Documentación proyecto API Rest:

Catálogo Musical

Realizado por: Asunción de los Ángeles Naranjo Rodríguez

Fecha de realización: 17 de Febrero del 2025

Profesora: Soraya Peceño Capilla

Curso: 2ºDam

Índice.

1. Documentación: objetivos, requisitos y UML.

1.1. Definición del proyecto y objetivos.

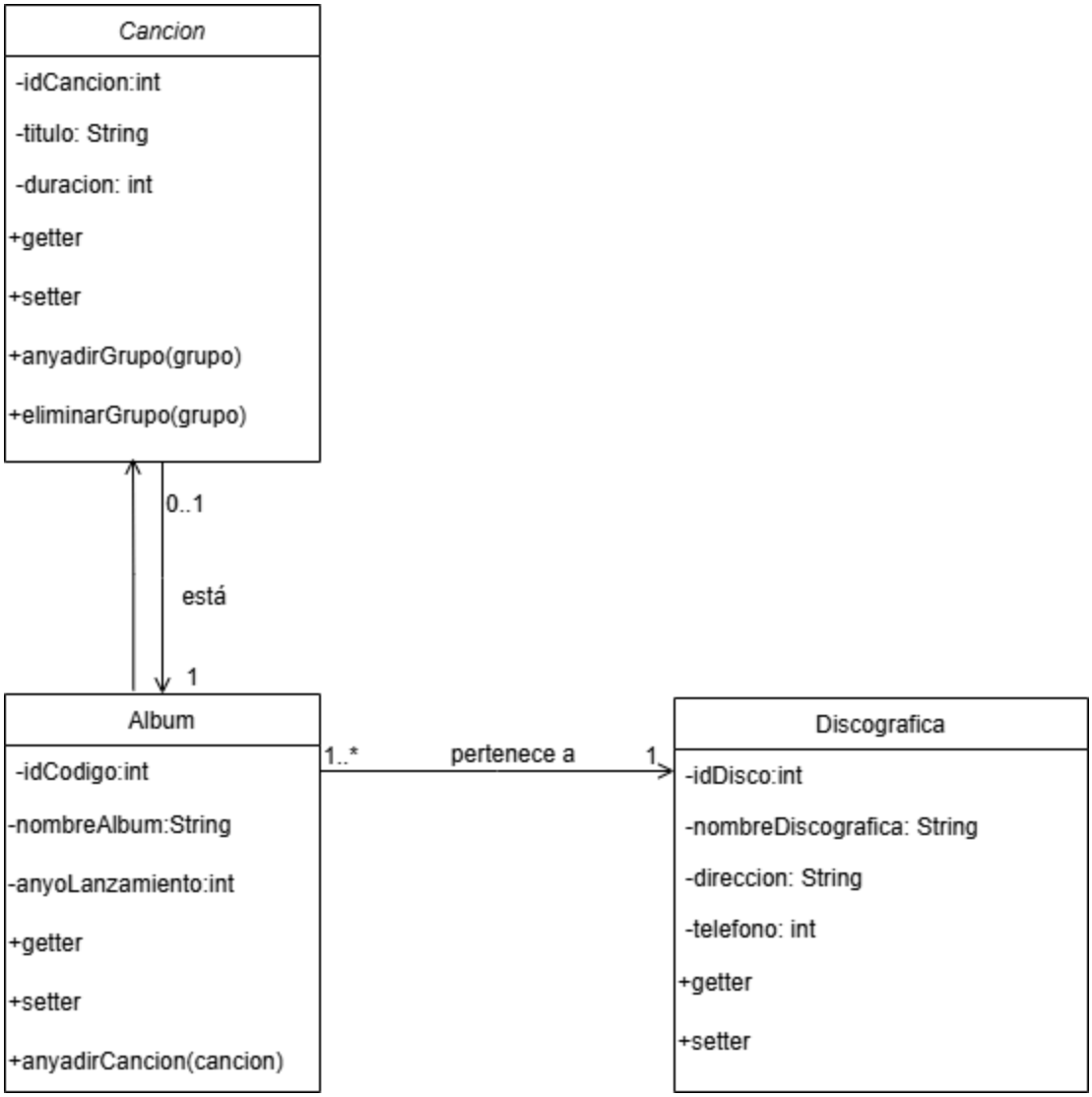
Desarrollaremos un sistema de gestión de música para los usuarios de una aplicación utilizando SpringBoot como framework , MySql como base de datos y Thymeleaf como plantilla para generar vistas en aplicaciones web.

El objetivo es que los usuarios de la aplicación tengan acceso a un catálogo de música donde puedan consultar, añadir o actualizar información de sus canciones, discográficas y álbumes favoritos.

1.2. Requisitos.

- La aplicación tiene que permitir a los usuarios poder iniciar sesión con su usuario y contraseña.
- La aplicación debe poder mostrar a los usuarios información sobre las canciones que escuchan.
- La aplicación debe informar a los usuarios sobre los álbumes en los que se encuentran las canciones que escuchan.
- La aplicación debe ofrecer información sobre la discográfica a la que pertenecen los álbumes.
- La aplicación permite introducir información nueva (dar de alta nuevos álbumes, discográficas y canciones) o modificar una parte de la existente (alterar algunos campos de álbumes, discográficas o canciones).

1.3. UML.



2. Arquitectura del proyecto.

La aplicación que estamos diseñando sigue el patrón Modelo-Vista-Controlador (MVC). Este patrón se basa en separar la lógica de negocio (Modelo), la presentación (Vista) y la interacción con el usuario (Controlador) para facilitar el mantenimiento y escalabilidad de la aplicación.

En nuestro proyecto tenemos la siguiente estructura de carpetas:

- **Modelos:** en este paquete se incluyen las entidades o modelos de datos que va a tener nuestra aplicación. Estas clases se mapean con las tablas de nuestra base de datos.

- **Repositorios:** este paquete se encarga de gestionar la interacción con la base de datos. En él se crean las interfaces que extienden de JpaRepository o CrudRepository.

- **Servicios:** paquete que incluye toda la lógica de negocio de la aplicación. Coordinan las operaciones entre el modelo, repositorio o contiene lógica compleja que no puede realizarse en otros paquetes.

- **Controladores:** este paquete contiene las clases que gestionan las solicitudes HTTP y devuelven una respuesta al usuarios que son quienes realizan las peticiones.

- **Excepciones:** paquete que contiene las clases que se encargan de gestionar los errores que puedan ocurrir durante el flujo de la aplicación.

3. Diseño de servicios. Catálogo de Servicios Web: Restful.

Los principales servicios que posee nuestra aplicación son:

- Servicio que da de alta a un usuario.

Alta usuario	
Ruta URL	http://localhost:9090/usuario/autenticacion/nuevousuario
Método	Post
Entrada	@RequestBody Usuario user
Modo de acceso	Postman
Formato de Respuesta	String
Código Respuesta	-

Descripción	Registramos un nuevo usuario en el sistema. Devuelve un String donde nos informa si el usuario se ha registrado con éxito.

- Servicio que me permite loguearme.

Login	
Ruta URL	http://localhost:9090/usuario/autenticacion/login
Método	Post
Entrada	@RequestBody Usuario usuario
Modo de acceso	Postman
Formato de Respuesta	String
Código Respuesta	-
Descripción	El usuario se loguea en el sistema. Nos devuelve un token que utilizamos en el resto de servicios para acceder a ellos con una autorización.

- Servicios que permiten crear una canción, un álbum y una discográfica.

Crear álbum	
Ruta URL	http://localhost:9090/album/altaAlbum
Método	Post
Entrada	@RequestBody Album album
Modo de acceso	Postman
Formato de Respuesta	ResponseEntity <Album>
Código Respuesta	CREATED -> HttpStatus.CREATED
Descripción	Este método permite crear una álbum llamando al método altaAlbum del servicio. Devuelve un JSON con todos los campos que posee álbum.

Crear discográfica	
Ruta URL	http://localhost:9090/discografica/altaDiscografica
Método	Post
Entrada	@RequestBody Discografica discografica
Modo de acceso	Postman
Formato de Respuesta	ResponseEntity <Discografica>
Código Respuesta	CREATED -> HttpStatus.CREATED
Descripción	Este método permite crear una discográfica llamando al método altaDiscografica del servicio. Devuelve un JSON con todos los campos que posee discográfica.

Crear canción	
Ruta URL	http://localhost:9090/cancion/altaCancion
Método	Post
Entrada	@RequestBody Cancion cancion
Modo de acceso	Postman
Formato de Respuesta	ResponseEntity <Cancion>
Código Respuesta	CREATED -> HttpStatus.CREATED
Descripción	Este método permite crear una canción llamando al método altaCancion del servicio. Devuelve un JSON con todos los campos que posee canción.

- Servicio que permite consultar el listado de las canciones, de los álbumes y de las discográficas.

Consultar listado de discográficas

Ruta URL	http://localhost:9090/discografica/listaDiscografica
Método	Get
Entrada	Model model
Modo de acceso	Navegador
Formato de Respuesta	HTML listaDiscografica.html
Código Respuesta	-
Descripción	Este método muestra una lista de discográficas llamando al servicio mostrarListaDiscografica. Devuelve un HTML con los campos de la discográfica que queremos mostrar.

Consultar listado de canciones	
Ruta URL	http://localhost:9090/cancion/listaCancion
Método	Get
Entrada	Model model
Modo de acceso	Navegador
Formato de Respuesta	HTML listaCancion.html
Código Respuesta	-
Descripción	Este método muestra una lista de canciones llamando al servicio mostrarCanciones. Devuelve un HTML con los campos de la canción que queremos mostrar.

Consultar listado de álbumes	
Ruta URL	http://localhost:9090/album/listaAlbum
Método	Get
Entrada	Model model

Modo de acceso	Navegador
Formato de Respuesta	HTML listaAlbum.html
Código Respuesta	-
Descripción	Este método muestra una lista de álbumes llamando al servicio mostrarAlbumes. Devuelve un HTML con los campos de los álbumes que queremos mostrar.

- Servicio que permite consultar el listado de discográficas por el nombre.

Consultar listado de discográficas por el nombre	
Ruta URL	http://localhost:9090/discografica/listaDiscograficaNombre/Sony Music
Método	Get
Entrada	@PathVariable String nombre, Model model
Modo de acceso	Navegador
Formato de Respuesta	HTML listaDiscograficaNombre.html
Código Respuesta	-
Descripción	Este método muestra una lista de discográficas filtradas por el nombre que le pasemos. Devuelve un HTML con el nombre, dirección y teléfono de la discográfica

- Servicio con el que podemos consultar el listado de canciones por el título.

Consultar listado de canciones por el título	
Ruta URL	http://localhost:9090/cancion/listaCancionTitulo/Copenhague
Método	Get
Entrada	@PathVariable String titulo, Model model
Modo de acceso	Navegador

Formato de Respuesta	HTML listaCancionTitulo.html
Código Respuesta	-
Descripción	Este método muestra una lista de canciones filtradas por el título que le pasemos. Devuelve un HTML con la información consultada.

- Servicio con el que podemos consultar el listado de álbumes según su año de lanzamiento.

Consultar listado de álbumes según el año de lanzamiento	
Ruta URL	http://localhost:9090/album/listaAlbumAnyo/2016
Método	Get
Entrada	@PathVariable Integer anyo, Model model
Modo de acceso	Navegador
Formato de Respuesta	HTML listaAlbumAnyo.html
Código Respuesta	-
Descripción	Este método muestra una lista de álbumes filtrados por el año en que fueron lanzados. Devuelve un HTML con la información consultada.

- Servicio que nos permite mostrar una discográfica según su identificador.

Mostrar discográfica según su id	
Ruta URL	http://localhost:9090/discografica/4
Método	Get
Entrada	@PathVariable Integer idDiscografica
Modo de acceso	Postman
Formato de Respuesta	ResponseEntity<Discografica>

Código Respuesta	ResponseEntity.ok DiscograficaNotFound -> Http.Status.NOT_FOUND error.html
Descripción	Este método muestra una discográfica en función del id que le pasemos. Devuelve un JSON donde se muestra la discográfica que hemos buscado. Si no encuentra la discográfica lanza el error DiscograficaNotFound y redirige a error.html con código 400.

- Servicio que nos muestra una canción por su título.

Mostrar canción por su título	
Ruta URL	http://localhost:9090/cancion/cancionTitulo/Todo lo que importa
Método	Get
Entrada	@PathVariable String titulo
Modo de acceso	Postman
Formato de Respuesta	ResponseEntity<Cancion>
Código Respuesta	ResponseEntity.ok
Descripción	Este método muestra una canción en función del título que le pasemos. Devuelve un JSON donde se muestra la canción que hemos buscado.

- Servicio que nos muestra un álbum por su nombre.

Mostrar álbum por nombre	
Ruta URL	http://localhost:9090/album/albumTitulo/Polvora
Método	Get
Entrada	@PathVariable String nombre
Modo de acceso	Postman

Formato de Respuesta	ResponseEntity<Album>
Código Respuesta	ResponseEntity.ok
Descripción	Este método muestra un álbum en función del nombre que le pasemos. Devuelve un JSON donde se muestra el álbum que hemos buscado.

- Servicio que actualiza el campo teléfono de una discográfica.

Actualizar discográfica	
Ruta URL	http://localhost:9090/discografica/1/897223114
Método	Put
Entrada	PathVariable Integer idDiscografica, @PathVariable String telefono
Modo de acceso	Postman
Formato de Respuesta	ResponseEntity<Discografica>
Código Respuesta	Ok ->HttpStatus.OK(200) DiscograficaNotFound -> Http.Status.NOT_FOUND error.html
Descripción	Este método permite actualizar el campo teléfono de la discográfica. Devuelve un JSON donde se muestra el campo actualizado. Si no encuentra la discográfica lanza el error DiscograficaNotFound y redirige a error.html con código 400.

- Servicio que actualiza el campo duración de una canción.

Actualizar duración	
Ruta URL	http://localhost:9090/cancion/2/345
Método	Put
Entrada	@PathVariable Integer idCancion, @PathVariable

	Integer duracion
Modo de acceso	Postman
Formato de Respuesta	ResponseEntity<Cancion>
Código Respuesta	Ok ->HttpStatus.OK(200) CancionNotFound -> Http.Status.NOT_FOUND error.html.
Descripción	Este método permite actualizar el campo duración de la canción. Devuelve un JSON donde se muestra el campo actualizado. Si no encuentra la canción lanza el error CancionNotFound y redirige a error.html con código 400.

- Servicio que actualiza el campo nombre de un álbum.

Actualizar nombre	
Ruta URL	http://localhost:9090/album/3/Un dia en el cielo
Método	Put
Entrada	@PathVariable Integer idCodigo, @PathVariable String nombre
Modo de acceso	Postman
Formato de Respuesta	ResponseEntity<Album>
Código Respuesta	Ok ->HttpStatus.OK(200) AlbumNotFound -> Http.Status.NOT_FOUND error.html
Descripción	Este método permite actualizar el campo nombre de la canción. Devuelve un JSON donde se muestra el campo actualizado. Si no encuentra el álbum lanza el error AlbumNotFound y redirige a error.html con código 400.

4. Secuencia de pasos para verificar el funcionamiento de JWT.

Paso 1: En Postman creamos un usuario utilizando el servicio @PostMapping (“/nuevousuario”). Devuelve un JSON que incluye el mensaje el usuario se ha registrado con éxito.

Paso 2: A continuación, nos logueamos con otro servicio, un PostMapping (“/login”) al que le pasamos el usuario y contraseña con las que registramos al usuario creador en el paso 1. Este servicio devuelve un JSON con un token cuyo código de respuesta es un Http.Status.OK (200).

Paso 3: Tras el logueo, hacemos un servicio Get que nos muestre una lista de discográficas. Para ello, tras escribir la petición en Postman, en la pestaña Authorization incluimos el token que obtuvimos en el paso 2 seleccionando la opción Bearer Token. En la pestaña Header comprobamos que aparece la autorización con el token y si clicamos sobre send, si nuestra seguridad está correctamente implementada podremos acceder al servicio. En caso de no tener autorización devuelve un código de respuesta Http.Status 401 Unauthorized Access.