SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Projet de PC3R :

Utilisation de l'API Deezer

BOUCHENAKI Habib 3522424

DEMANGEON Romain

MEYNIER Florent

L'application que nous avons souhaité implémenter est un site où chaque personne peut donner son avis sur une musique, expliquer une phrase d'une chanson, un texte en entier. Chaque musique sera récupérée, les paroles de la chanson pourront être écrite par un utilisateur.

Pour faire cela, nous allons utiliser l'API de Deezer afin de collecter les titres, les auteurs, les dates de sortie, toutes les informations dont nous aurons besoin. Il y aura la possibilité de faire une recherche par titre. Pour chaque recherche, nous appellerons utiliserons l'api afin de collecter les résultats correspondant à cette recherche. Chaque résultat de requête se présente sous forme de JSON, ce sera ensuite à nous de récupérer les éléments qui nous intéressent.

Un utilisateur pourra faire une recherche sur une musique et pourra donner son avis, commenter, écrire les paroles de la chanson ou même expliquer le texte. Il faudra donc au préalable avoir créée un compte et être connecté pour pouvoir poster un message. Les utilisateurs pourront « aimer » le message.

Lorsqu'un utilisateur arrive sur la page principale du site, il peut faire une recherche. S'il souhaite faire une recherche, il tape l'élément de sa recherche dans une boite de texte, et le résultat s'affichera à l'écran. Il peut donner son avis sur la musique. Pour cela il doit être connecté. S'il ne possède pas de compte, il devra en créer un en précisant son adresse e-mail et le mot de passe qu'il souhaite utiliser ainsi que le pseudonyme sous lequel il souhaite apparaître. S'il possède un compte, il pourra se connecter en utilisant son pseudonyme, et bien entendu écrire le mot de passe. Lorsqu'il est connecté, il y a une boite de texte où il peut écrire ce qu'il souhaite. Une fois son texte écrit, il clique sur « Valider » et son message est posté. Le pseudonyme et la date sont affichées en plus du texte.

Un utilisateur peut aussi « aimer » un commentaire. Pour cela il a la possibilité de cliquer sur « J'aime » et le nombre de « Like » sera incrémenté de « 1 » sous le message.

Par défaut les messages seront triés par date de publication décroissante, c'est à dire les messages les plus récents seront affichés en premier.

L'utilisateur aura la possibilité de se déconnecter de son compte grâce à un bouton « se déconnecter ».

Un utilisateur pourra directement écouter une musique depuis le site, écouter une playlist, copier le lien vers la playlist et même être directement redirigé sur le site « Deezer » en fonction d'où il a cliqué. Si c'est une playlist, il sera redirigé sur Deezer au niveau de la playlist, si c'est un artiste, la page Deezer de l'artiste s'affichera, etc.

Pour ces différentes fonctionnalités il y aura plusieurs bases de données :

Tout d'abord, il y aura une table qui stockera les informations relatives à chaque utilisateur tel que le nom d'utilisateur, l'adresse mail, le mot de passe mais aussi un identifiant unique afin de différencier chaque utilisateur.

Une table sera créée pour les musiques. Chaque musique aura un identifiant qui correspondra à celui de Deezer.

Une table commentaire sera créée. Chaque commentaire sera défini par un id unique. Les informations du commentaire telles que la musique commentée, l'id de l'utilisateur qui l'a posté, sa date de publication ainsi que le contenu du message seront stockées dans la base de donnée.

Une table music_like permet de stocker tous les likes sur les musiques. Chaque like est défini par l'id de la music ainsi que l'id de l'utilisateur qui a aimé.

Une table comment_like permet de stocker tous les likes sur les commentaires. Chaque like est défini par l'id du commentaire ainsi que l'id de l'utilisateur qui a aimé.

Dès qu'un nouveau commentaire est posté, les informations sont ajoutées à la base de donnée Commentaire.

Lorsqu'une recherche sera effectuée, un appel à l'API sera effectué et nous renverra un résultat sous forme de JSON. Si les différents résultats de la recherche ne sont pas présentes dans la base de donnée music, alors elles y sont ajoutées. Lorsqu'on souhaite afficher les informations d'une musique, un appel à l'API sera fait et nous renvoie toutes les informations relatives à cette musique.

Nous avons souhaité faire une approche ressource. En effet, les opérations et requêtes que nous allons effectué, feront soit appel à la base de donnée, soit un appel à l'API. Pour se faire, nous allons avoir une base de donnée comportant une table pour stocker les utilisateurs, une table pour stocker les musiques, une table pour stocker les commentaires ainsi qu'une table pour stocker les likes des musiques et des commentaires. De plus, nous ferons des appels à l'API pour collecter les informations dont nous avons besoin.

Nous avons un composant Authentification qui va gérer les connections et déconnections des users ainsi que les sessions. Un autre composant Musique qui permet de gérer les musiques et ses likes et pour finir un composant Commentaire qui permet de gérer les commentaires qui sont postés par les utilisateurs ainsi que les likes.

Le site va comporter 1 seul page. Par défaut, rien n'est affiché sur la page, sauf les boutons de connections/déconnections et inscriptions ainsi que la barre de recherche. Lorsqu'une recherche est effectuée, le résultat s'affiche sur la page. La liste des musiques est affichée avec quelques informations tels que le titre, l'auteur et l'album. Lorsqu'une musique est sélectionnée (on a cliqué dessus), le résultat de la recherche se supprime et les informations relatives à cette musique s'affiche à l'écran. Il y a la possibilité d'écouter un échantillon de la musique, ainsi qu'un bouton qui permet de se rediriger sur la page Deezer relative à cette musique. De plus, le nombre de like est affichée et si l'utilisateur n'a pas déjà aimé cette musique, il peut le faire. La liste des commentaires de cette musique est affichée et un utilisateur a la possibilité de poster un commentaire s'il est connecté et aimer un commentaire s'il ne l'a pas déjà fait.

Liste des appels AJAX :

- Bouton « Sign up » : Ouvre le formulaire de création de compte
    - « Register » : Crée un compte à partir des informations fournies
- Bouton « Sign in » : Ouvre le formulaire de connexion
    - « Connect » : Connecte l'utilisateur si les informations sont valides
- Bouton « Disconnect » : Déconnecte l'utilisateur
- Bouton « Search » : Effectue une recherche de musique

Dans le résultat d'une recherche :

- Clique sur l'image de la musique : Affiche la page de la musique

Sur la page d'une musique :

- Clique sur le logo Deezer : Ouvre une nouvelle fenêtre correspondant à la page Deezer de la musique
- Clique sur le bouton Play du lecteur : Lance un échantillon de la musique
- Bouton « Like music » : Permet de liker la musique
- Bouton « Post Comment » : Permet de poster un commentaire
- Bouton « Like » : Permet de liker un commentaire

Description des requêtes :

Utilisateur :

- GET id → récupère un utilisateur grâce à son id

← { login : *login* }

- POST id pseudo mail pwd → ajouter un utilisateur

Connection :

- GET is_Session → récupère l'id d'un utilisateur grâce à l'id de session

← { idSession : W424dxsaWXds5Q21… }

- POST login → connecte un utilisateur

← { idSession : W424dxsaWXds5Q21… }

- DELETE login → déconnecte un utilisateur

← { }

Commentaire :

- GET id_user → récupère tous les commentaires d'un utilisateur

← { result : [{id : 1, idMusic : 2772, idUser : 1, DateP : 12:50pm 2021/05/22, Msg : "j'aime"}]}

- GET id_music → récupère tous les commentaires d'une musique

← { result : [{id : 1, idMusic : 2772, idUser : 1, DateP : 12:50pm 2021/05/22, Msg : "j'aime"}]}

- POST id id_user id_music → ajouter d'un commentaires

- POST id like dislike → mettre à jour le nombre de Like et Dislike

Like_music :
- GET id_user → récupère les like d'un utilisateur
← { result : [{ idMusic : 2772, idUser : 1 } }

- GET id_music → récupère les like d'une musique
← { result : [{ idMusic : 2772, idUser : 1 } }

- POST id_user id_music → ajouter un like

Like_comment :

- GET id_user → récupère les like d'un utilisateur
← { result : [{ idMusic : 2772, idUser : 1 } }

- GET id_music → récupère les like d'une musique
← { result : [{ idMusic : 2772, idUser : 1 } }

- POST id_user id_music → ajouter un like