Pojet PC3R - théme MUSIQUE avec l'API deezer

Youra AHMED, Merieme BENAISSA

March 24, 2025

Sujet de l'application et principales fonctionnalités

L'application est un service web musical utilisant l'API Deezer, permettant aux utilisateurs de rechercher des morceaux, des albums et des artistes, de créer des playlists, d'écouter de la musique en streaming, de commenter les morceaux, de les liker et de les noter. Les utilisateurs peuvent gérer leurs playlists personnalisées en fonction de leurs goûts et préférences musicales.

API Web choisie

L'API choisie est l'API Deezer, qui permet d'accéder à une base de données musicale comprenant des morceaux, des albums, des artistes et des playlists. Voici les principaux objets disponibles dans l'API :

- **Album** : Permet d'accéder aux informations détaillées sur un album, telles que son nom, sa date de sortie, ses genres musicaux, ainsi que la liste des morceaux qui le composent. Chaque album est également associé à un artiste, et vous pouvez obtenir les albums similaires.
- **Artist** : Permet d'obtenir des informations sur un artiste, telles que son nom, son lien de profit sur deezer, le nombre de ses albums, un lien vers la liste de ses morceaux, le nomnbres de ses fans ainsi que le top liste de ses morceaux.
- **Chart** : Permet d'accéder aux classements de musique populaires par genre ou par période, par exemple les morceaux les plus populaires du moment ou les albums les plus écoutés.
- **Editorial** : Permet de récupérer des informations sur des recommandations éditoriales telles que des playlists, albums ou morceaux suggérés par Deezer en fonction des tendances actuelles.
- **Episode** : Permet d'obtenir des informations sur un épisode de podcast, y compris son titre, sa description et son lien pour l'écouter.
- **Genre** : Cette ressource permet de récupérer des informations sur les genres musicaux (pop, rock, rap, etc.), ainsi que les albums et artistes associés à chaque genre.
- **Infos** : Permet d'accéder aux informations générales sur l'API Deezer, comme la version de l'API et d'autres informations liées à son utilisation.
- **Options** : Permet de récupérer des informations sur les options des utilisateurs de l'API, comme les formats de sortie ou les configurations disponibles.
- **Playlist** : Représente une playlist musicale. Vous pouvez obtenir des informations sur la playlist, sa description, le nombre de ses morceaux etc.
- **Podcast** : Permet d'accéder aux informations sur les podcasts disponibles sur Deezer, incluant le titre, la description et les épisodes associés à un podcast etc.
- **Radio** : Permet d'obtenir des informations sur les radios de Deezer et un lien vers les piste du radio.
- **Search** : Permet d'effectuer des recherches dans la base de données Deezer en fonction de différents critères (album, artiste, morceau, playlist, etc.). Elle renvoie des résultats correspondant aux critères de recherche fournis.

- **Track** : Représente un morceau de musique. Il inclut des informations comme le titre, la durée, l'artiste associé, l'album auquel il appartient, ainsi que les liens pour l'écouter en streaming.
- **User** : Permet de récupérer des informations sur un utilisateur de Deezer, comme son nom, ses playlists, ses morceaux favoris, etc.

Ces objets sont utilisés pour récupérer des informations pertinentes sur la musique, les utilisateurs, et les préférences musicales, permettant ainsi de créer une expérience interactive et dynamique pour les utilisateurs de l'application.

Fonctionnalités de l'application

L'application offre plusieurs fonctionnalités :

- **Recherche de musique** : Recherche de morceaux, albums et artistes via la barre de recherche.
- **Création et gestion de playlists** : Les utilisateurs peuvent créer et personnaliser leurs playlists musicales.
- **Écoute en streaming** : Les utilisateurs peuvent écouter des morceaux directement depuis l'application.
- **Commentaires et likes** : Les utilisateurs peuvent laisser des commentaires et liker les morceaux.
- **Notation** : Les utilisateurs peuvent attribuer une note aux morceaux écoutés.

Cas d'utilisation classiques

Recherche de musique

L'utilisateur peut rechercher un morceau, un album ou un artiste en utilisant une barre de recherche. Après avoir entré un terme de recherche, les résultats sont affichés sous forme de plusieurs publications contenant le nom dont il a fait la recherche.

Création de playlists

L'utilisateur peut créer une nouvelle playlist, y ajouter des morceaux qu'il aime et l'écouter en streaming. Il peut également trier soit par l'alphabet ou les musiques les plus récentes.

Écoute de musique

L'utilisateur peut écouter des morceaux en streaming depuis l'application grâce a un bouton play et l'arrêter avec le même bouton, sinon si il arrêt pas la musique va boucler a l'infinie.

Commentaires et Likes

L'utilisateur peut commenter les morceaux et albums qu'il écoute, et lui attribuer un "like" pour indiquer qu'il apprécie un morceau.

Notation des morceaux

L'utilisateur peut attribuer une note aux morceaux écoutés, avec un système de notation simple, par exemple de 1 à 5 étoiles.

Données stockées dans la base de données

Les données suivantes seront stockées dans la base de données du serveur :

- Utilisateurs : identifiant, nom, email, mots de passe cryptés.
- Playlists : identifiant de la playlist, nom, description, liste des morceaux (références Deezer).

- Notations : identifiant de l'utilisateur, identifiant du morceau, note attribuée.
- Commentaires : identifiant du morceau, identifiant de l'utilisateur, texte du commentaire.
- Likes : identifiant du morceau, identifiant de l'utilisateur.

Schéma de la base de données

Un schéma possible de la base de données pourrait inclure les tables suivantes :

- **Users** : id, name, email, password_hash.
- **Playlists** : id, user_id, name, description.
- **Tracks** : id, playlist_id, track_id.
- **Ratings** : user_id, track_id, rating.
- **Comments** : user_id, track_id, comment_text.
- **Likes** : user_id, track_id.

Mise à jour des données et appel de l'API externe

Afin de garantir l'actualisation des données et de fournir aux utilisateurs des informations à jour, l'application effectuera des mises à jour automatiques en interrogeant l'API Deezer chaque jour pendant la nuit. Cette mise à jour inclura :

- La récupération des nouveaux morceaux et albums ajoutés sur Deezer.
- La mise à jour des informations sur les artistes populaires.
- L'actualisation des classements et tendances musicales.

Ce processus sera exécuté sous forme de tâche planifiée. Il consistera en des requêtes API pour obtenir les dernières données et les enregistrer dans la base de données locale afin d'améliorer la réactivité et réduire les appels API en temps réel.

Les requêtes principales utilisées pour la mise à jour incluront :

- `GET /chart` : récupération des morceaux et albums tendances.
- `GET /editorial` : récupération des recommandations éditoriales.
- `GET /artist/top` : mise à jour des classements des artistes les plus écoutés.

Grâce à cette mise à jour quotidienne, l'application pourra offrir une meilleure expérience utilisateur tout en minimisant les appels API inutiles.

Description du serveur

Le serveur sera écrit en Servlets naturelles et s'exécutera dans Tomcat/Catalina. Il gérera plusieurs ressources comme :

- **Gestion des utilisateurs** : inscription, connexion, gestion des profils.
- **Gestion des playlists** : création, modification.
- **Gestion des notations** : enregistrement des évaluations des morceaux.
- **Gestion des commentaires et likes** : enregistrement des interactions des utilisateurs avec les morceaux.

Le serveur communiquera avec l'API Deezer pour récupérer des données à la demande des utilisateurs.

Description du client

L'application sera une application monopage (SPA) utilisant React pour le rendu dynamique des données. Les pages principales incluront :

- **Page d'accueil** : affichage des morceaux populaires et options de recherche.
- **Page de recherche** : liste des résultats de recherche d'artistes, albums et morceaux.
- **Page de playlist** : gestion et écoute des playlists.

Les appels aux différents composants du serveur (création de playlists, notation, commentaires et likes) seront effectués via AJAX en utilisant des requêtes HTTP asynchrones.

Requêtes et réponses HTTP

Les requêtes entre le client et le serveur seront principalement des requêtes GET et POST pour récupérer et envoyer des données. Par exemple, pour rechercher un morceau :

- Requête : `GET /api/search?query=trackName`.
- Réponse : JSON contenant les informations sur les morceaux trouvés.

Pour enregistrer une notation :

- Requête : `POST /api/rate` avec {track_id, user_id, rating}.
- Réponse : Confirmation de la notation.

Pour enregistrer un commentaire :

- Requête : `POST /api/comment` avec {track_id, user_id, comment_text}.
- Réponse : Confirmation du commentaire.

Pour liker un morceau :

- Requête : `POST /api/like` avec {track_id, user_id}.
- Réponse : Confirmation du like.