L3 Informatique 2024-2025

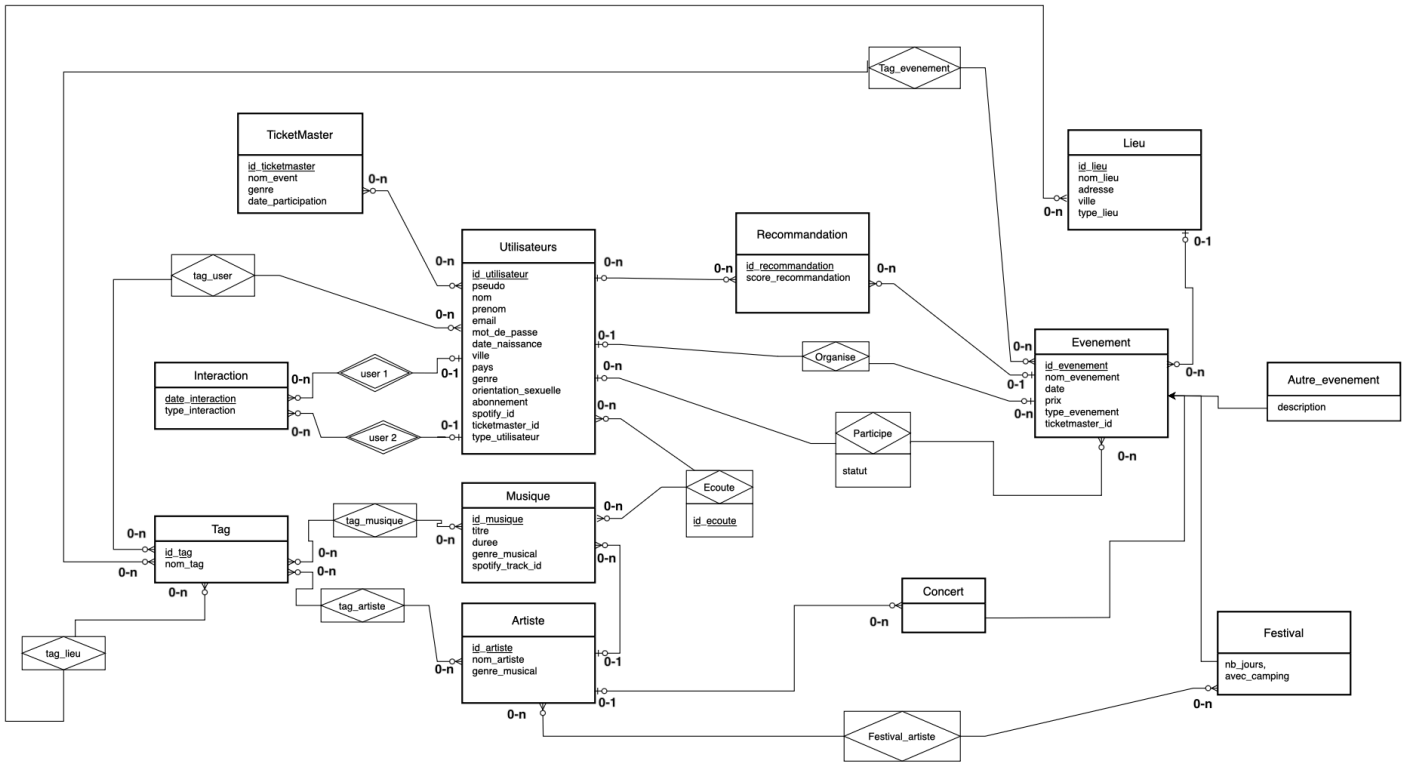
Projet Bases de Données (BD6)

Le Big Match : Big Brother Is Matching You

PERRIER-BABIN Ludivine 22200483
KACI Amel 22200653

Un Big Match Musical

Pour ce projet, nous avons décidé d'imaginer une application se basant sur les goûts musicaux des utilisateurs afin qu'ils puissent trouver le match parfait lors d'événements en lien avec la musique (concerts et festivals principalement).



Le modèle entités-associations

Le schéma relationnel

Utilisateurs (id_utilisateur, pseudo, nom, prenom, email, mot_de_passe, date_naissance, ville, pays, genre, orientation_sexuelle, abonnement, spotify_id, ticketmaster_id, type_utilisateur)
Lieu (id_lieu, nom_lieu, adresse, ville, type_lieu)
Artiste (id_artiste, nom_artiste, genre_musical)
Musique(id_musique, titre, #id_artiste, duree, genre_musical, spotify_track_id)

- Musique.id_artiste ⊆ Artiste.id_artiste

Ecoute(id_ecoute,#id_utilisateur,#id_musique)

- Ecoute.id_utilisateur ⊆ Utilisateur.id_utilisateur
- Ecoute.id_musique ⊆ Musique.id_musique

TicketMaster(id_ticketmaster, #id_utilisateur, nom_event, genre_musical, date_participation)

- TicketMaster.id_utilisateur ⊆ Utilisateur.id_utilisateur

Evenement(id_evenement, nom_evenement, date, #lieu, prix, #id_organisateur, type_evenement, ticketmaster_id)

- Evenement.id_organisateur ⊆ Utilisateur.id_utilisateur
- Evenement.lieu ⊆ Lieu.id_lieu

Participe (#id_utilisateur,#id_evenement, statut)
- Participant.id_utilisateur ⊆ Utilisateur.id_utilisateur
- Participant.id_evenement ⊆ Evenement.id_evenement

Concert (#id_evenement, #id_artiste)
- Concert.id_evenement ⊆ Evenement.id_evenement
- Concert.id_artiste ⊆ Artiste.id_artiste

Festival (#id_evenement, nb_jours, avec_camping)
- Festival.id_evenement ⊆ Evenement.id_evenement

Festival_Artiste (#id_evenement, #id_artiste)
- Festival_Artiste.id_evenement ⊆ Festival.id_evenement
- Festival_Artiste.id_artiste ⊆ Artiste.id_artiste

Autre_Evenement (#id_evenement, description)
- Autre_Evenement.id_evenement ⊆ Evenement.id_evenement

Interaction(#id_utilisateur_1, #id_utilisateur_2, date_interaction,type_interaction)
- Interaction.id_utilisateur_1 ⊆ Utilisateur.id_utilisateur
- Interaction.id_utilisateur_2 ⊆ Utilisateur.id_utilisateur

Recommandation (id_recommandation, #id_utilisateur, #id_evenement, score_recommandation)
- Recommandation.id_utilisateur ⊆ Utilisateur.id_utilisateur
- Recommandation.id_evenement ⊆ Evenement.id_evenement

Tag (id_tag, nom_tag)

Tag_lieu(#id_tag, #id_lieu)
- Tag_lieu.id_tag ⊆ Tag.id_tag
- Tag_lieu.id_lieu ⊆ Lieu.id_lieu

Tag_user(#id_tag, #id_user)
- Tag_user.id_tag ⊆ Tag.id_tag
- Tag_user.id_user ⊆ Utilisateur.id_utilisateur

Tag_artiste(#id_tag, #id_artiste)
- Tag_artiste.id_tag ⊆ Tag.id_tag
- Tag_artiste.id_artiste ⊆ Artiste.id_artiste

Tag_musique(#id_tag, #id_musique)
- Tag_musique.id_tag ⊆ Tag.id_tag
- Tag_musique.id_musique ⊆ Musique.id_musique

Tag_evenement(#id_tag, #id_evenement)
- Tag_evenement.id_tag ⊆ Tag.id_tag
- Tag_evenement.id_evenement ⊆ Evenement.id_evenement

Contrainte externes :
- Utilisateur.genre : ENUM (homme, femme, non-binaire, autre)
- Utilisateur.orientation_sexuelle : ENUM (hétéro, bi, gay, pan)
- Utilisateur.type_utilisateur : CHECK (organisateur, participant)
- Evenement.type_evenement : CHECK (concert, autre, festival)
- Participe.statut : CHECK (intéressé, participant)
- Interaction.type_interaction : ENUM (like, nope, match, message)
- Festival.nb_jours doit être >= 1 (CHECK)

- Utilisateur.spotify_id ou Utilisateur.ticketmaster_id ne peuvent pas être null en même temps car il faut se connecter avec au moins un des deux pour avoir un compte (CHECK)
- La suppression d'un utilisateur peut entraîner la suppression de ses participations, interactions, écoutes (ON DELETE CASCADE)
- La suppression d'un événement entraîne la suppression de ses concerts, participations, recommandations, festivals et leurs artistes
- La suppression d'un festival entraîne la suppression de ses enregistrements dans Festival_Artiste
- Les artistes associés à un festival ne sont pas supprimés automatiquement
- L'héritage pour Evenement est total sur Festival, Concert et Autre_Evenement

hors SGBD :
- les matchs ne sont possible que si le like est mutuel entre deux utilisateur
- un utilisateur ne peut pas like deux fois le même utilisateur deux fois
- les adresses mails, id spotify et ticketmaster, ville, artiste, musique et autre données doivent exister (pour une version réelle du projet)
- TicketMaster.date_participation doit être passée
- TicketMaster.id_utilisateur doit forcément avoir Utilisateur.ticketmaster_id non null pour avoir les informations
- Ecoute.id_utilisateur doit forcément avoir Utilisateur.spotify_id non null pour avoir les informations

La conception de la base

Tables principales

1. Utilisateur

- Contient toutes les informations de base d'un utilisateur (pseudo, nom, prenom, email, genre, orientation, abonnements, Spotify/Ticketmaster ID).
- Enum pour genre et orientation pour garantir des valeurs contrôlées.
- Différencié par type (participant ou organisateur) pour flexibilité.

2. Lieu

- Stocker les lieux de concerts, festivals ou autres.
- Indispensable pour la gestion géographique des événements.

3. Artiste

- Permet d'identifier les artistes liés aux musiques et concerts.
- Important pour tout l'univers musical de l'application.

4. Musique

- Stocker les morceaux écoutés par les utilisateurs.
- Relie les musiques aux artistes via une liaison directe.

5. Ecoute

- table d’association reliant Utilisateur et Musique.
- Permet d’analyser les goûts musicaux des utilisateurs pour le matching.

6. TicketMaster

- table qui permet de mettre les événements auxquels un utilisateur a déjà assisté via son compte ticketmaster
- permet d’avoir les genres musicaux qu’un utilisateur a déjà vu en concert

7. Evenement

- Entité de base pour tous les événements (concerts, festivals, autres).
- Contient toutes les informations générales (lieu, date, prix, organisateur).

8. Participe

- Entité faible reliant les participants aux événements.
- Statut : intéressé ou participant actif.

9. Concert

- Spécialisation d'Evenement dédiée aux concerts uniques.
- Relie un artiste à un événement unique.

10. Festival

- Nouvelle spécialisation d'Evenement pour les festivals musicaux.
- Spécificités : nb de jours, possibilité de camping, description étendue.

11. Festival_Artiste

- Table d'association entre Festival et Artiste.
- Permet d'associer plusieurs artistes à un festival et vice-versa.

12. Autre_Evenement

- Spécialisation d'Evenement pour d'autres types d'événements (ex: cinéma, théâtre, fanmeeting).

13. Interaction

- Gestion des interactions sociales : like, nope, match ou messages.
- Représente les choix d'utilisateur lors du swipe/matching.
- montre s'il y a eu des échanges entre eux

14. Recommandation

- Suggestions d'événements personnalisées pour chaque utilisateur.
- Basées sur les écoutes et préférences détectées.

15. Tag et les tables de tag associé

- Permet de taguer des utilisateurs, des événements, des lieux, des artiste ou des musiques.
- Système de mots-clés ouvert pour affiner le matching.

Le remplissage de la base

Pour remplir la table utilisateur on est directement passer par un csv propre généré automatiquement avec certains NULL pour l'id_ticketmaster car il en faut au moins un des deux mais pas forcément les deux, mais on a ajouté l'id spotify qui suit les règle de convention de l'api spotify pour tous les utilisateur pour avoir plus de jeux de données lié à l'écoute de musique. Pour la table Utilisateur, les mot_de_passe en hash car comme s' ils étaient passé par un code de hachage mais ne sont du coup que des chaînes de caractères.

A la suite pour la table lieu on a ajouté quelques lieux de France qui accueillent des évènements comme des salles de concerts etc. A l'aide d'un premier fichier csv pour avoir des salles de spectacle vivant en Ile-de-France. Puis avec un autre csv fait avec un générateur pour avoir des salles de concerts et sortir de l'ile-de-france.

Pour Artiste, nous avons généré des artistes divers et variés avec beaucoup de genres différents. La table Musique, on a pu prendre des chansons des artistes qu'on avait trouvé avec quelques id pour les liens spotify, la durée est en seconde pour que cela soit plus facile. Pour Ecoute, on a généré de tel sorte à ce qu'un utilisateur ai écouté des chansons différentes et certaines plusieurs fois pour avoir plus de de choses à compter pour les recommandation.

Pour TicketMaster, comme son nom l'indique, on voulait mettre en œuvre les données d'un utilisateur qui a bien voulu lié son compte ticketmaster pour avoir ses événements passés auxquels il a déjà assisté. On a donc ajouté des concerts qui ont déjà eu lieu pour avoir des genres et donc plus de matière pour les matchs possibles et recommandations.

La table Evenement était une grosse partie a faire mais nous avons dû inventer quelques concert ou autres pour match avec les artistes qu'on avait déjà dans la base de donnée.

Pour remplir Participe un fichier csv avec les id_utilisateur selon le csv d' Utilisateur et leur type_utilisateur, avec aussi l'id_evenenement et un statut choisi.

Ensuite la table Concert, en vérifiant le type_evenement on a ajouté des concerts pour les artistes. Dans le lot il n'y a que des concerts futurs. À la table festival, on a ajouté les événements de type festival, qui sont eux aussi dans le futur. Pour la table d'association Festival_Artiste on y a ajouté plusieurs artistes pour des festivals aléatoirement. Enfin pour Autre_Evenement, tous ceux qui avaient le type "autre" qui sont des pièces de théâtre, des show d'humoriste etc.

La table Interaction avec un csv simple générer nous-mêmes pour avoir des interactions déjà faites, quelques like, nope, matchs et messages entre des utilisateurs tout en gardant le lien entre le genre des utilisateurs et leur orientation sexuelle.

Dernière grosse table compliquée, Recommandation, les choses ont été très différentes car il a fallu choisir comment définir le score, on l'a donc rendu de 0 à 1 et plus proche on est de 1 plus l'événement est recommandé à l'utilisateur.

On s'est aidé d'un algo qui prend en compte :

- Historique d'écoute (musiques écoutées par chaque utilisateur)
- Préférences musicales (genres les plus écoutés)
- Événements à venir (concerts et festivals)

Avec une méthode de scoring :

- Match direct (90% du score) : Lorsqu'un utilisateur a écouté un artiste participant à un événement

  Score = 0.9 + (nb_écoutes x 0.02)

Ex: 3 écoutes → 0.9 + (3x0.02) = 0.96

- Match par genre (70% du score): Quand le genre de l'événement correspond aux préférences

  Score = 0.7 + (affinité_genre x 0.1) + random(0.0, 0.1)

Ex: Affinité forte → 0.7 + (0.2) + (0.05) =~ 0.95

- Tri par score décroissant
- Conservation des 3 meilleures recommandations par utilisateur
- Exclusion des utilisateurs déjà participants (10,15,20,25,30,35,40)
- Poids artistes écoutés > genres préférés
- Variabilité aléatoire pour diversifier
- Limite à 3 recommandations pertinentes

Statistiques :

- 41 utilisateurs traités
- 123 recommandations générées (3 par utilisateur)
- Répartition des scores :
  - 90%+ : 32% des recommandations
  - 80-89% : 58%
  - 70-79% : 10%

Pour finir on a rempli un peu les tables liées aux tags, avec quelques tag et leur lien direct avec les tables d'associations de tag pour Lieu, Musique, Artiste et Evenement.

Les requêtes implémentées

1. Événements à venir avec au moins 2 tags communs : Trouve les événements futurs où un utilisateur donné partage au moins 2 tags.
2. Utilisateurs habitant la même ville : Identifie les paires d'utilisateurs vivant dans la même ville (auto-jointure).
3. Événements avec tous les "likes" d'un utilisateur : Trouve les événements auxquels participent tous les utilisateurs qu'un utilisateur donné a likés (sous-requête corrélée).
4. Top 5 artistes les plus écoutés : Affiche les artistes avec le plus d'écoutes (sous-requête dans FROM).
5. Utilisateurs ayant écouté un artiste précis : Liste les utilisateurs ayant écouté au moins une musique d'un artiste spécifique (sous-requête dans WHERE).
6. Événements avec plus de 3 participants : Affiche les événements ayant plus de 3 participants (GROUP BY + HAVING).
7. Artistes avec plus de 5 écoutes : Liste les artistes ayant plus de 5 écoutes, triés par popularité (GROUP BY + HAVING).
8. Moyenne des maximums d'écoutes par utilisateur : Calcule la moyenne des écoutes maximales par utilisateur pour un artiste.
9. Événements et leurs lieux (même inconnus) : Affiche tous les événements, y compris ceux sans lieu (LEFT JOIN).
10. Utilisateurs ayant participé à tous les événements "autre" : Trouve les utilisateurs ayant participé à tous les événements de type "autre" (sous-requête corrélée).
11. Version alternative de la requête 10 : Même résultat mais avec GROUP BY et HAVING.
12. Gestion des NULL
    - A: Utilisateurs sans ticketmaster_id.
    - B: Utilisateurs avec ticketmaster_id différent d'une valeur (inclut les NULL).
13. Dates libres pour un lieu : Trouve les prochaines dates disponibles pour un lieu donné (requête récursive).
14. Top 3 utilisateurs par mois pour les messages : Affiche les 3 utilisateurs ayant reçu le plus de messages par mois (fenêtrage).
15. Musiques jamais écoutées : Liste les musiques sans aucune écoute (LEFT JOIN + NULL).
16. Moyenne d'événements par utilisateur : Calcule le nombre moyen d'événements auxquels un utilisateur participe.
17. Top 5 artistes par moyenne d'écoutes : Artistes avec la moyenne d'écoutes par musique la plus élevée.
18. Lieux utilisés pour >= 4 événements : Liste les lieux ayant accueilli au moins 4 événements.
19. Événements avec participants de villes différentes : Affiche les événements où des participants viennent de villes différentes.
20. Événements où deux personnes qui ont match sont déjà allé en même temps via ticketmaster.
21. Recommandation d'utilisateurs à rencontrer dans un événement déjà recommandé

L’algorithme de matching

L’algorithme de matching a pour objectif de recommander à un utilisateur les concerts les plus pertinents selon plusieurs critères pondérés. Il repose sur une approche multi-sources combinant :

- les centres d’intérêt via les tags partagés entre l’utilisateur et les événements
- les préférences musicales en identifiant les concerts d’artistes déjà écoutés
- la proximité géographique en privilégiant les événements dans la même ville que l’utilisateur
- les interactions sociales positives en mettant en avant les événements où des personnes likées ou matchées sont présentes.

Chaque critère contribue à un score global pondéré qui permet de classer les événements à recommander.
Pareil pour l’algorithme de matching d’utilisateurs mais qui renvoie au final des utilisateurs et non des événement avec un score global.

Questions d’éthique

Une telle application pourrait poser des problèmes d’atteinte à la vie privée au vue du nombre d’informations récoltées notamment des données sensibles comme l’orientation sexuelle ou l’identité de genre d’une personne. De plus, le traitement de ces données doit se faire en respectant les réglementations en vigueur dans les pays où elle est utilisée. Il faudra donc respecter une transparence totale vis-à-vis de l’utilisateur au moment où il rejoint l’application concernant la récolte et l’utilisation de ses données, son consentement étant essentiel, il faudra également garantir à l’utilisateur la possibilité à n’importe quel moment que ses données soient supprimées ou modifiées. On pourrait par ailleurs définir une durée limite d’inactivité d’un compte utilisateur au-delà de laquelle son compte (et les données y étant liées) sont supprimées, surtout avec les lois du type RGPD qui obligent maintenant les suppressions de comptes en cas d’inactivité pour la protection des données.
L’utilisateur doit également avoir la possibilité de masquer certaines informations sur son profil.
De plus les autres utilisateurs s’ils sont matcher à cause d’un événement cela veut dire que l’un comme l’autre sait où la personne sera à tel moment donné et pourra donc connaître sa position exacte sans forcément qu’elle souhaite vraiment le dire à tout le monde même avec des gens avec lesquelles elle ne voulait pas matcher.