1 Implémentation

1.1 Description

L'implémentation se fait en deux étapes. La première consiste à effectuer l'authentification, c'est-à-dire récupérer le Bearer Token à partir du client_id et du client_secret, afin de pouvoir accéder aux routes protégées, comme les données des artistes. Voir Figure1

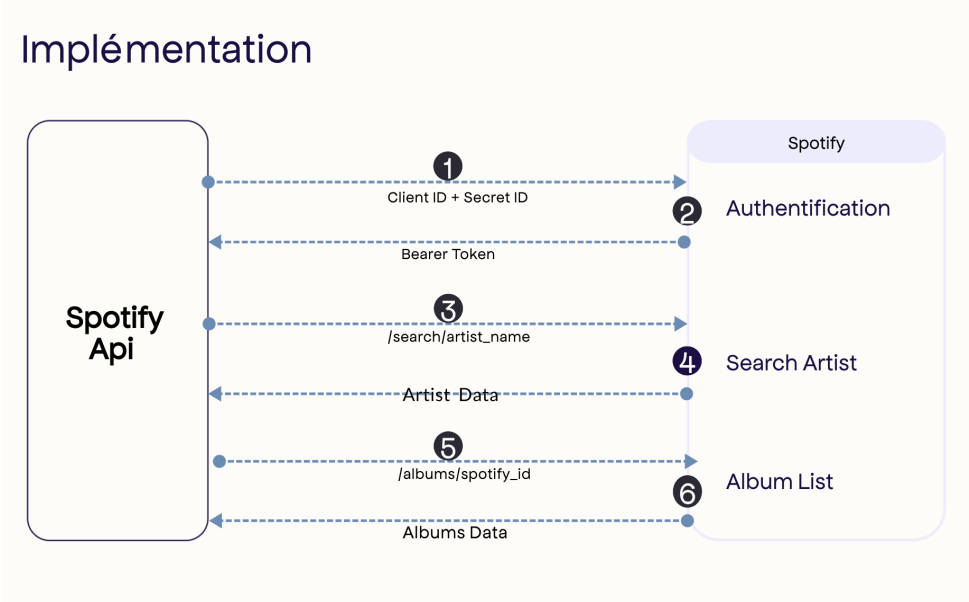

FIGURE 1 – Implémentation

— Comme on souhaite récupérer la liste des albums d'un artiste à partir de son nom, il est d'abord nécessaire de récupérer son identifiant Spotify. Cet identifiant permet ensuite d'obtenir la liste de ses albums.
— La recherche par nom d'artiste peut retourner plusieurs résultats. J'ai donc fait le choix de ne prendre en compte que le premier artiste de la liste.
— Le bearer token est régénéré à chaque requête pour s'assurer qu'il n'a pas expiré.

Le format de retour est du JSON. J'ai utilisé @derive pour sérialiser automatiquement les structures. Cette approche fonctionne, mais elle n'est pas recommandée si l'application évolue et qu'on ajoute plusieurs routes avec des formats de réponse différents, il sera plus adapté de mettre en place des view.

1.2 Schéma de base de données

Le schéma de base de données reste assez simple, avec deux tables : une pour les artistes et une autre pour les albums. Une table de liaison entre les deux n'est pas nécessaire, car on ne cherche pas à historiser ou à conserver des informations qui justifieraient une telle structure.

Important : Même si un artiste existe déjà en base, j'ai fait le choix d'interroger systématiquement l'API Spotify, qui reste la source de vérité. Cela permet de s'assurer que la base est toujours à jour. Par exemple, si un artiste a sorti un nouvel album qui n'est pas encore enregistré en base, cette vérification permet de récupérer et stocker les nouvelles données automatiquement.

1.3 Documentation

J'ai mis en place une petite documentation Swagger (Voir Figure 2) qui permet de visualiser les routes de l'API et de les tester directement via une interface web. Cette documentation repose sur la librairie OpenAPI, open_api_spex.

J'ai structuré les différentes réponses et entrées de l'API à l'aide de schémas définis à part, ce qui permet de factoriser le code et d'éviter la répétition des mêmes structures dans chaque route.

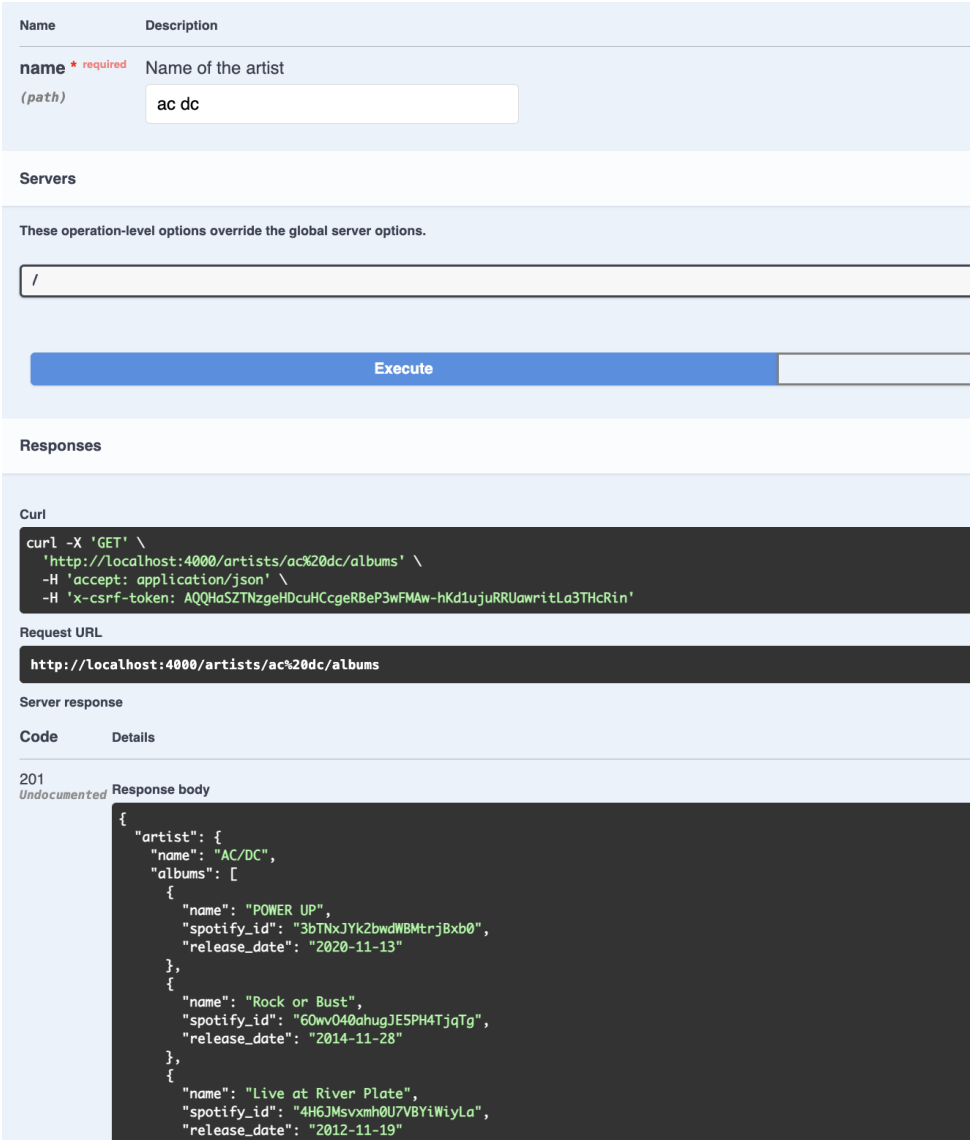

FIGURE 2 – Documentation