DESARROLLO DE APLICACIONES PARA CIENCIA DE DATOS (PRIMER TRIMESTRE)

PROYECTO SPOTIFY: MEMORIA

SEGUNDO CURSO

GRADO EN CIENCIA E INGENIERÍA DE DATOS

ESCUELA DE INGENIERÍA INFORMÁTICA

UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA

ÍNDICE

RESUMEN

El proyecto está dividido en tres partes dependiendo de la responsabilidad que tienen,por lo que cuando hacemos un cambio en alguna parte del código, esto no afecta a otra parte del mismo. De estas tres partes dos se encuentran en paquetes.

En el primer paquete llamado “controller” se encuentran varias clases cuyo objetivo es controlar el flujo del programa. La clase “Controller” se encarga de almacenar los artistas, los álbumes y las canciones en las tablas correspondientes. La clase “MapManager” se encarga de manejar el mapa en el cual se encuentran los artistas con sus Ids. Es decir, contiene el mapa y te permite añadir nuevos elementos a este, obtenerlos para hacer uno de ellos y mostrarlos. La clase “SpotifyAccessor” se encarga de pedir los datos de los artistas, álbumes y canciones a la API de Spotify la cual está basada en principios REST simples. Está devuelve metadatos JSON sobre artistas musicales, álbumes y pistas, directamente desde el catálogo de datos de Spotify. La clase “SpotifyAuthorization” se encarga del proceso de Autorización para qué Spotify nos permita acceder a sus datos.

En el segundo paquete, llamado “model” se encuentran varias clases cuyo objetivo es modelizar los datos obtenidos proporcionándoselos al controlador cuándo el usuario lo solicita. La clase DbManager se encarga de conectar con nuestra base de datos local creada con SQLite, esta se encuentra en el paquete “database” y se llama “spotifydb”, se encarga de manejar la base de datos borrando las tablas en caso de que existan para no intentar crear las tablas ya existentes de una ejecución anterior, creando tres tablas, una para artistas, una para álbumes y otra para canciones con sus correspondientes filas y de insertar los datos en las diferentes tablas. Finalmente cierra la conexión con la base de datos. Por último las tres clases “Artist”, “Album”, “Tracks”, clases POJO (Plain Old Java Object). Son clases a partir de las cuales se instancian este tipo de objetos, son clases simples e independientes del framework utilizado. Estas clases no poseen restricciones especiales (más allá de las proporcionadas por el lenguaje), y se usan para simplificar la estructuración de los desarrollos, reduciendo su complejidad, aumentando la legibilidad y facilitando la reutilización de código.

Por último, el “Main” o la clase principal actúa en cierto sentido como una vista ya que contiene el flujo del programa de manera que el usuario pueda interactuar con él. Al iniciar el programa se han de introducir un client id y un client secret, estos nos los proporciona Spotify al crear una cuenta. Tras esto te permite seleccionar tres opciones: añadir artista, en cuyo caso se necesitará el nombre del artista y su id. Almacenar un artista, se almacena el artista seleccionado con sus álbumes y sus canciones en la base de datos. Y cerrar el programa.

RECURSOS UTILIZADOS

Entornos de desarrollo utilizados

- Intellij
- Maven

Herramientas de control de versiones

- Git
- GitHub

Herramientas de documentación

- JDK 11 Documentation
- SQLite Tutorial
- Spotify Documentation
- Maven Central

DISEÑO

Patrones y principios de diseño utilizados

He intentado aplicar un estilo arquitectónico en mi código, escogí el MVC, es decir el Model View Controller, ya que es el que hemos visto en clase y el que entendí mejor además me parece una un estilo arquitectónico adecuado para este proyecto ya que crea independencia de funcionamiento. Facilita el mantenimiento en caso de errores. Ofrece maneras más sencillas para probar el correcto funcionamiento del sistema y permite el escalamiento de la aplicación en caso de ser requerido.

He intentado tener en cuenta los 5 principios SOLID de diseño de aplicaciones de software que son los siguientes:

S – Single Responsibility Principle (SRP)

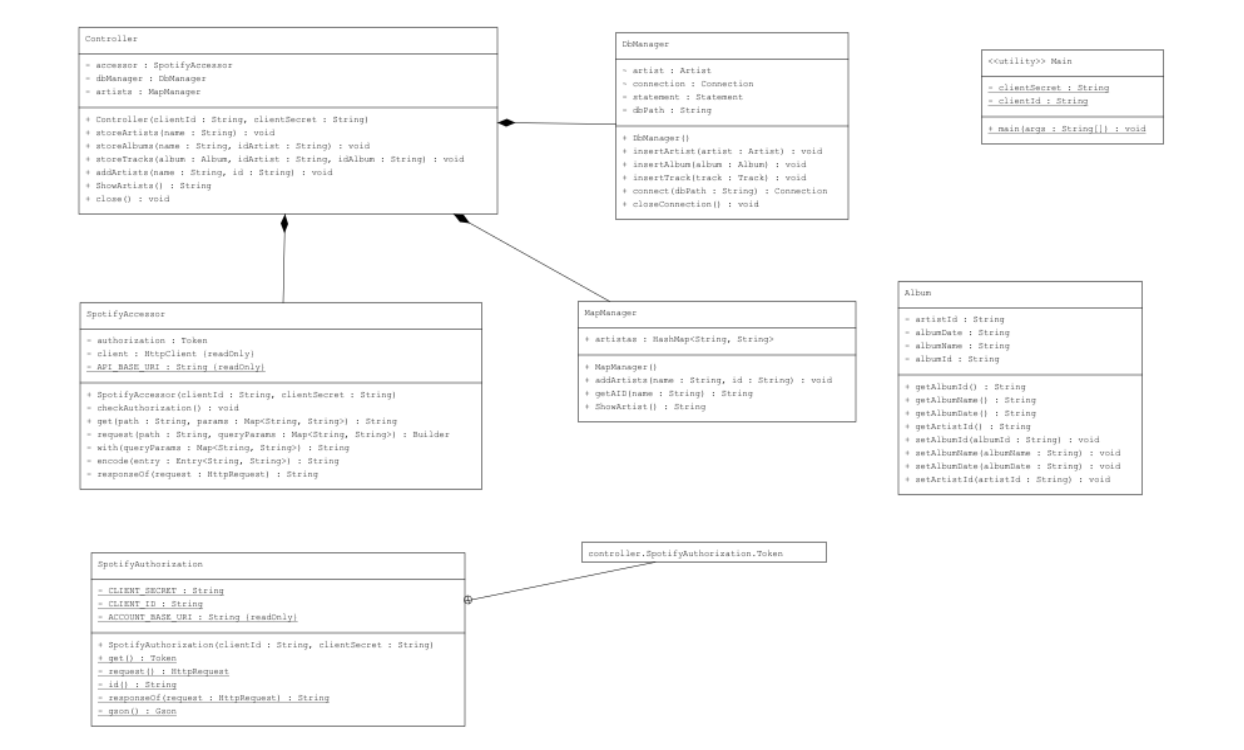
O – Open/Closed Principle (OCP)

L – Liskov Substitution Principle (LSP)

I – Interface Segregation Principle (ISP)

D – Dependency Inversion Principle (DIP)

Diagrama de clases



CONCLUSIONES

Este proyecto ha sido el primero que he tenido qué desarrollar en Java con más de dos clases con lo cual he aprendido bastante sobre como programar en Java. También he aprendido como crear una base de datos e insertar datos en esta. Además he entendido mejor como funciona una API REST y cómo trabajar con ella.

En cuanto al entorno de desarrollo intellij he de decir que me ha gustado mucho su funcionalidad de autocompletado y refactoring que permite programar de forma mucho más eficiente.

Si tuviera que empezar otra vez me gustaría implementar una interfaz gráfica con la qué el usuario pueda interactuar o poder mostrar la base de datos por consola. Además implementar el estilo arquitectónico MVC de manera qué la vista tenga su propio paquete y modularizar el código un poco más.

LÍNEAS FUTURAS

Como mencioné en el apartado anterior me gustaría añadir una interfaz gráfica o al menos mostrar los datos de la base de datos por consola. Me gustaría hacer un poco más eficiente el código para que si el usuario quiere obtener datos de muchos artistas la API de Spotify no se sature.

BIBLIOGRAFÍA

- https://docs.oracle.com/en/java/javase/11/
- https://jsoncrack.com/editor
- https://developer.spotify.com/documentation/web-api/reference/#/operations/get-track
- https://www.sqlitetutorial.net/sqlite-java/create-database/
- https://docs.oracle.com/en/java/javase/12/docs/api/java.base/java/util/Map.html#of()

CONTRAPORTADA

Autor

Mara Pareja del Pino

Fecha de creación de la memoria

6/11/2022

Versión y revisión

Versión final 1.3