CURSO 23 - 24

Documentación
BOOK 'N BOOK

REALIZADO POR:
MARÍA ESCRIBANO VERDE

2º DAW - IES ALIXAR

ÍNDICE

INTRODUCCIÓN

Book 'N Book es una aplicación web diseñada para amantes de la lectura. Funciona como una red social y gestor de lecturas, proporcionando una plataforma interactiva para compartir experiencias, recomendar libros y conectar con otros lectores. En general, se denomina como gestión de lectura la creación de perfiles por parte de los usuarios desde donde detallar sus preferencias de lectura, agregar libros a sus bibliotecas virtuales, marcándolos como leídos, en proceso o por leer, y también pueden registrar reseñas y calificaciones. Además, la aplicación ofrece estadísticas detalladas sobre los hábitos de lectura de los usuarios, como el número de libros leídos, géneros preferidos y tiempo dedicado a la lectura.

Una característica destacada de *Book 'N Book* es la capacidad de formar y participar en clubes de lectura, como clases de instituto o grupos de amigos, donde los usuarios participan en lecturas conjuntas. Tras la selección del libro, se sigue el progreso de lectura colectiva, y cuando todos los miembros la han completado, se abre un espacio de debate donde los participantes pueden discutir y compartir sus opiniones. Todos los debates se guardan como entradas en el chat del grupo, permitiendo a los usuarios acceder al historial de discusiones.

En resumen, *Book 'N Book* ofrece una experiencia integral para los amantes de la lectura, permitiéndoles interactuar, descubrir nuevas lecturas y disfrutar de una comunidad literaria virtual.

IDENTIFICACIÓN DE LAS NECESIDADES DEL PROYECTO

LG → Acceso a la aplicación

LG01 - El sistema deberá permitir a los usuarios registrarse mediante un formulario de registro.

LG02 - El sistema deberá permitir a los usuarios iniciar sesión utilizando credenciales válidas y hacer logout.

LG03 - El sistema deberá tener un sistema de recuperación de contraseña para los usuarios que hayan olvidado sus credenciales.

US → Usuarios

US01 - El sistema deberá permitir a los usuarios editar su perfil, incluyendo información personal, preferencias de lectura, foto de perfil, etc.

US02 - El sistema deberá proporcionar una página de inicio donde los usuarios puedan ver recomendaciones de libros, actualizaciones de sus amigos y clubes a los que pertenecen.

US03 - El sistema deberá permitir a los usuarios buscar libros por título, autor, género, etc.

US04 - El sistema deberá permitir a los usuarios agregar libros a su biblioteca personal, indicando su estado de lectura (leído, en proceso, por leer).

US05 - El sistema deberá permitir a los usuarios dejar reseñas y calificaciones para los libros en su biblioteca.

US06 - El sistema deberá permitir a los usuarios seguir a otros usuarios y ver sus actividades en un feed de noticias.

US07 - El sistema deberá permitir a los usuarios crear y unirse a clubes de lectura.

US08 - El sistema deberá generar estadísticas sobre los hábitos de lectura de los usuarios, como el número de libros leídos, géneros preferidos, etc.

US09 - El sistema deberá permitir a los usuarios compartir su perfil o libros mediante enlace.

US10 - Los usuarios podrán denunciar comentarios inapropiados tanto en debates como en comentarios públicos en la aplicación. Esto incluye la capacidad de marcar comentarios que contengan contenido ofensivo, spam o cualquier otra violación de las normas de la comunidad.

US11 - Los usuarios podrán reaccionar a los comentarios de libros de otros usuarios mediante emoticonos tipo corazón, pulgar arriba, risa, etc. Esto permitirá a los usuarios expresar sus opiniones y sentimientos de manera rápida y sencilla en respuesta a los comentarios de otros usuarios.

US12 - Los usuarios podrán recomendar la publicación de un libro, proporcionando información sobre el título, autor, género, etc.
AG → Administradores de clubes de lectura
AG01- El sistema deberá permitir a los administradores de clubes proponer libros para la lectura y que los miembros voten por el libro elegido.
AG02 - El sistema deberá proporcionar herramientas de gestión para los administradores de clubes , como la capacidad de agregar usuarios, asignar privilegios (convertirlos en administradores), etc.

GL → Grupo de lectura
GL01 - El sistema deberá mostrar el progreso de lectura colectiva en los clubes, indicando cuántos miembros han completado la lectura del libro seleccionado.
GL02 - El sistema deberá abrir un espacio de debate dentro de los clubes una vez que todos los miembros hayan completado la lectura del libro, donde puedan discutir y compartir opiniones.
GL03 - El sistema deberá almacenar el historial de debates en los clubes para su consulta posterior.

AA → Administradores de la aplicación
AA01 - El sistema deberá permitir a los administradores gestionar la lista de usuarios, incluyendo la capacidad de agregar, eliminar y editar cuentas de usuario.
AA02 - El sistema deberá permitir a los administradores gestionar los clubes de lectura, incluyendo la capacidad de crear, eliminar y editar clubes.
AA03 - El sistema deberá proporcionar a los administradores herramientas para moderar los debates en los clubes de lectura (eliminar contenido inapropiado).
AA04 - El sistema deberá permitir a los administradores moderar los comentarios y reseñas de libros, incluyendo la capacidad de eliminar o editar contenido inapropiado o fuera de las normas de la comunidad.

AA05 - El sistema deberá proporcionar a los administradores herramientas de comunicación para enviar mensajes y notificaciones a los usuarios y clubes.
AA06 - El sistema deberá proporcionar a los administradores la capacidad de realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) sobre la base de datos de lecturas.
AA07 - El sistema deberá proporcionar a los administradores un panel de control intuitivo y fácil de usar para acceder a todas las herramientas de gestión y administración de la aplicación.
AA08 - Los administradores tendrán la capacidad de revisar las denuncias de los usuarios sobre comentarios inapropiados y tomar decisiones apropiadas, como eliminar el comentario, advertir al usuario infractor o tomar medidas disciplinarias adicionales según corresponda.
AA09 - Los administradores podrán validar la solicitud del libro recomendado por los usuarios antes de su publicación.

SA → Sistema de la aplicación
SA01 - El sistema deberá ser compatible con dispositivos móviles para que los usuarios puedan acceder a la aplicación desde sus smartphones y tabletas.

POSIBLES MEJORAS A FUTURO

MF01 - Permitir a los usuarios guardar comentarios sobre libros para referencia futura, con la capacidad de acceder a una lista de comentarios guardados para su revisión.
MF02 - Implementar soporte multilingüe en la interfaz de usuario para que los usuarios puedan elegir su idioma preferido.
MF03 - Integrar la aplicación con Google Libros u otra API de biblioteca interactiva para proporcionar acceso directo a una amplia variedad de libros y opciones de búsqueda avanzada.
MF04 - Incluir enlaces de compra directa de libros, tanto en formato físico como electrónico, desde los detalles de cada libro en la

aplicación, permitiendo a los usuarios comprar rápidamente los libros que deseen.

MF05 - Incluir opciones de accesibilidad que permitan a los usuarios ajustar el contraste, el tamaño de las fuentes u otras configuraciones relevantes para mejorar la accesibilidad de la aplicación, garantizando una experiencia de usuario óptima para usuarios con diferentes necesidades y preferencias.

MF06 - El sistema deberá implementar un modo de visualización oscuro o claro en la aplicación web, permitiendo a los usuarios alternar entre diferentes temas de color según sus preferencias de lectura y comodidad visual.

ANÁLISIS CON ALTERNATIVAS DEL MERCADO

En el mercado actual, existen varias alternativas para la gestión de lecturas y la conexión entre lectores, sin embargo, cada una presenta sus propios puntos fuertes y débiles en comparación con *Book 'N Book*.

1. **Goodreads** es una de las plataformas más populares para los amantes de la lectura. Permite a los usuarios gestionar sus lecturas, dejar reseñas y calificaciones, y participar en grupos de discusión. Su principal punto fuerte radica en su gran comunidad de usuarios y en su extenso catálogo de libros. Sin embargo, algunas críticas señalan que su interfaz puede resultar abrumadora y poco intuitiva para algunos usuarios.
2. **LibraryThing** es otra opción para la gestión de bibliotecas virtuales y la conexión con otros lectores. Ofrece características similares a Goodreads, pero se destaca por su enfoque en la catalogación de bibliotecas personales y en la recomendación de libros. Su punto fuerte reside en su capacidad para gestionar grandes colecciones de libros de manera organizada. No obstante, su interfaz puede resultar menos moderna y atractiva en comparación con otras plataformas.
3. **Shelfari**, propiedad de Amazon, ofrece una experiencia similar a Goodreads y LibraryThing. Permite a los usuarios gestionar sus bibliotecas virtuales, dejar reseñas y calificaciones, y participar en

clubes de lectura. Su integración con Amazon facilita la compra de libros recomendados. Sin embargo, Shelfari ha perdido popularidad en los últimos años y su comunidad de usuarios es más reducida en comparación con otras plataformas.

Comparativamente, *Book 'N Book* busca destacarse ofreciendo una experiencia integral que combine la gestión de lecturas con la formación de clubes de lectura y las estadísticas detalladas sobre los hábitos de lectura de los usuarios. Su enfoque en la interactividad y la comunidad virtual literaria le permite diferenciarse de otras alternativas en el mercado. Sin embargo, su éxito dependerá de su capacidad para ofrecer una interfaz intuitiva y atractiva, así como de su capacidad para atraer y retener a una amplia base de usuarios.

JUSTIFICACIÓN DEL PROYECTO

La principal motivación que impulsa este proyecto es mi pasión por la lectura y mi formación como pedagoga. Personalmente, hasta el día de hoy, no he encontrado en el mercado aplicaciones dedicadas a la gestión de lecturas que satisfagan completamente mis necesidades, incluso entre las opciones de pago disponibles. Esta realidad es una de las razones por las cuales planteo el desarrollo de *Book 'N Book*, además, considerándola no sólo como una aplicación que cumpla con mis expectativas, sino que también integre características innovadoras que puedan beneficiar el ámbito educativo.

La idea de incorporar la funcionalidad de grupos y organizaciones dentro de la aplicación surge del hecho de que las tecnologías pueden ser herramientas interesantes para fomentar la lectura y la interacción social en entornos educativos. Considerando, de tal modo, *Book 'N Book* como una aplicación web que puede contribuir significativamente al objetivo del fomento lector en las nuevas generaciones tan apegadas a las tecnologías al proporcionar una plataforma interactiva y colaborativa para la comunidad educativa.

STACK TECNOLÓGICO

En este proyecto, se ha seleccionado un conjunto específico de tecnologías para crear una aplicación moderna, eficiente y escalable. A continuación, se describe cada componente del stack tecnológico y las razones por las que se ha optado por cada una de ellas:

Backend: se ha elegido Java debido a su madurez y estabilidad. Java es un lenguaje de programación ampliamente adoptado, conocido por su rendimiento en aplicaciones de misión crítica. Su ecosistema rico en bibliotecas y frameworks, específicamente Spring Boot, facilita el desarrollo rápido y eficiente de aplicaciones complejas. Las características de seguridad integradas de Java ha sido crucial para su elección.

Frontend: se ha desarrollado con Angular, una plataforma que proporciona una arquitectura estructurada basada en componentes. Esto facilita el desarrollo y mantenimiento de aplicaciones web complejas. Además, Angular incluye herramientas y optimizaciones integradas que mejoran el rendimiento de las aplicaciones web. La gran comunidad y el robusto ecosistema de bibliotecas y herramientas de Angular aceleran el desarrollo y la resolución de problemas.

Base de Datos: se gestiona por MySQL, un sistema de gestión de bases de datos relacional conocido por su fiabilidad y rendimiento. MySQL es capaz de manejar grandes volúmenes de datos de manera eficiente, y es ampliamente compatible con diversas plataformas y lenguajes de programación.

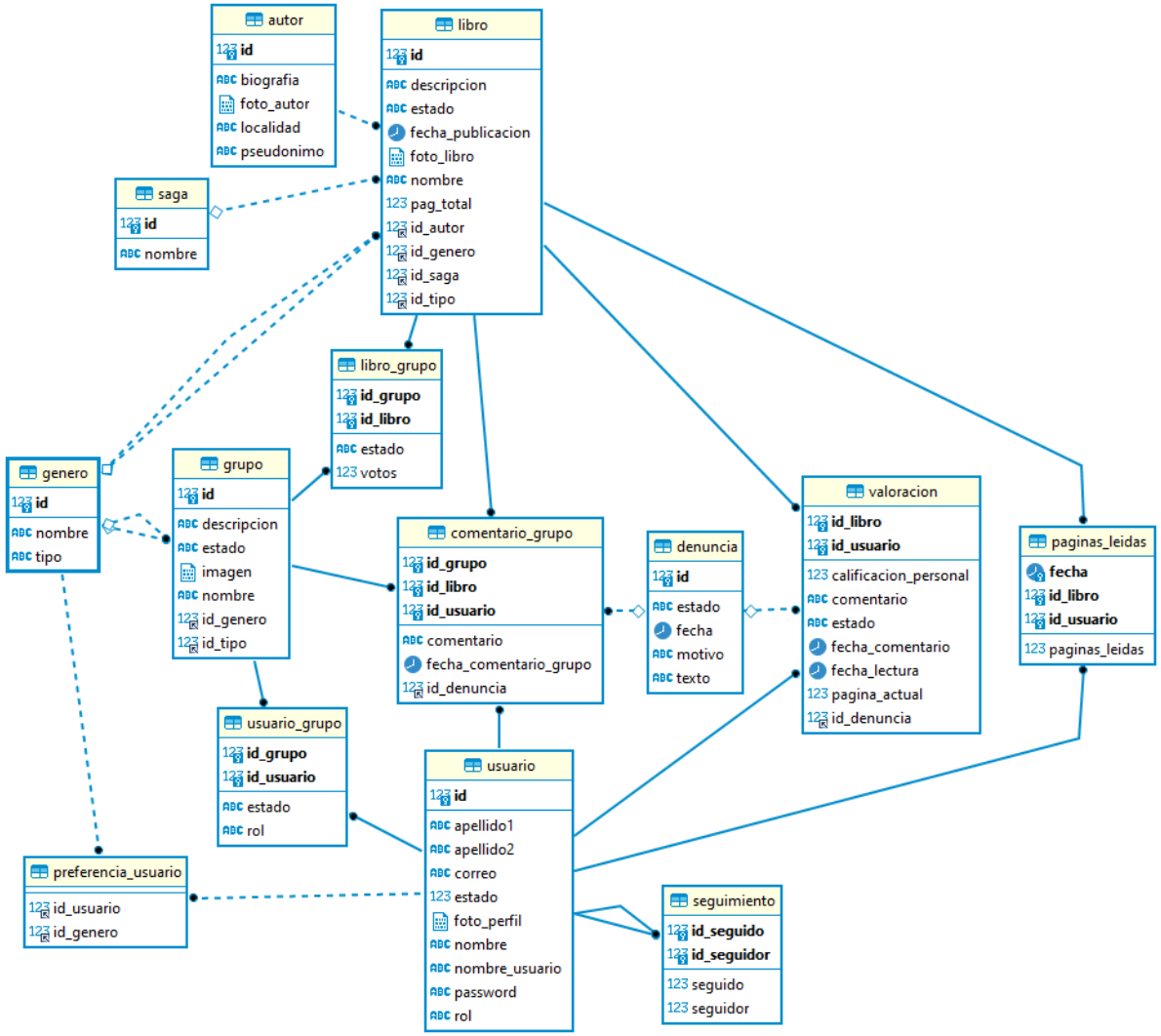
Pruebas y Documentación de APIs: se utiliza Swagger. Esta herramienta proporciona una interfaz intuitiva que permite visualizar y probar los endpoints de la API directamente en el navegador, mejorando la experiencia del desarrollador. Swagger permite generar documentación interactiva y actualizada automáticamente a partir del código fuente, asegurando el acceso a una referencia precisa y actualizada.

Despliegue: se ha decidido por Docker. Esta tecnología permite empaquetar la aplicación y todas sus dependencias en contenedores que pueden ejecutarse de manera consistente en cualquier entorno, eliminando problemas de configuración. Docker simplifica la

configuración y el despliegue de aplicaciones, permitiendo una integración continua y despliegue continuo (CI/CD) más eficiente.

MODELO DE DATOS

ESQUEMA E-R



DESCRIPCIÓN ENTIDADES Y CAMPOS

El modelo de datos de una aplicación web es fundamental para su funcionamiento y estructura subyacente. En el caso de Book 'N Book, un gestor de lectura y red social para amantes de los libros, el diseño se visualiza como crucial a la hora organizar y gestionar eficientemente la información relacionada con libros, autores, usuarios y otras entidades

clave. En esta sección, presentaremos una descripción detallada de las entidades y campos que conforman el modelo, brindando una visión integral de cómo se estructura y almacena la información en la plataforma.

1. Autor:
    - Descripción: Almacena información sobre los autores de los libros.
    - Campos:
        id (Identificador único del autor)
        pseudonimo (Pseudónimo del autor)
        biografia (Breve biografía del autor)
        localidad (Localidad del autor)
        foto_autor (Foto del autor)
2. Denuncia:
    - Descripción: Registra denuncias realizadas por los usuarios sobre contenido inapropiado o problemático expuesto en comentarios/reseñas por otros usuarios.
    - Campos:
        id (Identificador único de la denuncia)
        fecha (Fecha en que se realizó la denuncia)
        motivo (Motivo de la denuncia)
        texto (Descripción detallada de la denuncia)
3. Género:
    - Descripción: Contiene información sobre los géneros y/o tipos de libros.
    - Campos:
        id (Identificador único del género)
        nombre (Nombre del género)
        tipo (Tipo como ficción, no ficción, ... // Género romántico, humor, ...)
4. Saga:
    - Descripción: Almacena información sobre las sagas de libros.
    - Campos:

id (Identificador único de la saga)
nombre (Nombre de la saga)

5. Usuario:
    - Descripción: Contiene detalles sobre los usuarios registrados en la plataforma.
    - Campos:
        id (Identificador único del usuario)
        nombre_usuario (Nombre de usuario único para identificación)
        nombre (Nombre del usuario)
        apellido1 (Primer apellido del usuario)
        apellido2 (Segundo apellido del usuario)
        correo (Correo electrónico del usuario)
        password (Contraseña del usuario)
        rol (Rol del usuario en la plataforma)
        foto_perfil (Foto de perfil del usuario)
6. Grupo:
    - Descripción: Representa clubes de lectura formados por usuarios.
    - Campos:
        id (Identificador único del grupo)
        nombre (Nombre del grupo)
        descripcion (Descripción del grupo)
        id_genero (ID del género asociado al grupo)
        id_tipo (ID del tipo de género asociado al grupo)
7. Libro:
    - Descripción: Contiene información detallada sobre los libros.
    - Campos:
        id (Identificador único del libro)
        nombre (Nombre del libro)
        descripcion (Descripción del libro)
        id_autor (ID del autor del libro)
        fecha_publicacion (Fecha de publicación del libro)

foto_libro (Foto de portada del libro)
pag_total (Número total de páginas del libro)

id_genero (ID del género del libro)
id_saga (ID de la saga a la que pertenece el libro, si aplica)
id_tipo (ID del tipo de género del libro)

Por otro lado, las relaciones entre las tablas son fundamentales para establecer la conexión y la coherencia entre los diferentes conjuntos de datos. En esta sección, se explica en detalle las relaciones entre las tablas en el modelo de datos destacando cómo cada entidad se entrelaza con otras para proporcionar una experiencia integral y fluida para los usuarios de la plataforma.

1. Autor y Libro:
    - Relación: Un autor puede haber escrito varios libros, pero un libro solo puede tener un autor.
    - Implementación: En la tabla libro, el campo id_autor establece una relación con la tabla autor (clave externa).
2. Género/Tipo y Libro:
    - Relación: Un libro puede pertenecer a un género, pero un género puede tener varios libros asociados.
    - Implementación: En la tabla libro, el campo id_genero e id_tipo establece una relación con la tabla genero (clave externa).
3. Saga y Libro:
    - Relación: Un libro puede pertenecer a una saga, pero una saga puede contener varios libros.
    - Implementación: En la tabla libro, el campo id_saga establece una relación con la tabla saga (clave externa).
4. Usuario y Grupo:
    - Relación: Un usuario puede pertenecer a varios grupos, y un grupo puede tener varios usuarios.

- Implementación: La tabla usuario_grupo establece esta relación. Los campos id_grupo e id_usuario forman una clave compuesta que vincula los usuarios a los grupos a los que pertenecen.

5. Usuario y Seguimiento:
   - Relación: Un usuario puede seguir a otros usuarios, creando una relación de seguimiento.
   - Implementación: La tabla seguimiento registra esta relación, donde los campos id_seguido e id_seguidor indican quién sigue a quién.
6. Usuario y Preferencia de Género:
   - Relación: Un usuario puede tener varias preferencias de género, y un género puede ser preferido por varios usuarios.
   - Implementación: La tabla preferencia_usuario establece esta relación, donde los campos id_usuario e id_genero forman una clave compuesta que vincula los usuarios con sus preferencias de género.
7. Libro y Grupo:
   - Relación: Un libro puede estar asociado a varios grupos, y un grupo puede tener varios libros asociados.
   - Implementación: La tabla libro_grupo registra esta relación, donde los campos id_grupo e id_libro forman una clave compuesta que vincula los libros con los grupos a los que pertenecen.

PROTOTIPO DE LA APLICACIÓN WEB

En este apartado se presenta una visión detallada de la interfaz de usuario diseñada para *Book 'N Book* utilizando la herramienta de prototipado Figma. Este prototipo captura las diversas características y funcionalidades de la aplicación, desde la gestión de lecturas hasta la formación de clubes de lectura, las estadísticas de lectura y panel de administrador de la aplicación. A través de este enlace será posible visitar el Figma específico de *Book 'N Book*.

DEFINICIÓN API REST

En este apartado se detalla la API REST de *Book N' Book*. Para ello se ha utilizado Swagger en la documentación y prueba de la API de manera interactiva. A continuación, se describen los pasos para acceder mediante Swagger, así como una tabla con los detalles de los endpoints disponibles:

ACCESO A LA DOCUMENTACIÓN CON SWAGGER

Crear la base de datos de MySQL:

```
create database booknbook
```

Crear en la ruta principal de la aplicación Spring Boot un archivo env en el cual insertar usuario, contraseña y enlace para la conexión de la API a la base de datos:

```
DATABASE_PASSWORD=
DATABASE_URL=
DATABASE_USERNAME=
```

Iniciar el servidor de la aplicación Spring Boot:

```
mvn spring-boot:run
```

Abrir el navegador e ingresar en la ruta:

```
http://localhost:8080/swagger
```

En la interfaz de Swagger se puede ver una lista de todos los endpoints disponibles, junto con sus métodos HTTP, parámetros, y descripciones. Swagger ofrece una manera intuitiva y visual de explorar la API, permitiéndote entender rápidamente cómo interactuar con cada endpoint. Además de simplemente visualizar la documentación, permite probar los endpoints directamente desde la interfaz, lo cual es extremadamente útil para verificar el funcionamiento de la API sin necesidad de utilizar herramientas externas. De tal modo, es posible enviar solicitudes a los endpoints, observar las respuestas en tiempo real y ajustar los parámetros para ver cómo afectan a las respuestas.

TABLA DE ENDPOINTS

A continuación, en esta tabla se muestran los detalles de los endpoints de la API REST, incluyendo el método HTTP, los parámetros y una breve descripción de cada uno.

Endpoint	Método	Parámetros	Descripción
Métodos de uso público			

/register	POST	body: {
		"usuario": "string",
		"nombre": "string",
		"apellidoPrimero":"string",
		"apellidoSegundo": "string",
		"idGenero": 0,
		"idTipo": 0,
		"email": "string",
		"password": "string"
		}
			Crear nuevo usuario

/login	POST	body: {
		"username": "string",
		"password": "string"
		}
			Iniciar sesión con usuario registrado

/contador	GET		Obtener estadísticas generales de la aplicación
/libros	GET	Parameters: pageIndex: integer, size: integer, genero: string filter: string	Obtener los libros según el género de los mismos
/libros-recomendados	GET	Parameters: pageIndex: integer, size: integer, genero: string	Obtener listado libros recomendados
/libros-propuestas	GET	Parameters: size: integer, genero: string	Obtener listado libros aleatorios
/libros-novedades	GET	Parameters: pageIndex: integer, size: integer, genero: string	Obtener los libros novedosos
/libros-mas-leidos	GET	Parameters: pageIndex: integer, size: integer, genero: string	Obtener los libros más leídos
/libro/{idLibro}	GET	Parameters: idLibro: integer	Obtener la ficha de un libro

/clubes	GET		Obtener listado de clubes
Métodos de autor			
/autor/{idAutor}	GET	Parameters: idAutor: integer	Obtener los datos del perfil del autor
/autor/{idAutor}/libros	GET	Parameters: idAutor: integer	Obtener listado de los libros de un autor
Métodos de libro			

/api/libro/puntuar	PUT	body: {
		"idLibro": 0,
		"puntuacion": 0,
		"comentario": "string"
		}
			Puntuar un libro

/api/libro/{idLibro}/valoracion/{estado}	POST	Parameters: idLibro: integer, estado: string	Vincular un libro con el usuario

api/libro/valoracion	POST	body:{
		"estado": "string",
		"paginaActual": 0,
		"calificacionPersonal": 0,
		"comentario": "string",
		"fechaComentario":
		"2024-05-31",
		"fechaLectura": "2024-05-31",
		"idLibro": 0
		}
			Actualizar la valoración de un usuario sobre un libro

Métodos de clubes			

/api/grupo/{idGrupo}/imagen	PUT	Parameters:
		idGrupo: integer
		body: {
		"imagen": "string"
		}
			Poner imagen al grupo

/api/grupo	POST	body: {
		"nombreGrupo": "string",
		"genero": 0,
		"tipo": 0,
		"descripcion": "string"
		}
			Crear un nuevo grupo

/api/grupo/mis-clubes	GET	Parameters: type: string, pageIndex: integer, size: integer, filter: string	Obtener los clubes que el usuario tiene (pertenece/administra)
/api/grupo/clubes	GET	Parameters: pageIndex: integer, size: integer, filter: string	Obtener listado de clubes
/api/grupo/{i	DELETE	Parameters:	Borrar el club

dGrupo}		idGrupo: integer	seleccionado
/api/grupo/{idGrupo}/self/{accion}	DELETE	Parameters: idGrupo: integer, accion: P/A	Pertenecer(P) o abandonar(A) un club
Métodos de devolución de combos (para formularios)			
/combo/saga/{idAutor}	GET	Parameters: idAutor: integer,	Obtener combo de sagas del autor
/combo/genero	GET		Obtener combo de género y tipo
/combo/denuncia/motivo	GET		Obtener combo de los motivos de denuncia
/combo/denuncia/grupo/comentario/estado	GET		Obtener combo de estados de denuncia de un comentario en grupo
/combo/denuncia/comentario/estado	GET		Obtener combo de estados de denuncia de un comentario
/combo/autor	GET		Obtener combo de autores
Métodos de usuarios			

/api/user/perfil	PUT	body: {
		"nombre": "string",
		"apellidoPrimero": "string",
		"apellidoSegundo": "string",
		"idGenero": 0,
		"idTipo": 0,
		"email": "string",
		"password": "string"
		}
			Actualizar el perfil del usuario logueado

/api/user/imagen	PUT	body: {
		"imagen": "string",
		}
			Poner imagen al usuario

/api/user/usuario/{username}/unfollow	POST	Parameters: username: string	Dejar de seguir a otro usuario
/api/user/usuario/{username}/follow	POST	Parameters: username: string	Seguir a otro usuario
/api/user/{username}/perfil	GET	Parameters: username: string	Obtener el perfil del usuario
/api/user/{username}/perfil/valoracion	GET	Parameters: username: string	Obtener los comentarios que ha dejado un usuario en las diferentes lecturas

/api/user/{username}/perfil/libros	GET	Parameters: username: string	Obtener los libros favoritos de un usuario en el perfil
/api/user/libros/{estado}	GET	Parameters: pageIndex: integer, size: integer, filter: estado	Obtener listado de los libros del usuario según el estado (Leído, Favorito, Progreso)
/api/user/selff/desactivacion	DELETE		Desactivar la cuenta
Métodos de administrador			

/admin/usuario/rol	PUT	body; {
		"username": "string",
		"rol": "string"
		}
			Cambiar el rol a un usuario

/admin/libro	PUT	body: {
		"id": 0,
		"nombre": "string",
		"descripcion": "string",
		"fechaPublicacion": "2024-05-31",
		"estado": "string",
		"pagTotal": 0,
		"autor": 0,
		"genero": 0,
		"tipo": 0,
		"saga": 0
		}
			Actualizar un libro

/admin/libro	POST	body: {
		"descripcion": "string",
		"fechaPublicacion": "2024-05-31",
		"nombre": "string",
		"paginas": 0,
		"idAutor": 0,
		"genero": 0,
		"tipo": 0,
		"saga": 0,
		"nuevaSaga": "string"
		}
			Crear un nuevo libro

/admin/libro/{idLibro}/imagen	PUT	Parameters:
		idLibro: integer
		body: {
		"imagen": "string",
		}
			Poner imagen al luiçirbo

/admin/denuncia/{idDenuncia}/{estado}	PUT	Parameters: idDenuncia: integer, estado: string	Establecer un estado a la denuncia
/admin/autor	PUT	body: { "id": 0, "pseudonimo": "string",	Actualizar un autor

		"localidad": "string", "biografia": "string" }	

/admin/autor	POST	body: {
		"pseudonimo": "string",
		"localidad": "string",
		"biografia": "string"
		}
			Crear un nuevo autor

/admin/autor/{idAutor}/imagen	PUT	Parameters:
		idAutor: integer
		body: {
		"imagen": "string",
		}
			Poner imagen al autor

/admin/usuario/lista	GET	Parameters: username: string, pageIndex: integer, size: integer	Obtener listado usuarios de la aplicación
/admin/libro/gestion	GET	Parameters: filtro: string, pageIndex: integer, size: integer	Obtener listado libros para su gestión
/admin/denuncia/valoracion/mensaje	GET	Parameters: idLibro: integer, idUsuario: integer	Obtener la valoración denunciada (comentario específico)
/admin/comentarios/denuncia	GET	Parameters: filter: string, estado: string, pageIndex: integer, size: integer	Obtener listado de comentarios denuncias por el estado
/admin/{username}/desactivacion	DELETE	Parameters: username: string	Deshabilitar un usuario
/admin/libro/{idLibro}/desactivacion	DELETE	Parameters: idLibro: integer	Eliminar un libro
Métodos de denuncia			
/api/denuncia	PUT		Denunciar una valoración/comentario

MANUAL DE DESPLIEGUE

En este apartado se describen todos los pasos necesarios para desplegar la aplicación *Book N' Book* utilizando Docker. La aplicación está compuesta por tres componentes principales: una base de datos MySQL, un backend desarrollado en Spring Boot y un frontend en

Angular. El despliegue se realiza utilizando contenedores Docker y Docker Compose.

PRE-REQUISITOS

Antes de comenzar, asegúrate de tener instalados los siguientes componentes en tu máquina:

- Docker
- Docker Compose

ESTRUCTURA DE PROYECTO

La estructura del proyecto debe ser similar a la siguiente:

```
booknbook/
├── BooknBookBACK/
│   ├── Dockerfile
│   ├── target/
│   │   └── booknbook-0.0.1-SNAPSHOT.jar
│   └── ...
├── BookNBookFRONT/
│   ├── BooknBookFront/
│   │   ├── Dockerfile
│   │   ├── nginx.conf
│   │   └── ...
├── docker-compose.yml
└── init.sql
```

EXPLICACIÓN DE LOS ARCHIVOS DE DESPLIEGUE

Dockerfile del Frontend: se divide en dos etapas para optimizar la construcción y despliegue de la aplicación Angular. En la primera etapa, se utiliza una imagen ligera de Node.js para construir la aplicación. Esto incluye la instalación de dependencias y la construcción del proyecto en un directorio de trabajo específico. En la segunda etapa, se utiliza una imagen de Nginx para servir la aplicación construida. Los archivos estáticos generados se copian al directorio de Nginx y se configura el servidor web para servirlos. Esta separación en

etapas permite mantener la imagen final ligera y eficiente, ya que solo incluye los archivos necesarios para servir la aplicación.

Dockerfile del Backend: define una configuración para ejecutar una aplicación Java utilizando OpenJDK 17. Se establecen variables de entorno para la configuración de la base de datos, incluyendo la URL, el nombre de usuario y la contraseña. El archivo JAR de la aplicación, generado previamente con Maven, se copia al contenedor. El comando de entrada especifica que la aplicación se ejecutará utilizando java -jar para iniciar el archivo JAR. Este enfoque simplifica el despliegue de la aplicación Java, asegurando que todas las dependencias y configuraciones necesarias estén contenidas dentro de la imagen del contenedor.

Archivo docker-compose.yml: agrupa el despliegue de múltiples servicios de contenedores necesarios para la aplicación. Define tres servicios principales: el frontend, el backend y la base de datos MySQL. Cada servicio tiene su propia configuración de construcción, puertos expuestos y dependencias. Por ejemplo, el servicio del frontend depende del backend para garantizar que el servidor de la aplicación esté listo antes de iniciar el servidor web. La configuración de MySQL incluye variables de entorno para la creación y autenticación de la base de datos, así como un health check para asegurar que el servicio de base de datos esté saludable antes de permitir que los demás servicios se inicien. Este enfoque asegura una coordinación adecuada entre los distintos componentes de la aplicación, facilitando un despliegue eficiente y ordenado.

PASOS PARA EL DESPLIEGUE

Para desplegar la aplicación, desde una terminal en el directorio raíz del proyecto (booknbook) y ejecutar los siguientes comandos:

```
docker-compose down
docker-compose build --no-cache
docker-compose up
```

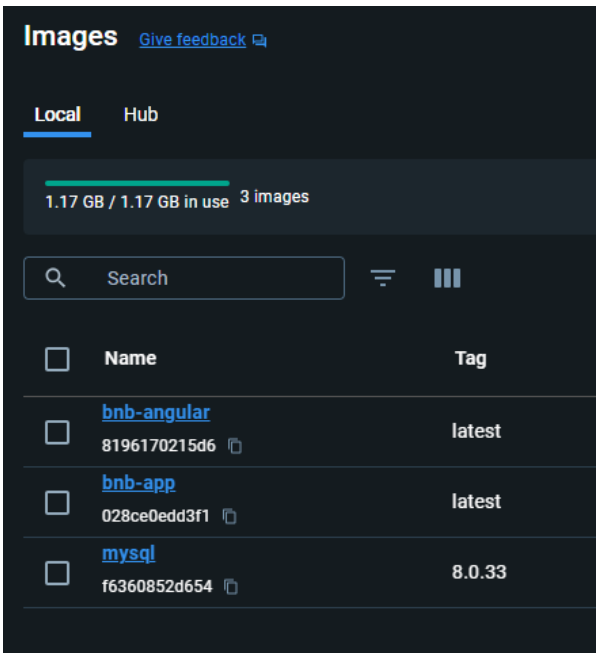
```
PS C:\Users\maria\Desktop\BNB> docker-compose down
[+] Running 4/4
 ✔Container frontcontainer   Removed
 ✔Container backcontainer    Removed
 ✔Container mysqlcontainer   Removed
 ✔Network bnb_default        Removed
```

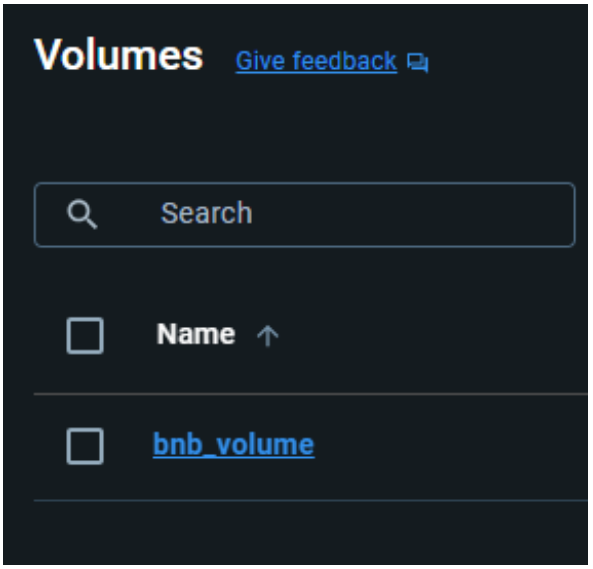
```
PS C:\Users\maria\Desktop\BNB> docker-compose build --no-cache
[+] Building 4.8s (8/8)                                                                                  docker:default
 => [app internal] load build definition from Dockerfile                                                          0.0s
[+] Building 5.1s (8/8)                                                                                  docker:default
 => [app internal] load build definition from Dockerfile                                                          0.0s
[+] Building 280.8s (24/24) FINISHED                                                                     docker:default
 => [app internal] load build definition from Dockerfile                                                          0.0s
 => => transferring dockerfile: 349B                                                                              0.0s
 => [app internal] load metadata for docker.io/library/openjdk:17                                                 1.2s
 => [app auth] library/openjdk:pull token for registry-1.docker.io                                                0.0s
 => [app internal] load .dockerignore                                                                             0.0s
 => => transferring context: 2B                                                                                   0.0s
 => [app internal] load build context                                                                             2.6s
 => => transferring context: 55.52MB                                                                              2.5s
 => CACHED [app 1/2] FROM docker.io/library/openjdk:17@sha256:528707081fdb9562eb819128a9f85ae7fe000e2fbaeaf9f8766  0.0s
 => [app 2/2] COPY target/booknbook-0.0.1-SNAPSHOT.jar app.jar                                                    0.5s
 => [app] exporting to image                                                                                      0.2se
 => => exporting layers                                                                                           0.2s
 => => writing image sha256:028ce0edd3f1d5f13845f0b38269aea7f58c15d563aecc92725fce6eb76f9441                      0.0se
 => => naming to docker.io/library/bnb-app                                                                        0.0s
 => [angular internal] load build definition from dockerfile                                                      0.0s
 => => transferring dockerfile: 312B                                                                              0.0s
 => [angular internal] load metadata for docker.io/library/nginx:alpine                                           1.0s
 => [angular internal] load metadata for docker.io/library/node:18-alpine                                         1.0s
 => [angular auth] library/nginx:pull token for registry-1.docker.io                                              0.0s
 => [angular auth] library/node:pull token for registry-1.docker.io                                               0.0s
 => [angular internal] load .dockerignore                                                                         0.0s
 => => transferring context: 52B                                                                                  0.0s
 => [angular build 1/5] FROM docker.io/library/node:18-alpine@sha256:cf350f8bb497d824f71f1f735df5d6d3321138be3b9f7  0.0s
 => CACHED [angular stage-1 1/3] FROM docker.io/library/nginx:alpine@sha256:69f8c2c72671490607f52122be2af27d4fc09  0.0s
 => [angular internal] load build context                                                                       134.1s
 => => transferring context: 1.91GB                                                                             134.1s
 => CACHED [angular build 2/5] WORKDIR /app                                                                       0.0s
 => [angular build 3/5] COPY . .                                                                                 36.6s
 => [angular build 4/5] RUN npm install                                                                          43.4s
 => [angular build 5/5] RUN npm run build                                                                        58.5s
 => [angular stage-1 2/3] COPY --from=build /app/dist/BooknBookFront/ /usr/share/nginx/html                       0.4s
 => [angular stage-1 3/3] COPY nginx.conf /etc/nginx/conf.d/default.conf                                          0.2s
 => [angular] exporting to image                                                                                  0.9s
 => => exporting layers                                                                                           0.8s
 => => writing image sha256:8196170215d6a3846f4a29bc50e5997c630722d0d3b872950604 5a657115688f                     0.0s
 => => naming to docker.io/library/bnb-angular                                                                    0.0s
```

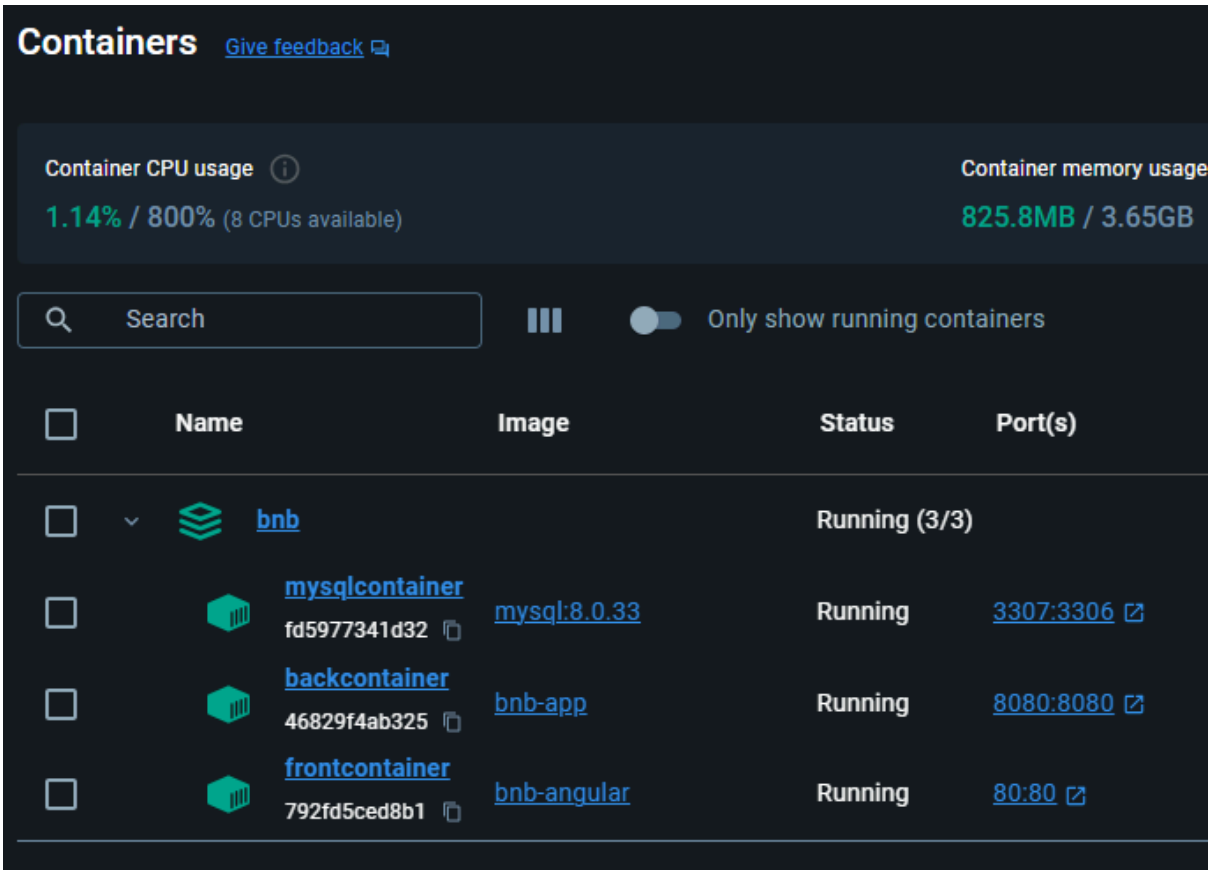
```
PS C:\Users\maria\Desktop\BNB> docker-compose up
[+] Running 4/4
 ✔Network bnb_default        Created
 ✔Container mysqlcontainer   Created
 ✔Container backcontainer    Created
 ✔Container frontcontainer   Created
```

Estos comandos harán lo siguiente:

- docker-compose down: Detiene y elimina los contenedores existentes.
- docker-compose build --no-cache: Construye los contenedores sin utilizar la caché, asegurándose de que se utilicen las versiones más recientes de las imágenes.
- docker-compose up: Inicia los contenedores en modo adjunto, permitiendo ver los logs en tiempo real.



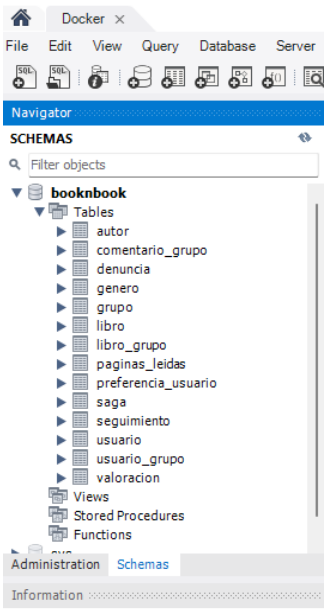


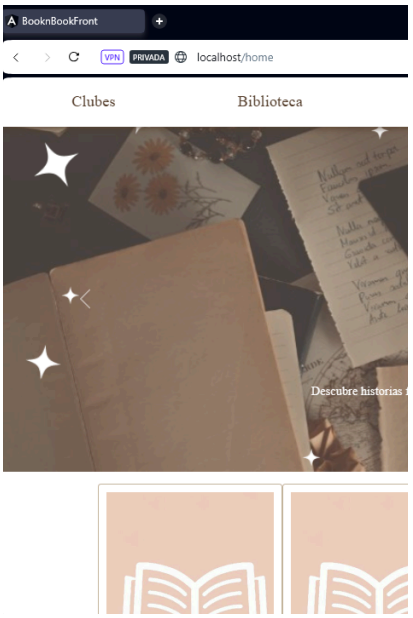


Tras ello, se puede verificar el despliegue de la siguiente forma:

- Backend: Acceder a http://localhost:8080/swagger para verificar que el backend de Spring Boot está corriendo correctamente.

- Frontend: Acceder a http://localhost para verificar que el frontend de Angular está corriendo correctamente.
- Base de Datos: Acceder a MySQL a través del puerto 3307 en tu máquina local con el usuario root y la contraseña 123456.





Por último a tener en cuenta, en este despliegue, se ha configurado un volumen para la base de datos MySQL, lo que permite la persistencia de datos incluso si los contenedores se detienen o reinician. Este volumen se define en el archivo docker-compose.yml y asegura que los datos de la aplicación no se pierdan, proporcionando una solución de almacenamiento confiable y persistente.
Además, el frontend de Angular se sirve utilizando Nginx, un servidor web ligero y de alto rendimiento. Nginx se utiliza para servir los archivos estáticos generados por Angular, asegurando una entrega rápida y eficiente de los recursos del frontend. La configuración de Nginx se especifica en el archivo nginx.conf, optimizando así la experiencia del usuario final al interactuar con la aplicación *Book N' Book*.

POSTMORTEN Y CONCLUSIONES

La ejecución del proyecto *Book N' Book* ha sido un desafío significativo. Durante el desarrollo, uno de los principales obstáculos ha sido la integración de la seguridad en la API mediante el uso de tokens, lo cual

resultó ser una tarea frustrante debido a la complejidad y los problemas técnicos encontrados. Además, la ambiciosa propuesta inicial del proyecto requería un alcance amplio de funcionalidades, lo que llevó a tomar la decisión de reducir algunas de ellas para asegurar que las implementadas se completaran con un alto nivel de calidad, una decisión que ha sido crucial para una ejecución completa.

Otro aspecto que ha complicado el desarrollo fue el manejo de la base de datos mediante JPA. La base de datos, con una cantidad elevada de conexiones entre tablas, hizo que las llamadas y consultas fueran complejas de implementar y optimizar. Esto añadió una capa adicional de dificultad al proyecto, especialmente en lo que respecta a mantener un rendimiento aceptable y evitar problemas de eficiencia.

A pesar de las dificultades enfrentadas, se han identificado varias áreas de mejora que pueden considerarse en futuras iteraciones del proyecto para ampliar su funcionalidad y mejorar la experiencia del usuario (descritas en el segundo apartado de la documentación).

Por último, considerando la ejecución del proyecto y las mejoras propuestas, la viabilidad de poner en marcha *Book N' Book* en un entorno real es alta, asimismo las futuras mejoras pueden ser integradas gradualmente, permitiendo que el proyecto evolucione y se adapte a las necesidades de los usuarios.

Book N' Book se posiciona como una aplicación competitiva en el mercado. Comparado con otras alternativas existentes para la gestión de lecturas y la conexión entre lectores, esta aplicación ofrece una experiencia integral que combina la gestión con la formación de clubes de lectura y estadísticas detalladas sobre los hábitos de lectura de los usuarios. Su enfoque en la interactividad y la comunidad virtual literaria le permite diferenciarse de otras opciones en el mercado.

En conclusión, a pesar de los desafíos encontrados durante el desarrollo, *Book N' Book* ha logrado establecer una base sólida sobre la cual se puede construir y expandir. Las posibles mejoras identificadas ofrecen una hoja de ruta clara para el futuro, asegurando que la aplicación pueda ofrecer un valor añadido significativo a sus usuarios y mantenerse competitiva en el mercado.