
2020/2021

# Module WebService

# Projet « DizifyMusic »

Sébastien PRODHOMME
Matthieu BACHELIER

# Description du Projet

- Réaliser une (fausse) plateforme de streaming musical moderne, type Deezer, Spotify ou Apple Music

- Sans musique évidemment, pour raisons de droits

# REST : TP

**Objectif du projet sur la partie REST**

- **Réaliser un projet complet où communiquent, via REST, un client et un serveur**
- **Pas de lecteur de musique à implémenter !**
- **Projet à réaliser en groupe**
  - Minimum 2 personnes
  - Maximum 4 personnes
- **À rendre le jeudi 19/11/2020 en fin de journée**

# Contraintes techniques (1)

- WS + Backend codé en Java + Spring

- Frontend codé en JS + React

  OU

- Frontend codé en JS + React-native

# Contraintes techniques (2)

- **Base de données relationnelle**
  - MySQL (ou MariaDB)

# Modèle relationnel attendu (1)

- **Table Artistes**
  - Contient un ensemble d'artistes identifés par un Nom (ou alias) et une Image
- **Table Albums**
  - Contient un ensemble d'albums identifiés par un Nom, une Date, une Image et un Auteur (pas de modèle complexe type Compilation)
- **Table Titres**
  - Contient un ensemble de titres, identifiés par un Nom, une Durée, un Auteur et un Album ou non (pour les EP, Single, titres exclusifs)

# Modèle relationnel attendu (2)

- **Table Utilisateur**
  - Contient un ensemble d'utilisateurs du service identifiés par une adresse Email au minimum. On peut ajouter d'autres champs type Avatar, Pseudo
- **Table Playlist**
  - Contient un ensemble de Titres qu'un utilisateur a choisi pour construire des playlists personnalisées
- **Table Favoris**
  - Contient un ensemble d'items variés (Artistes, Albums ou Titres) qu'un utilisateur a aimé

# Modèle relationnel attendu (3)

- **Table Administrateur**
  - Contient des utilisateurs spéciaux qui peuvent ajouter du contenu : Artiste, Album et Titre

# Fonctionnalités à implémenter (1)

- **Page d'accueil en mode publique, sans authentification**
  - Afficher 3 Artistes (en aléatoire pour plus de simplicité)
  - Idem pour les Albums
- **Page de listing des Artistes**
  - Afficher ses Albums dans une section
  - Afficher ses Titres dans une autre section
- **Page de listing des Albums**
  - Afficher quel est l'Artiste à l'origine de l'Album
  - Afficher ses Titres dans une section

# Fonctionnalités à implémenter (2)

- Page de listing des Playlist
- Page de listing des Favoris

# Fonctionnalités à implémenter (3)

- **En tant qu'utilisateur, je veux pouvoir ajouter un Artiste, un Album ou un Titre dans mes Favoris**
  - Je veux également pouvoir supprimer un favori
- **En tant qu'utilisateur, je veux pouvoir ajouter des Titres dans des Playlists**
  - Je veux également pouvoir supprimer une playlist
- **En tant qu'utilisateur, je veux pouvoir changer mon Avatar**
  - Je veux également pouvoir supprimer mon Avatar

# Gestion des rôles (1)

- **Un utilisateur doit être authentifié pour créer du contenu qui lui est propre**
- **Un utilisateur ne peut consulter le contenu d'un autre utilisateur**
- **Un administrateur peut ajouter du contenu de type Artiste / Album / Titre**
  - Il peut également modifier / supprimer ce contenu
- **Un administrateur n'est pas un utilisateur et ne peut pas créer du contenu type Playlist / Favori**

# Gestion des rôles (2)

- **Un Administrateur est créé à la main par le super-administrateur dans la base de données**
- **Un Utilisateur doit pouvoir créer un compte en automie avec son adresse email**
  - Pas de gestion réelle des emails à prévoir : on se servira juste de l'email pour ajouter un nouvel utilisateur (email = clé primaire unique)
  - Pas de suppression de compte, d'anonymisation ou de RGPD à implémenter

# Gestion de la sécurité

- **Implémenter un système de jeton JSON Web Token (JWT) pour sécuriser les appels au WebService**
- **Les actions de l'Utilisateur effectuées côté Client doivent être vérifiées côté Serveur**
  - Par exemple, pour la suppression d'une Playlist
    - DELETE /user/{userId}/playlist/{playlistId}
    - Vérifier côté serveur que le user ID du jeton et le user ID demandé dans l'URL sont identiques pour autoriser la suppression
- **Idem pour les actions de l'Administrateur**

# Pagination

- **Implémenter une gestion de la pagination**
  - Soit avec « page 1, page 2, …, page N-1, page N »
  - Soit en mode Infinite Scrolling

# Contenu Media / Image

- **Pour les images à afficher, partir soit sur une API externe pour afficher des images aléatoires**
- **Pour les plus motivés, référencer dans la base une image sur votre système de fichiers**
  - Pas de contenu Image stockée au sein même de la base de donnée façon BLOB
  - Toujours utiliser une URI pointant vers votre image en local

# Contenu Media / Image

- **Pour utiliser des images externes et ne pas s'embêter avec cette gestion d'images, utiliser les services suivants par exemple :**
  - API pour les Aristes / Avatars : https://pravatar.cc/
  - API pour les pochettes d'Albums : https://picsum.photos/
  - Ou autre !

# Pour aller plus loin (1)

- **Pour ceux qui auraient eu le temps de tout faire (!), vous pouvez implémenter les fonctionnalités suivantes :**
  - Afficher un Top Artiste / Album / Titre en fonction du nombre de Favoris sur la page d'Accueil
    - Top individuel ou global à tous les utilisateur, au choix
  - Afficher une zone de recherche dans le haut du site
    - Rechercher un type d'entité en particulier (grâce à groupe de boutons exclusifs pour Artiste / Album / Titre)
    - Présenter les résultats dans une drop-down liste

# Pour aller plus loin (2)

- **Rendre le projet disponible sur Internet**
- **Intégrer une usine logicielle**
  - But : automatiser le déploiement au commit