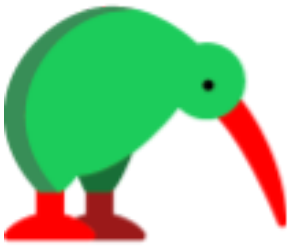
RAPPORT DE PROJET



KIWI

UE LIFProjet
L3 informatique
Semestre de printemps 2022

BERNOT Camille, *p1908800*
CECILLON Enzo, *p1805901*
FIETIER Loris, *p1805561*

RICO Fabien
CAZABET Rémy

I. Introduction

Dans le cadre de l'UE LIFProjet, nous avons choisi le sujet FR4 qui correspond à la mise en place d'un site web.
Nous avions compris que le but était de mener à bien la conception et le développement d'une application web de bout en bout. De ce constat, nous est venue cette idée d'application de 'streaming musical' réunissant les services de Spotify et de Youtube.
En effet, n'est-il pas frustrant d'être amené à écouter une version alternative de la musique que l'on voulait à l'origine car cette dernière n'est pas présente sur Spotify ? D'autre part, devoir ouvrir son navigateur et aller sur Youtube pour chercher cette musique est un processus répétitif et peu plaisant.

Afin de concevoir une application robuste et attractive, nous devions choisir les bons outils, les bonnes librairies, séparer le fonctionnement et l'affichage, gérer la communication dans l'équipe et tenir les délais fixés.

II. Front end

Pour créer l'interface utilisateur, nous avons fait le choix d'utiliser la bibliothèque JavaScript React et le framework CSS Tailwindcss. Nous avons donc créé plusieurs composants réutilisables (barre de menu, barre de recherche, lecteur musical, affichage des détails de la musique écoutée, affichage des playlists, des morceaux, …) et nous avons utilisé tailwindcss et ses propriétés CSS que l'on peut appeler au travers de classes HTML afin d'apporter une homogénéité et une cohérence graphique à l'application.

Ainsi avons également développé un thème sombre dont l'activation se base sur le thème du système d'exploitation et notre application est responsive. Cette dernière s'adapte donc aussi bien utilisable sur ordinateur que sur smartphone.

III. Back end

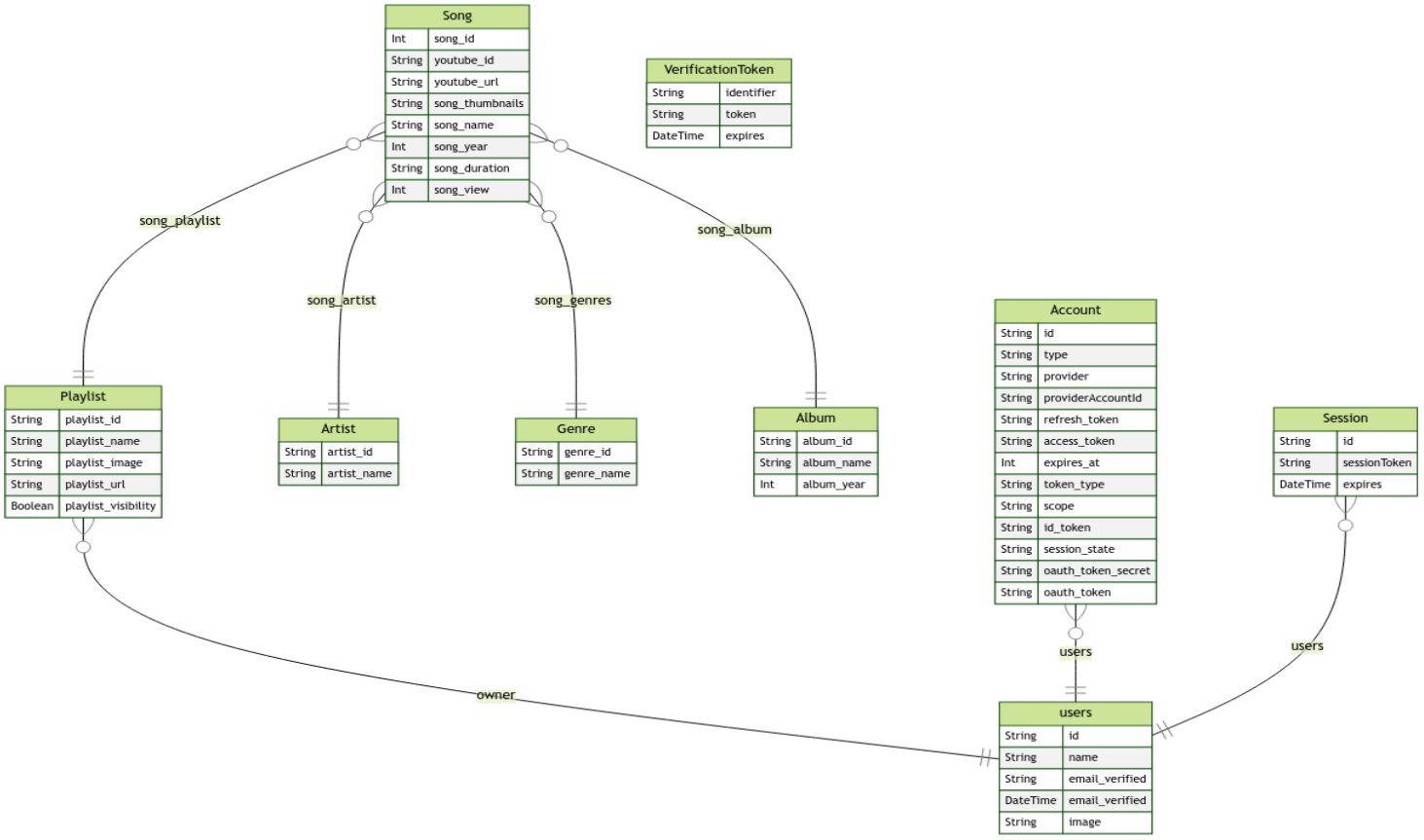
1. Authentification

Nous donnons à l'utilisateur la possibilité de se connecter par le biais de Google ou de Spotify, afin d'avoir à la fois un moyen de retrouver ce dernier dans notre base de données et un moyen de communiquer avec les services de Youtube et Spotify.

Ce dernier doit accepter les conditions d'utilisations de notre application qui demande si nous pouvons récupérer son nom, son mail, ses playlists, …
Chaque utilisateur se voit octroyer un "pass" qui permet à Spotify et à Youtube de l'identifier lorsque l'on soumet une demande à son nom pour une recherche ou autre.

Le processus de création dans la base de données est géré par la bibliothèque NextAuth que l'on configure avec les IDs de nos applications. Enfin, on effectue les opérations (ajouts, création, modification …) sur cette dernière via l'ORM (object-relational mapping**)** Prisma.

2. Schéma de la base de données



Chaque utilisateur possède un identifiant unique et des informations sur lui qui ont été récupérés lors de l'identification via le service Google ou Spotify.
Il a la possibilité d'avoir 1 ou 2 comptes suivant les services qu'il veut utiliser.
Ces comptes stockent des informations comme le "pass" (token d'accès) de l'utilisateur et d'autres informations pour ces services (provider).

Enfin, un utilisateur a la possibilité d'avoir des playlists qui lui appartiennent et qu'il peut partager. On attache à une ou plusieurs playlist des musiques que l'on identifie avec un id unique.
Ces dernières possèdes des informations comme leur nom, l'année, la durée ….

3. Gestion et utilisation des données

Pour que l'utilisateur ait une bonne expérience, nous récupérons ces données d'abord sur le serveur et les partageons aux éléments qui affiche le contenu.
Cette fonctionnalité offerte par NextJS, améliore grandement le chargement des pages et permet de gérer certaines subtilités comme le renouvellement des "pass" des utilisateurs s'ils ont expirés.

Nous stockons dans notre base de données, 2 catégories d'éléments, les utilisateurs et les musiques.
Très tôt nous avons dû réfléchir à une manière de stocker les musiques dans notre base de données à cause des limites de quota imposées par Youtube.
Notre approche a été la suivante :

- Si nous n'avons pas la musique ou quelque chose qui correspond à ce que l'utilisateur veut :
    - Nous demandons à Youtube ce qui correspond à la recherche.
    - Nous récupérons les données qui nous intéressent dans ce que Youtube nous a transmis.
    - Puis nous les envoyons à l'utilisateur, en parallèle nous ajoutons ces éléments sur notre base de données. Ainsi, la prochaine fois si nous recherchons cette même musique, nous n'aurons plus besoin de demander à Youtube.

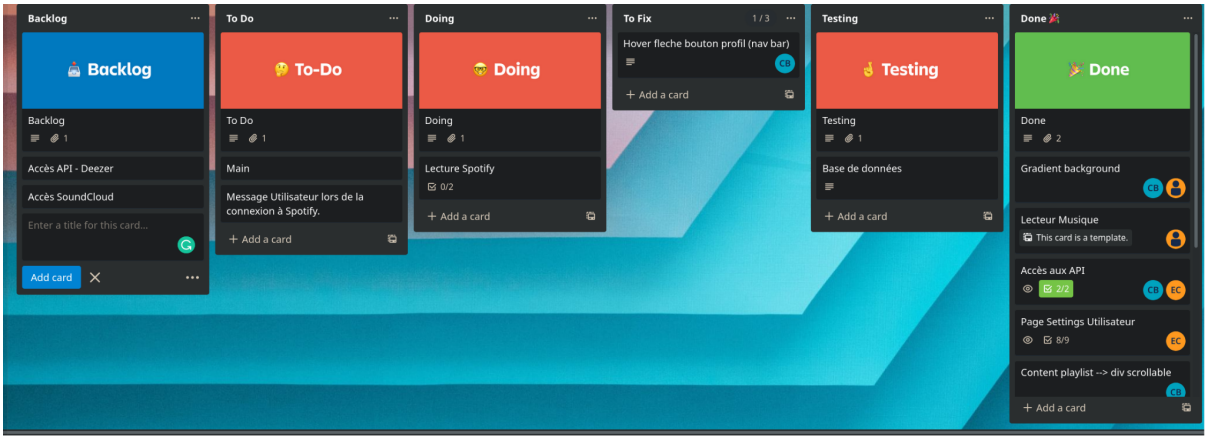
Enfin, de manière à ce que notre application puisse être plus tard mise en ligne, nous avons décidé d'héberger notre base de données sur la plateforme en ligne Heroku.

IV. Répartition des tâches

De façon à pouvoir suivre les avancements de chacun et pouvoir répartir le travail équitablement nous avons décidé d'utiliser la méthode Kanban via la plateforme Trello.
Grâce à cela, nous avons essayé de reproduire un environnement de développement qui se rapproche de ce que nous serions amené à utiliser dans une entreprise. Après concertation, chaque membre s'est vu attribuer une fonctionnalité qu'il devait

réaliser et passer sa carte dans les colonnes : En développement, A corriger, A Tester et terminé.
Ci-dessous, vous trouverez un tableau regroupant les différentes fonctionnalités réalisées et une capture du tableau Kanban utilisé.

Tâches	Réalisées par ….
Base de données	CECILLON Enzo
Authentification	CECILLON Enzo
Exploitation de l'API YouTube	CECILLON Enzo
Exploitation de l'API Spotify	BERNOT Camille
Gestion des comptes / connexion	CECILLON Enzo
Recherche	FIETIER Loris
Front-end / Responsive et Design	BERNOT Camille



V. Conclusion

Nous avons rencontré quelques difficultés lors de la réalisation de ce projet. Notamment sur l'utilisation de l'API YouTube qui restreint le nombre de requêtes quotidiennes.
Nous avons également dû revoir nos objectifs à la baisse à cause d'une mauvaise gestion du temps de notre part. Par conséquent, nous n'avons pas pu implémenter tout le côté "social" du projet (liste d'amis, visualisation des écoutes actuelles des amis, etc…). C'est donc tout naturellement que nous ajoutons ces fonctionnalités aux pistes d'amélioration de ce projet.

Nous pensons également qu'il serait agréable pour l'utilisateur d'ajouter les fonctionnalités suivantes :

- Mode invité (musiques jouables sans connexion de l'utilisateur)
- Messagerie interne entre abonnés / amis
- Playlists Personnalisés
- D'autres API (Deezer, soundcloud ….)

VI. Annexe

Next.js documentation
A Node.js wrapper for Spotify's Web API.
Documentation | Spotify for Developers
YouTube Data API Overview | Google Developers
React
Débutez avec React - OpenClassrooms
Spotify Database Schema | javascriptgorilla
JavaScript Debounce Function | Réglage de performance