SPOTIFY A SQLITE 2022-2023

9 noviembre

Universidad de Las Palmas de Gran Canaria

Escuela de Ingeniería Informática

Grado en Ciencia e Ingeniería de Datos

ULPGC
Universidad de
Las Palmas de
Gran Canaria

Autor:

Óscar Rico Rodríguez

Versiones:

0.0 [01/11/2022] (No pude comenzar antes por problemas con el IntelliJ)

Programa Base

1.0 [04/11/2022]

Esta versión venía con menú y base de datos, pero era algo complicado de entender, funcional, aunque mal estructurado, repleto de statics y sin seguir ningún tipo de patrón o diseño.

2.0 [04/11/2022]

Esta versión implementa una CLI más sencilla, misma base de datos, pero con un código más estructurado siguiendo el patrón MVC y con un diseño de Clean Code.

2.5[06/11/2022]

Esta versión implementó una recogida de datos de la API y subida de datos a la base de datos más eficiente, diversos controles para evitar excepciones con la API de spotify, muestra de resultados al finalizar el programa y eliminación de registros repetidos. Además de mejoras en el patrón y diseño.

3.0[08/11/2022] (FINAL)

Esta versión implementó una mejora en el patrón, distribución y diseño del código, mayor claridad, implementación de una interfaz para las bases de datos y reducción en el número de clases.

Resumen
Spotify a SQLite

Para este proyecto se nos pedía realizar un programa que recogiera información sobre cinco o más artistas solistas mediante la API de spotify y se almacenaran en una base de datos SQLite en la que existieran al menos tres tablas.

Contenido

Mi programa consta de trece clases y una interfaz para la base de datos. Doce de estas clases se dividen en seis paquetes:

- controller. Contiene la clase Controller desde la que se maneja el programa entero.
- database. Contiene la clase SQLiteDatabase y la interfaz Database.
- spotify. Contiene tres clases Spotify (clase donde se descarga la información), SpotifyAccessor y SpotifyAuthorization. Además de otro paquete propio llamado schemas que contiene clases POJO usadas en la extracción de datos.
- model. Contiene tres clases POJO Artist,Album y Track usadas para la muestra de datos en el view.
- view. Contiene la clase View encargada de mostrar datos al usuario y recoger los datos que el usuario inserte. Podría verser como una CLI (Command Line Interface).

La clase que se encuentra fuera de estos paquetes es la clase Main encargada de inicializar el programa.

Funcionamiento

Al ejecutar el programa la clase main inicializa la clase Controller que a su vez inicializa las clases SQLiteDatabase, Spotify y View. El main además pone en funcionamiento el método control que manda a la clase View (mediante el método show) mostrar un mensaje donde se le pide al usuario insertar las ids de los artistas y permite (mediante el método read) que el usuario introduzca las ids de uno o varios artistas de los que quiera obtener la información.

Tras esto, el control manda a la clase Spotify realizar una descarga de los datos de dichos artistas, sus álbumes y sus canciones (mediante los métodos downloadArtists, downloadAlbums y downloadTracks) en la API. Después de esto, control recoge los datos y los distribuye a la clase SQLiteDatabase para que esta almacene la información en la base de datos(mediante los métodos storeArtists, storeAlbums y storeTracks) , tras eso control manda eliminar la información repetida (mediante el método deleteRepeatedRows) , y por último el control seleciona los datos de 50 artistas, 100 álbumes y 150 canciones (mediante los métodos selectArtists, selectAlbums y selectTracks), y se los manda manda a la clase View para que los muestre al usuario (mediante los métodos showArtists, showAlbums y showTracks).

Anotaciones

En esta pequeña parte del trabajo, quería anotar que conozco la existencia de métodos más eficaces con añadidos en el insert como “ON CONFLICT” para el borrado de datos repetidos, pero como es una práctica, preferí mostrar otras opciones que ampliaban el manejo de instrucciones en SQL a no solo “INSERT”.

Índice

Recursos utilizados

Entorno de desarrollo

- IntelliJ

Controlador de Versiones

- Git

Editor de texto

- Word
- Sublime

Alojamiento del Proyecto

- GitHub

Diseño

Patrón y principio de diseño usado

Como patrón de diseño para mi proyecto decido usar el Movel View Controller (MVC) ya que es un patrón sencillo, escalable y mantenible. Por lo que no lo vi mala opción para un primer proyecto.

El MVC es un modelo divido en 3 capas donde cada cual cumple una función espéficica.

- View. Parte de la aplicación donde se contempla la interfaz gráfica. En esta capa se obtiene información sobre lo que quiere el usuario (Eventos) y se muestra la información del Modelo.
- Model (Lógica de negocio). Parte de la aplicación donde se relacionan los datos con los que se van a trabajar. Es el encargado de mapear las actividades del mundo real a la forma en la que se va a modificar la información.
- Controller. Parte de la aplicación que responde a los eventos del usuario y decide que view mostrar al usuario dependiendo de la solicitud recibida (Se puede ver como la conexión entre el model y el view).

En cuanto al principio de diseño, he seguido la guía de Clean Code que se nos proporcionó en el Campus Virtual.

Clean Code es una filosofía de desarrollo de software que consiste en aplicar técnicas simples que facilitan la escritura y lectura de un código.

Algunas de las principales prácticas para tener un código limpio son las siguientes:

1. Los nombres son importantes
    - Debe ser preciso, sin preocupación sobre el tamaño del nombre, es preferible claridad antes que brevedad.
2. Regla del boy scout
    - Esta regla indica que el código debe quedar más limpio de lo que estaba antes de ser editado
3. Métodos breves y descriptivos
    - Debemos evitar el anidamiento excesivo (complejidad ciclomática). Esto se consigue asignando a cada método una única funición.
4. DRY (Don´t Repeat Yourself)
    - Evitar el código duplicado. El código repetido entre diferentes clases, se pasa a clases abstractas.
5. Comentar solamente lo necesario
    - El código bien escrito se explica solo, sin necesidad de comentarios.
6. Tratamiento de errores
    - Para un código limpio, tratar las excepciones de forma correcta es un gran paso.
7. Tests limpios
    - Realizar tests es una etapa muy importante en la programación, un código solo se considera limpio después de haber sido válidada con varias pruebas.

Diagrama de clases

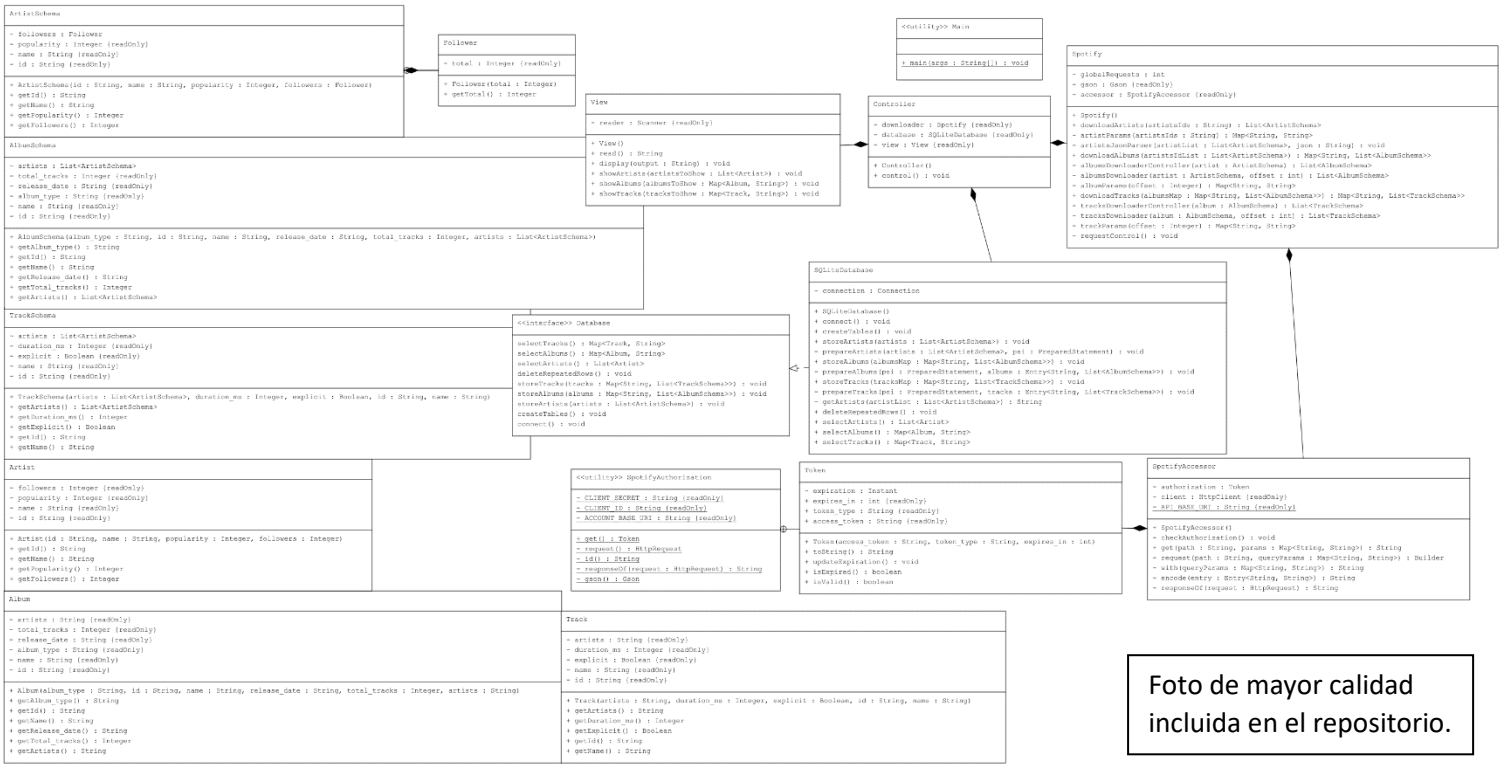

Foto de mayor calidad incluida en el repositorio.

Conclusiones

Durante el desarrollo de esta actividad he afianzado conocimientos con el uso de APIs, con la utilidad de los offset, he ampliado conocimientos sobre SQL con el uso de SQLite para el manejo y creación de la base de datos y con el tratado de Strings con el uso de format o regex en java. Además de tener un primer contacto con la programación siguiendo un estilo arquitectónico como es el MVC y un diseño de programación como el Clean Code. Por último, fuera de los requisitos del proyecto, he realizado investigaciones donde descubrí métodos para incrementar la velocidad de insertados en las bases de datos y conversión de listas de objetos enteras, en lugar de ir objeto por objeto.

En cuanto a experiencias en el proyecto, si comenzara de nuevo, empezaría por el uso de Git desde el inicio del proyecto, para tener un control de las versiones y evitar problemas como pérdidas de código, y enfocaría el trabajo de una manera distinta, buscando un aumento de rendimiento y la eficiencia en la descarga de datos en lugar de la inserción, pues me di cuenta tarde de que esta segunda parte es la que más tiempo lleva, y para un código más rápido es donde más me tendría que haber centrado. Una de las principales cosas que lamento no haber hecho desde el inicio es el estudio de las funciones several que agilizan bastante y reducen el número de peticiones.

Líneas Futuras

Bajo mi punto de vista si, quisiéramos evolucionar este producto hasta convertirlo en un producto comercializable, debería tener en cuenta los siguientes aspectos:

- Creación de una interfaz gráfica.
- Mejora en el sistema de peticiones a la API de Spotify (realiza muchas peticiones).
- Posibilidad de insertar el nombre del artista, en lugar de la id.
- Posibilidad de reproducir la música almacenada.
- Funcionalidad 24/7 alojando el producto en un servidor online.
- Mejora en el rendimiento y eficiencia en la descarga de datos.

Bibliografía

MVC

¿Qué es patron MVC y cómo funciona?

MVC Explained in 4 Minutos

Clean Code

Clean Code: Código limpio, ¿qué es?

Guía de estilo en Java y buenas prácticas para un código limpio

Java

Deserialize a List<T> object with Gson

delete a repeated database table rows that have the same name

SQLite - Increase speed of insertion