **Descripción**: Obtiene una lista de shows/series destacados
- **Respuesta**: Array de objetos show/serie
```json
[
  {
    "title": "Show Title",
    "year": "Year",
    "genres": ["Genre1", "Genre2"],
    "rating": "Rating",
    "coverUrl": "https://..."
  }
]
```
````

```
### User API Endpoints

#### Obtener Perfil de Usuario
```http
GET /api/users/profile
```
- **Descripción**: Obtiene el perfil del usuario autenticado
- **Requiere**: Token JWT
- **Respuesta**: Objeto con datos del usuario
```json
{
  "id": 1,
  "username": "usuario123",
  "email": "usuario@email.com",
  "nombre": "Nombre",
  "apellido": "Apellido",
  "profilePhotoUrl": "https://...",
  "rol": "USER",
  "fechaRegistro": "2024-01-20T10:30:00Z"
}
```

Actualizar Perfil de Usuario
```http
PUT /api/users/profile
```
- **Descripción**: Actualiza los datos del perfil del usuario
- **Requiere**: Token JWT
- **Body**:
```json
{
  "nombre": "Nuevo Nombre",
  "apellido": "Nuevo Apellido",
  "email": "nuevo@email.com"
}
```
- **Respuesta**: Objeto con datos actualizados

Subir Foto de Perfil
```http
POST /api/users/profile/photo
```
- **Descripción**: Sube o actualiza la foto de perfil
- **Requiere**: Token JWT
- **Body**: Form-data con archivo de imagen
- **Parámetros**:
  - `photo` (requerido): Archivo de imagen (max 5MB, formatos: jpg, png)
- **Respuesta**: URL de la nueva foto

Gestión de Usuarios (Solo Admin)
```http
GET /api/users
```
- **Descripción**: Lista todos los usuarios
- **Requiere**: Token JWT con rol ADMIN
- **Respuesta**: Array de usuarios

```http
DELETE /api/users/{id}
```
- **Descripción**: Elimina un usuario
- **Requiere**: Token JWT con rol ADMIN
- **Parámetros**:
  - `id` (requerido): ID del usuario
- **Respuesta**: Confirmación de eliminación
```

Figura 20 Documentación de endpoints.

5 Diagrama de Despliegue

El siguiente diagrama de despliegue UML representa la infraestructura del sistema de recomendación, destacando los nodos físicos y las conexiones entre ellos, así como las tecnologías utilizadas. Este diagrama es crucial para entender cómo se distribuyen los componentes del sistema y cómo interactúan entre sí.

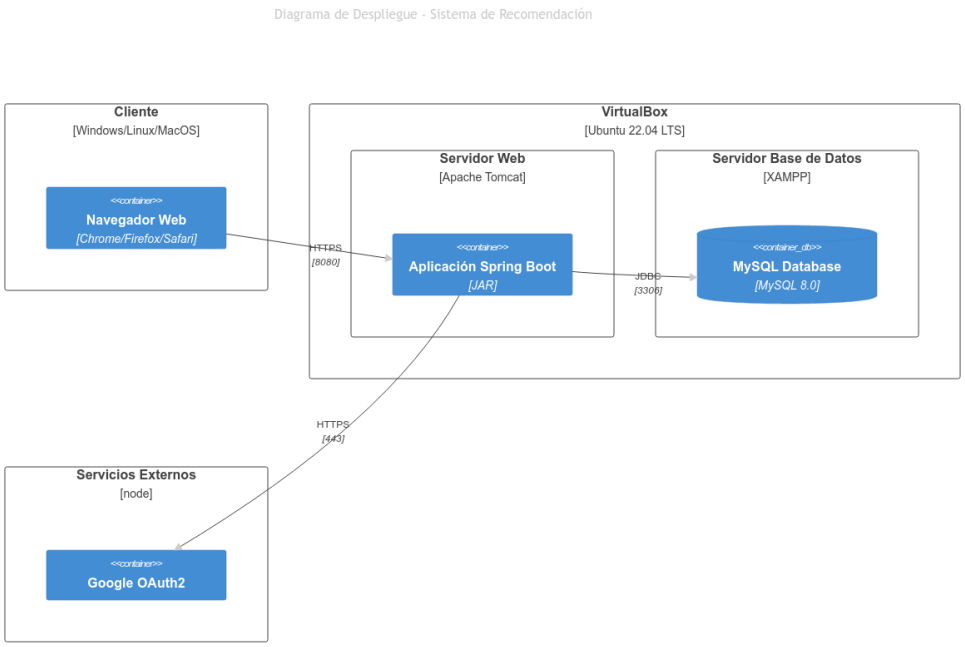


Figura 21 Diagrama de Despliegue.

Descripción de Componentes

1. **Nodo Cliente**
   - Navegador web que consume la aplicación
   - Requisitos: Navegador moderno con soporte JavaScript
   - Puerto de acceso: 8080 para desarrollo
2. **Nodo VirtualBox**
   - Sistema Operativo: Ubuntu 22.04 LTS
   - Recursos: 4GB RAM, 50GB almacenamiento
   - Configuración de red: Modo puente
3. **Servidor Web (Apache Tomcat)**
   - Contenedor de servlets para Spring Boot
   - Aplicación empaquetada como JAR

- Configuración SSL para HTTPS

4. **Servidor Base de Datos (XAMPP)**
    - MySQL 8.0 para persistencia
    - PHPMyAdmin para administración
    - Puertos: 3306 (MySQL), 80 (PHPMyAdmin)

Interacciones entre Nodos

1. **Cliente → Servidor Web**
    - Protocolo: HTTPS
    - Puerto: 8080
    - Tráfico: Peticiones REST, contenido estático
2. **Servidor Web → Base de Datos**
    - Protocolo: JDBC
    - Puerto: 3306
    - Conexión pool configurada en Spring Boot
3. **Servidor Web → Servicios OAuth**
    - Protocolo: HTTPS
    - Puerto: 443
    - Autenticación mediante tokens

6 Pruebas Funcionales

6.1 Pruebas de Endpoints

6.1.1 Pruebas de Autenticación y Autorización

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

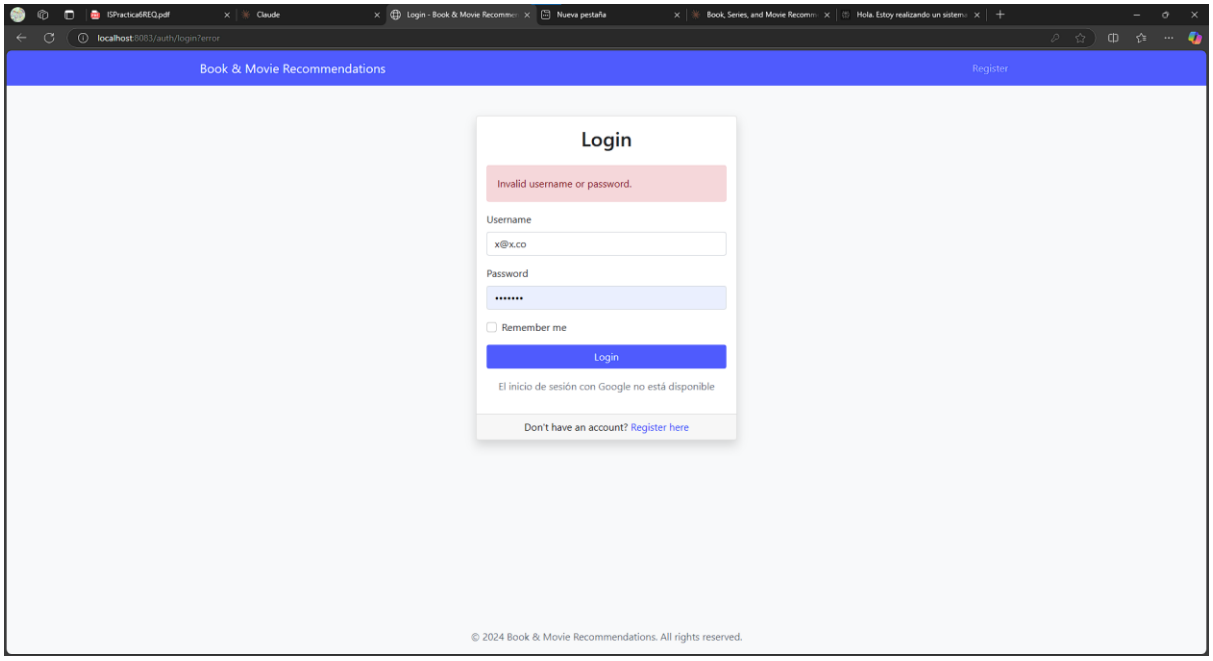


Figura 22 Resultados de pruebas de autenticación

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

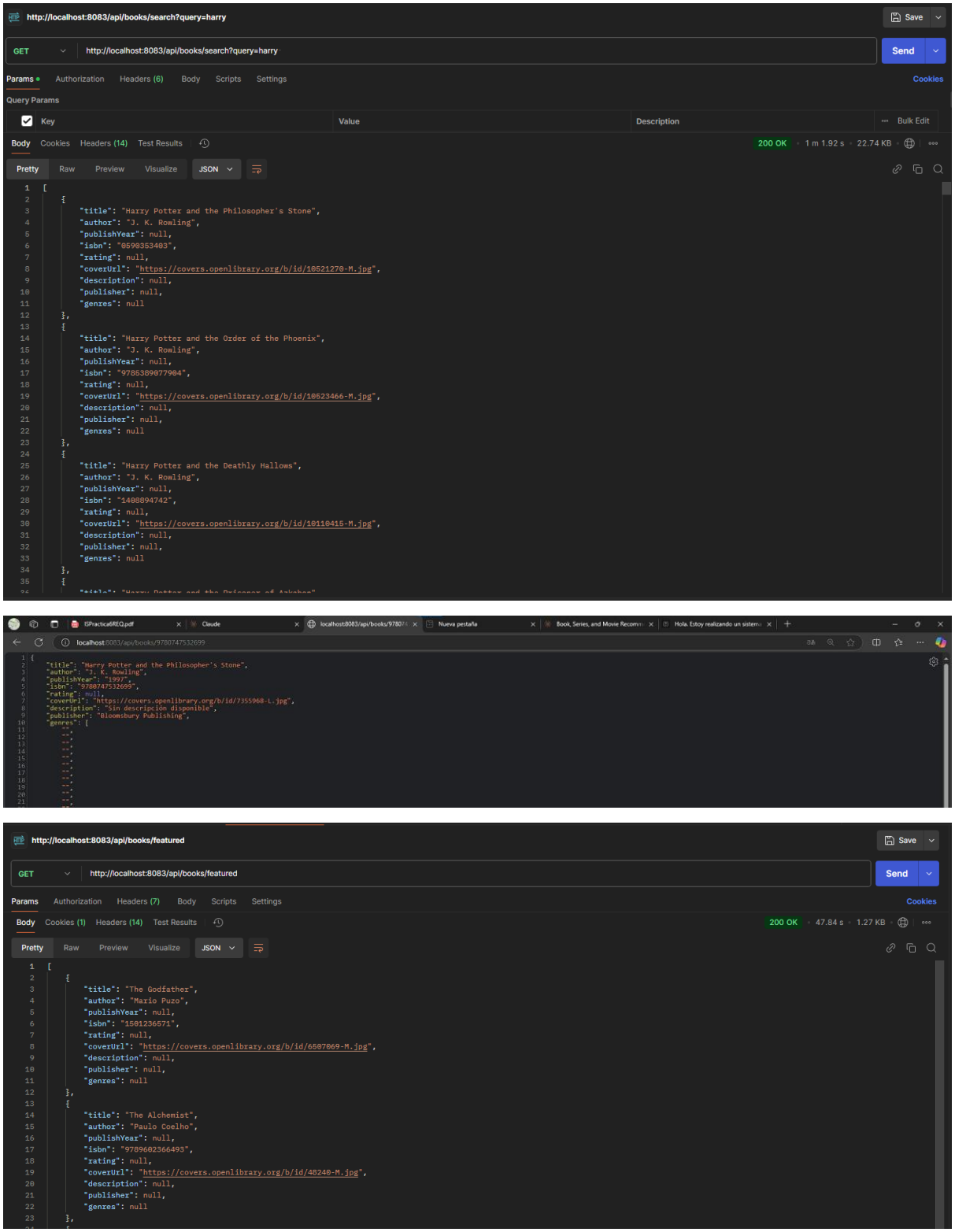


Figura 23 Resultados de pruebas de API de libros

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

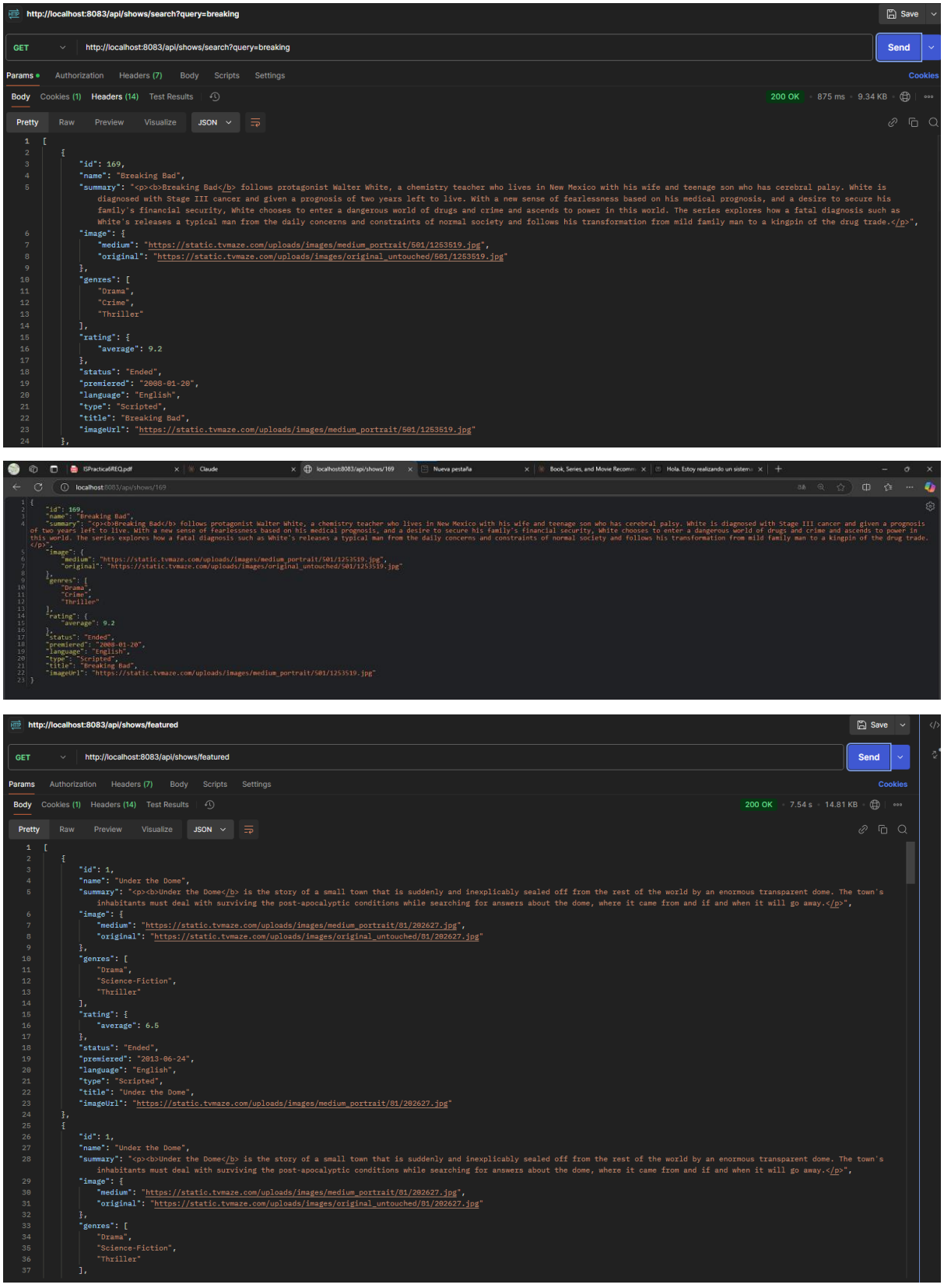


Figura 24 Resultados de pruebas de API de shows

6.1.2 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

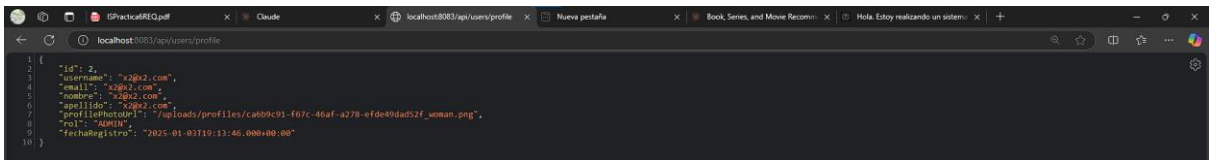


Figura 25 Resultados de pruebas de API de usuario

6.1.3 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.

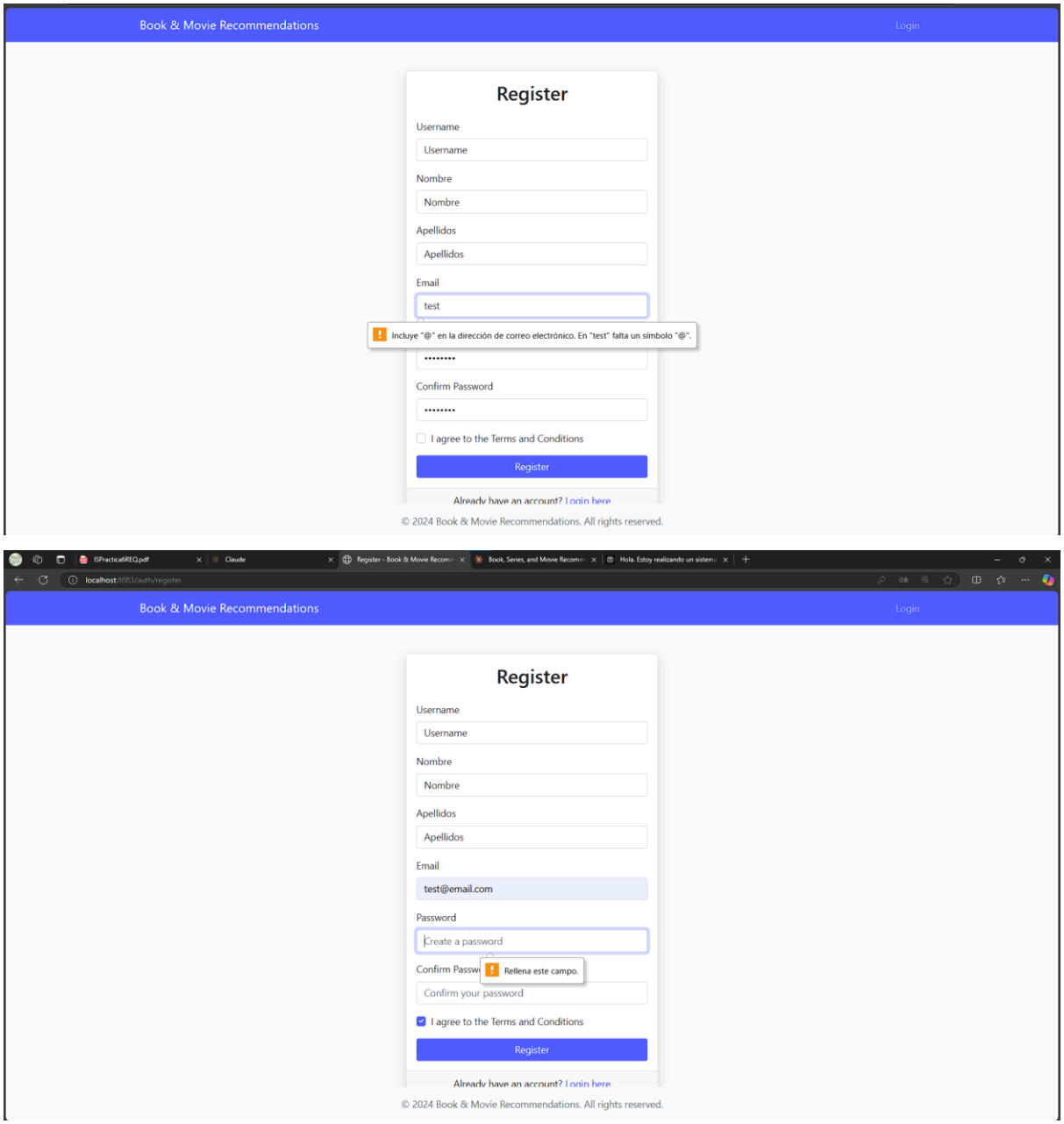


Figura 26 Resultados de pruebas de validación

6.1.4 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

6.2 Pruebas No Funcionales

6.2.1 Pruebas de Rendimiento

Las **Pruebas de Rendimiento** se dividen en dos escenarios: carga normal y carga pico. En el escenario de carga normal, se evalúa el comportamiento del sistema con 100 usuarios concurrentes durante 10 minutos, analizando métricas como el tiempo promedio y máximo de respuesta, así como el porcentaje de errores para varios endpoints. En contraste, el escenario de carga pico simula una situación más extrema con 500 usuarios concurrentes durante 5 minutos para evaluar cómo responde el sistema bajo presión máxima. 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

Escenario 1: Carga Normal

- **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 Normal.

Escenario 2: Carga Pico

- **Usuarios concurrentes**: 500
- **Duración**: 5 minutos
- **Tasa de peticiones**: 200 por segundo

Pruebas de Estrés

Objetivo: Identificar el punto de ruptura del sistema

Usuarios	Tiempo Respuesta	CPU %	Memoria %	Estado
1000	1.2s	65%	75%	Estable
2000	2.5s	85%	85%	Degradado

3000	5.0s	95%	95%	Crítico

Tabla 8 Pruebas de Estrés

6.2.2 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

Pruebas de Autenticación

Escenario	Resultado	Notas
Fuerza Bruta	Bloqueado	Implementado rate limiting
Token JWT	Seguro	Tokens con expiración de 1 hora
OAuth2	Verificado	Flujo seguro con Google

Tabla 10 Pruebas de Autenticación

6.2.3 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.

Compatibilidad de Navegadores

Navegador	Versión	Resultado
Chrome	120+	✅ Completo
Firefox	115+	✅ Completo

Safari	16+	✅ Completo
Edge	120+	✅ Completo

Tabla 11 Compatibilidad de Navegadores

Pruebas de Responsive Design

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

Tabla 12 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

6.2.4 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.

6.2.5 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 13 Uso de Recursos

8 Conclusiones

El desarrollo de la práctica 5 ha permitido a nuestro equipo desarrollar una API funcional en un entorno simulado, cumpliendo con los requisitos establecidos en el documento de práctica. A través de la configuración del entorno de desarrollo, hemos aprendido a instalar y configurar herramientas esenciales como Java Development Kit, Visual Studio Code y XAMPP, lo que nos ha proporcionado una base sólida para el desarrollo de software.

Durante el desarrollo de la API, enfrentamos desafíos relacionados con la autenticación y autorización, pero logramos implementar un sistema seguro utilizando JWT y configuraciones adecuadas en Spring Security. Las pruebas funcionales y no funcionales realizadas nos han permitido validar el rendimiento y la seguridad de la aplicación, asegurando que se cumplan los estándares requeridos.

Además, la creación de un diagrama de despliegue nos ayudó a visualizar la arquitectura del sistema y las interacciones entre los diferentes componentes. La documentación detallada del proceso de desarrollo y las pruebas realizadas facilitará futuras implementaciones y mantenimientos del sistema. Esta práctica reforzó nuestros conocimientos técnicos y que también ha mejorado nuestra capacidad para trabajar en equipo y gestionar proyectos de software siguiendo metodologías ágiles como Scrum.

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.