|  |
| --- |
|  |
| DM Programmation C |
| Mots-mêlés |

|  |
| --- |
| Clément GAUDET / Maxime JAILLARD  19/11/2020 |

|  |
| --- |
| L2 – Bases de données |
| Projet The Wave |
| Etape 3 |

Clément GAUDET / Maxime JAILLARD

08/01/2021

Sommaire

[Manuel utilisateur 3](#_Toc60938007)

[Architecture du site 5](#_Toc60938008)

[Arborescence des pages 5](#_Toc60938009)

[Requêtes SQL majeures 5](#_Toc60938010)

[Gestion des sessions 6](#_Toc60938011)

[Fonctionnalités non-implémentées 6](#_Toc60938012)

The Wave est site web de distribution de musique à la demande, offrant à ses abonnés, en plus de la possibilité d’écouter librement tout un nombre de morceaux de tous genres, la possibilité de suivre des groupes, d’avoir des suggestions personnalisées ou encore d’accéder à de nombreuses playlists faites par d’autres utilisateurs, ainsi que de faire ses propres playlists.

# Manuel utilisateur

En dehors de la page d’accueil, l’accès au contenu de The Wave requiert une authentification avec un compte valable. En cliquant sur le bouton de connexion en haut à droite, on accède à la page de connexion, qui contient également un lien pour créer un compte si nécessaire.

Depuis la page d’accueil, on peut consulter les morceaux les plus écoutés de la semaine, les groupes les plus suivis et les derniers albums sortis. Les noms de morceau, album, groupes, etc. Sont des hyperliens permettant de consulter la page associée.

* La page de groupe permet à l’utilisateur de consulter les informations (nationalité, date de création, genre, etc.) ainsi que la liste des membres du groupe. Elle contient aussi la liste des albums et morceaux publiés par le groupe afin que l’utilisateur puisse naviguer directement vers leur page. En plus de ces fonctionnalités, on peut aussi ajouter le groupe à la liste des groupes suivis en cliquant sur le bouton « Suivre » ou enlever le groupe de la liste en cliquant sur « Ne plus suivre ».
* La page des morceaux contient les informations disponibles sur le groupe qui l’a publié, le genre, l’album d’origine (s’il y en a un) et la durée de l’audio. On peut aussi y trouver la liste des artistes ayant participé au morceau.
* La page des albums montre la couverture et leur description. La liste des morceaux contenus dans l’album permet de consulter leur page respective.

Depuis les pages des morceaux ou des albums, on peut lancer l’écoute en cliquant sur le bouton « Ecouter ». Si c’est un morceau, la page du lecteur propose uniquement l’audio et les paroles s’il y en a. Si c’est un album, le lecteur commence au premier titre de l’album et il est possible de passer d’un morceau au suivant ou au précédent en cliquant sur les boutons « PREC » et « SUIV ».

Depuis n’importe quelle page, on peut également accéder à notre profil à l’aide du bouton portant le pseudo du compte, situé en haut à droite dans l’en-tête de page. Cette page répertorie les informations personnelles de l’utilisateur connecté. Il peut aussi y consulter les morceaux les plus écoutés, son historique d’écoute et les dernières activités des utilisateurs ou des groupes suivis. On peut également créer nos propres playlists avec le bouton dédié ou éditer une playlist déjà existante. On pourra modifier le titre, la description et l’accessibilité publique ou privée de la playlist, ainsi que rajouter des morceaux, et supprimer la playlist complètement.

En visualisant la page de profil d’un autre utilisateur, on ne pourra pas consulter ses playlists ou groupes suivis, mais on peut ajouter l’utilisateur à la liste des utilisateurs suivis en cliquant sur le bouton « Suivre » ou enlever l’utilisateur de la liste en cliquant sur « Ne plus suivre ».

Pour trouver une page précise sur le site, on peut utiliser la page « Recherche » accessible depuis n’importe quelle page en cliquant sur le bouton correspondant dans l’en-tête. On est redirigé vers une zone à remplir avec les informations voulues.

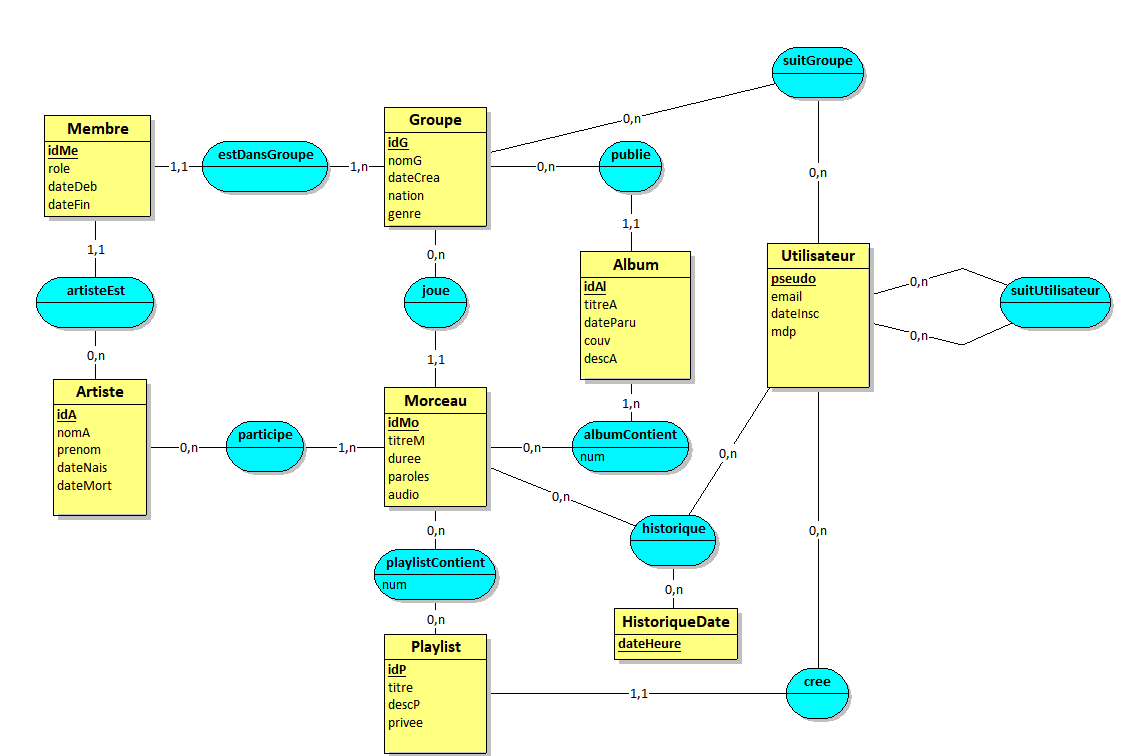
* On sélectionne tout d’abord le type d’entrées recherchées.
* La case Titre contient les mots-clés à rechercher dans le titre du morceau, de l’album, de la playlist ou le nom du groupe. Si la case reste vide, aucun titre ne sera éliminé lors de la recherche.
* La case Genre contient les mots-clés à rechercher dans le genre des morceaux, de l’album, de la playlist ou du groupe. Si la case reste vide, aucun genre ne sera éliminé lors de la recherche.
* La case Date contient l’année à rechercher dans une liste d’album. Si elle reste vide, aucune contrainte d’année de parution ne sera appliquée. (Remarque : l’année n’est utilisée que pour rechercher des albums)
* La case Artiste contient les noms ou prénoms à rechercher dans les participants aux morceaux, albums, playlists ou dans les membres d’un groupe. Si elle reste vide, tous les artistes seront considérés.

En plus des pages déjà présentées, on peut rechercher les playlists créées par d’autres utilisateurs. Sur ces pages, on trouve la composition de la playlist avec sa description et la liste des morceaux qui la composent. Comme pour les albums, on peut écouter ces playlists et le lecteur commence au premier titre.

La dernière page de l’en-tête est la page des suggestions. Sur cette page, on peut retrouver des suggestions personnalisées selon l’historique d’écoute de l’utilisateur : des morceaux de l’artiste que l’on écoute, les playlists créées contenant plusieurs morceaux écoutés habituellement et les groupes suivis par d’autres utilisateurs qui sont susceptibles de nous intéresser.

# Structure de la base de données

Schéma E-A :



**Schéma relationnel :**

**artiste**(idA, nomA, prenom, nationA, dateNais, dateMort)  
**membre**(idMe, role, dateDeb, dateFin, #idA, #idG)  
**groupe**(idG, nomG, dateCrea, nationG, genre)  
**morceau**(idMo, titreM, duree, paroles, audio, #idG)  
**album**(idAl, titreA, dateParu, couv, descA, #idG)  
**utilisateur**(pseudo, email, dateInsc, mdp)  
**playlist**(idP, titre, descP, privee, #pseudo)  
**participe**(#idA, #idMo)  
**albumContient**(#idAl, #idMo, num)  
**playlistContient**(#idP, #idMo, num)  
**suitGroupe**(#pseudo, #idG)  
**suitUtilisateur**(#suit, #suivi)  
**historique**(#pseudo, #idMo, dateHeure)

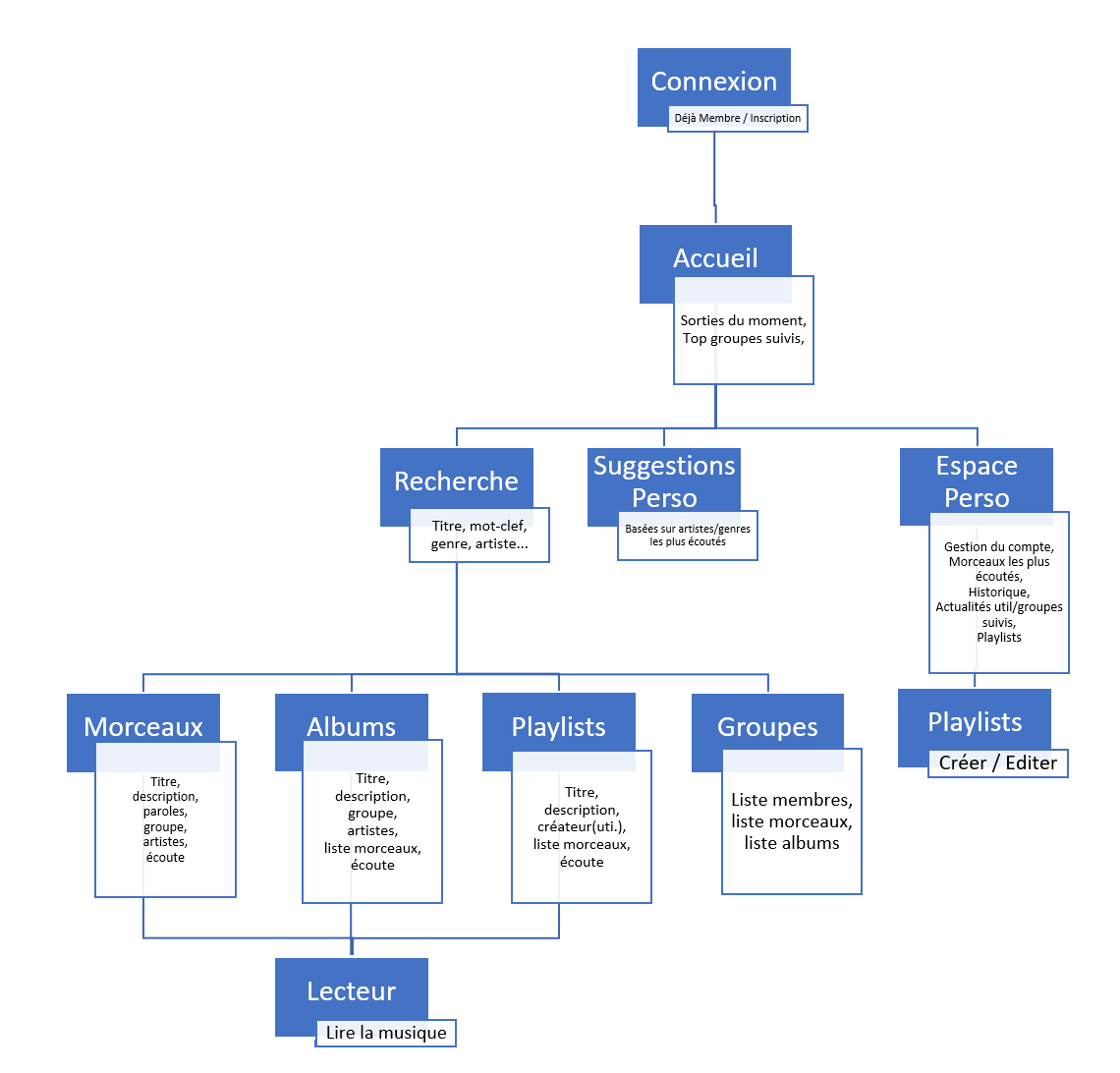
|  |  |
| --- | --- |
| **Clés étrangères** | **Colonne référencée** |
| * membre.idA * participe.idA | artiste.idA |
| * membre.idG * morceau.idG * album.idG * suitGroupe.idG | groupe.idG |
| * albumContient.idAl | album.idAl |
| * playlistContient.idP | playlist.idP |
| * playlistContient.idMo * historique.idMo * albumContient.idMo * participe.idMo | morceau.idMo |
| * suitGroupe.pseudo * playlist.pseudo * suitUtilisateur.suit * suitUtilisateur.suivi * historique.pseudo | utilisateur.pseudo |

Une **correction** par rapport au schéma EA a été apportée ici. En effet l’attribut de nationalité n’était pas dans l’entité artiste. Elle a été ajoutée dans le schéma relationnel,

# Architecture du site

## Arborescence des pages

La page d’authentification est le point d’entrée de l’utilisateur sur le site lors de sa première connexion. Bien que l’en-tête du site soit toujours visible, aucune page n’est accessible à l’utilisateur tant qu’aucune session n’est en cours. Après son authentification, la page d’accueil devient le nouveau point d’entrée de l’utilisateur sur le site.



## 

## Requêtes SQL majeures

Pour la partie recherche, on utilise la méthode de comparaison ILIKE qui ne tient pas compte de la casse des chaînes de caractères. Les requêtes SQL de la page de recherche sont :

* Groupes :

SELECT DISTINCT idg, nomg, datecrea, nationg, genre  
FROM groupe NATURAL JOIN membre NATURAL JOIN artiste  
WHERE nomg ILIKE '%' || :nomg || '%'  
AND genre ILIKE '%' || :genre || '%'  
AND ((noma ILIKE '%' || :artiste || '%' OR prenom ILIKE '%' || :artiste || '%')  
OR (:artiste ILIKE '%' || noma || '%' AND :artiste ILIKE '%' || prenom || '%'))  
ORDER BY nomg;

* Morceaux :

SELECT DISTINCT morceau.idmo, titrem, duree, genre, nomg  
FROM morceau  
LEFT JOIN groupe ON groupe.idg = morceau.idg  
LEFT JOIN participe ON morceau.idmo = participe.idmo  
LEFT JOIN artiste ON participe.ida = artiste.ida  
WHERE titrem ILIKE '%' || :titre || '%'  
AND genre ILIKE '%' || :genre || '%'  
AND ((noma ILIKE '%' || :artiste || '%' OR prenom ILIKE '%' || :artiste || '%')  
OR (:artiste ILIKE '%' || noma || '%' AND :artiste ILIKE '%' || prenom || '%')  
OR (noma IS NULL AND prenom IS NULL AND :artiste = ''))  
ORDER BY titrem;

* Albums :

**Si une date est mentionnée dans la recherche.**

SELECT idal, titrea, dateparu, couv, genre, nomg  
FROM (

SELECT DISTINCT album.idal, titrea, dateparu, couv, genre, nomg,   
dateparu - CAST(FLOOR(:date\*365.24) AS integer) AS depuis  
FROM album  
NATURAL JOIN albumcontient  
NATURAL JOIN morceau  
LEFT JOIN groupe ON groupe.idg = album.idg  
LEFT JOIN participe ON morceau.idmo = participe.idmo  
LEFT JOIN artiste ON participe.ida = artiste.ida  
WHERE titrea ILIKE '%' || :titre || '%'  
AND genre ILIKE '%' || :genre || '%'  
AND ((noma ILIKE '%' || :artiste || '%' OR prenom ILIKE '%' || :artiste || '%')  
OR (:artiste ILIKE '%' || noma || '%' AND :artiste ILIKE '%' || prenom || '%')  
OR (noma IS NULL AND prenom IS NULL AND :artiste = ''))  
ORDER BY depuis DESC

) AS tab1  
WHERE depuis < '0001-01-01' AND depuis > '0010-01-01 BC';

**Si une date n’est pas mentionnée dans la recherche.**

SELECT DISTINCT idal, titrea, dateparu, couv, genre, nomg  
FROM album  
NATURAL JOIN albumcontient  
NATURAL JOIN morceau  
LEFT JOIN groupe ON groupe.idg = album.idg  
LEFT JOIN participe ON morceau.idmo = participe.idmo  
LEFT JOIN artiste ON participe.ida = artiste.ida  
WHERE titrea ILIKE '%' || :titre || '%'  
AND genre ILIKE '%' || :genre || '%'  
AND ((noma ILIKE '%' || :artiste || '%' OR prenom ILIKE '%' || :artiste || '%')  
OR (:artiste ILIKE '%' || noma || '%' AND :artiste ILIKE '%' || prenom || '%')  
OR (noma IS NULL AND prenom IS NULL AND :artiste = ''))  
ORDER BY dateparu DESC;

* Playlists :

SELECT DISTINCT idp, titre, pseudo  
FROM playlist  
NATURAL JOIN playlistcontient  
NATURAL JOIN morceau  
LEFT JOIN groupe ON groupe.idg = morceau.idg  
LEFT JOIN participe ON morceau.idmo = participe.idmo  
LEFT JOIN artiste ON participe.ida = artiste.ida  
WHERE titre ILIKE '%' || :titre || '%'  
AND genre ILIKE '%' || :genre || '%'  
AND ((noma ILIKE '%' || :artiste || '%' OR prenom ILIKE '%' || :artiste || '%')  
OR (:artiste ILIKE '%' || noma || '%' AND :artiste ILIKE '%' || prenom || '%')  
OR (noma IS NULL AND prenom IS NULL AND :artiste = ''))  
ORDER BY titre DESC;

De manière générale, en utilisant la méthode ILIKE plutôt qu’une égalité entre chaînes de caractères, on cherche une correspondance partielle entre les mots-clés rentrés par l’utilisateur et les caractéristiques des entrées recherchées. Par exemple, pour un album, entrer « Olzon anette » dans le champ Artiste revient à rechercher tous les albums auxquels un artiste ayant participé porte un nom et un prénom contenus dans cette chaîne. S’il existe des artistes nommés Olzon Anette, Anette Olzon, Olzon Anette Olzon…etc, tous leurs albums seront affichés dans les résultats de la recherche.

On pourra trouver d’autres requêtes intéressantes dans la page de profil et celle des suggestions

Sur la page de profil, on pourra par exemple trouver celle-là :

SELECT idp, titre, pseudo, datemodif FROM playlist

WHERE pseudo IN (

SELECT suivi FROM suitutilisateur

WHERE suit = :pseudo

)

AND privee = False

ORDER BY datemodif DESC

LIMIT 10;

Cette requête utilise une sous-requête qui récupère tous les utilisateurs suivis, permettant de sélectionner les playlists publiques des utilisateurs suivis, en les triant par ordre décroissant par date de dernière modification. Ainsi, on a les 10 dernières playlists modifiées. La requête pour l’activité des groupes suivis fonctionne sur le même modèle.

Les requêtes de la page de recommandations personnalisées sont les plus compliquées du projet.

SELECT idmo, titrem, ida, noma, prenom FROM morceau

NATURAL JOIN participe

NATURAL JOIN artiste

WHERE ida IN (

SELECT ida as num FROM historique NATURAL JOIN participe NATURAL JOIN artiste

WHERE pseudo = :pseudo

AND age(dateheure) <= interval '7 days'

GROUP BY ida

ORDER BY count(\*) DESC

LIMIT 1

)

ORDER BY RANDOM()

LIMIT 5;

La sous-requête permet de trouver quel artiste est le plus écouté par l’utilisateur sur les 7 derniers jours. Elle part de l’historique de l’utilisateur et fait la jointure avec artiste en passant par participe qui fait l’association entre artiste et morceau, sélectionne uniquement les entrées des 7 derniers jours et utilise un agrégat pour trier par ordre décroissant d’apparition des artistes, et on sélectionne l’artiste en haut de la liste. La requête utilise cela pour sélectionner les morceaux dans lequel cet artiste participe, et en prend 5 au hasard.

WITH aux AS (

SELECT pseudo, count(DISTINCT idmo) as num FROM historique

WHERE idmo IN (

SELECT idmo FROM historique

WHERE pseudo = :pseudo

AND age(dateheure) <= interval '7 days'

)

GROUP BY pseudo

)

SELECT idg, nomg FROM utilisateur

NATURAL JOIN aux

NATURAL JOIN suitgroupe

NATURAL JOIN groupe

WHERE pseudo <> :pseudo

AND num >= (

SELECT num/5 FROM aux WHERE pseudo = :pseudo

)

GROUP BY idg, nomg

ORDER BY count(\*) DESC

LIMIT 10;

Cette requête est pour la partie « Groupes suivis par les utilisateurs écoutant les mêmes morceaux que vous ». Elle utilise de multiples sous-requêtes, y compris imbriquées. La sous-requête aux permet de compter pour chaque utilisateur le nombre de morceaux en commun entre leur historique et celui de l’utilisateur qui regarde la page sur les 7 derniers jours (y compris lui-même, permettant donc d’avoir le nombre de morceaux qu’il a écouté sur les 7 derniers jours).

La requête va ensuite retenir tous les utilisateurs ayant au moins 20 % de morceaux en commun, et va proposer les groupes que ces utilisateurs suivent, en utilisant un agrégat pour trier par nombre d’apparition du groupe dans l’ordre décroissant.

La dernière suggestion (les groupes suivis par les utilisateur suivant le groupe favori de l’utilisateur) se fait en 2 requêtes :

SELECT idg, nomg FROM historique

NATURAL JOIN morceau

NATURAL JOIN groupe

WHERE pseudo = :pseudo

GROUP BY idg, nomg

ORDER BY count(\*) DESC

LIMIT 1;

Celle-ci relativement simple, permet de trouver le groupe le plus écouté par l’utilisateur.

SELECT g1.idg, nomg, count(\*) as num FROM suitgroupe AS g1

JOIN suitgroupe AS g2 ON g1.pseudo = g2.pseudo

JOIN groupe ON g1.idg = groupe.idg

WHERE g2.idg = :idg AND g1.idg <> :idg AND g1.pseudo <> :pseudo

GROUP BY g1.idg, nomg

ORDER BY num DESC

LIMIT 10;

La 2ème procède par auto-jointure sur suitgroupe. On prend toutes les entrées où les utilisateurs suivent le groupe favori déterminé ci-dessus, ce qui permet d’avoir tous les groupes suivis par les gens qui suivent le groupe favori. De la même manière que les requêtes ci-dessus, on agrège et ont tri pour avoir ceux qui reviennent le plus.

## Gestion des sessions

Une session est démarrée quand l’utilisateur entre sur le site et est conservée entre toutes les pages. Quand l’utilisateur s’authentifie, une variable de session pseudo est initialisée, ce qui permet de savoir que l’utilisateur s’est bien connecté. La plupart des pages vérifient que cette variable de session soit déclarée pour permettre l’accès à la page, et l’utilise pour savoir le pseudo de la personne connectée.

Le code faisant cela étant répété partout, il est contenu dans un fichier .inc.php, et est inclus au début de toutes les pages protégées.

L’ID de sessions est régénéré sur chaque page pour des raisons de sécurité.

L’utilisateur peut également se déconnecter, ce qui aura la double action de supprimer la variable de session pseudo puis de détruire la session.

# Pistes d’amélioration

L’édition de playlist pourrait être améliorée. Elle ne permet que de rajouter des morceaux à la suite, en utilisant l’ID du morceau. On pourrait permettre d’ajouter des morceaux par leur nom, ainsi que d’ajouter des morceaux au milieu de la playlist. Enfin, il serait pratique de permettre de supprimer des morceaux au milieu de la playlist.

La sécurité sur le site est assez faible. Notamment, aucun encodage n’est fait pour les mots de passe et ceux-ci sont stockés en clair sur la base de données. Il serait judicieux de les encoder, avec par exemple, l’algorithme SHA-1.

Il pourrait également y avoir plus de contrôle sur les données entrées par l’utilisateur dans les formulaires. Une sécurité de base, notamment contre les injections SQL, est prévue par utilisation presque partout des méthodes prepare, bindParam et execute, mais on pourrait faire plus de contrôle sur les données entrées. De plus, il y a également peu de contrôle sur si les requêtes ont marché ou pas avant de les utiliser, un autre point à améliorer.

Bien que ce soit en dehors des spécifications demandées, on pourrait imaginer de rajouter une interface d’administration permettant de rajouter des groupes, des morceaux, des artistes, etc. Cela prendrait sans doute du temps.