Spotifake

I. Pitch

L’application que nous allons mettre en place permettra d’offrir une expérience musicale. Les utilisateurs connectés pourront découvrir des albums, écouter et gérer leur musique préférée.
Dès leur inscription, les utilisateurs auront accès à un compte personnel, ils pourront gérer leurs informations, réinitialiser leur mot de passe et même supprimer le compte si nécessaire.
Plusieurs fonctionnalités seront mises à disposition de l’utilisateur, notamment la création de playlists personnelles et ainsi ajouter ou supprimer des morceaux pour créer des playlists sur mesure selon les goûts de chacun.
La fonctionnalité de recherche sera l’élément clé de notre application. En effet, la recherche sera possible selon le titre, l’artiste ou même l’album ! Il sera donc facile de trouver le morceau qui vous plaît.
La possibilité d'avoir une vue détaillée d'un album ainsi que la liste des morceaux associés vous permettra de plonger dans l'univers musical de vos artistes préférés. D’autres fonctionnalités comme la lecture des titres depuis la page d'accueil, la possibilité d'arrêter la lecture à tout moment et même de passer automatiquement à la chanson suivante si vous le souhaitez seront un plus pour une expérience musicale fluide et sans interruption.

[ENGLISH VERSION]
The application we are developing aims to provide a music-centric experience. It will allow logged-in users to discover albums, listen to and manage their favorite music. Upon registration, users will have access to a personal account where they can manage their information, reset their password, and even delete their account if needed.
Several features will be available to the users, including the creation of personalized playlists where they can add or remove tracks to curate their own music collections according to their tastes.
The search functionality will be a key element of our application. Users will be able to easily search for specific tracks, artists, or albums, making it effortless to find the desired music. From search results, users will have the option to add tracks to their playlists.
Detailed album views, including track lists, will immerse users in the musical universe of their favorite artists. Our application offers a complete music experience, allowing users to play tracks directly from the homepage, stop playback at any time, and even automatically transition to the next track if desired.

II. Cahier des Charges

1. Recueil des besoins

- Compte utilisateur
    - Inscription : Nom, prénom, email, mot de passe
    - Gestion de l'authentification
    - Réinitialiser le mot de passe
    - Bonus: modification d'info de compte
    - Bonus : suppression de compte

- Recherche titre
    - Après la connexion accès à une page d'accueil permettant la recherche de pistes audio par titre, par nom d'artiste ou par album.
    - Pouvoir filtrer les résultats de la recherche par style ou artiste.
    - Accéder à la page, les informations et tracklists de l' album.

- Lecture des titres
    - Lecture des morceaux : permettre aux utilisateurs de lire des morceaux à partir de la page d'accueil, d'une recherche ou d'un album.
    - Arrêter une lecture
    - Bonus: Passage automatique à la chanson suivante si ce n'est pas la dernière chanson de l'album
    - Bonus: Passer à une musique suivante ou précédente si dans un album

- Playlist
    - Création de playlists.
    - Ajout de titre.
    - Retrait de titre.
    - Renommer la playlist
    - Supprimer playlist
    - Bonus: Enchaînement des titres dans la playlist
- Interface
    - Responsive: Ordinateur, Tablette, mobile
    - Navigation intuitive : permettre aux utilisateurs de naviguer de manière simple entre les différentes pages de l'application.

- Sécurité
    - Gestion de droits : authentification obligatoire.
    - Protection des données et ressources non accessibles
    - Protections contre les failles (injections).

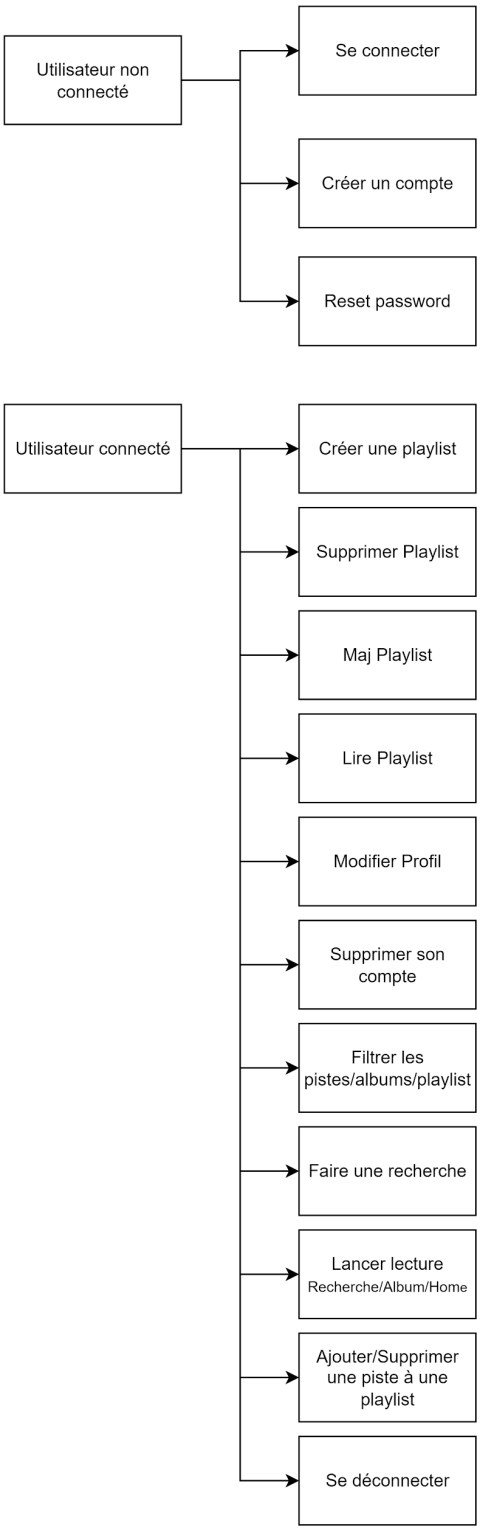
- Ressources
    - Intégration AWS S3 : permettre l’accès aux fichiers audio stockés sur le service AWS.

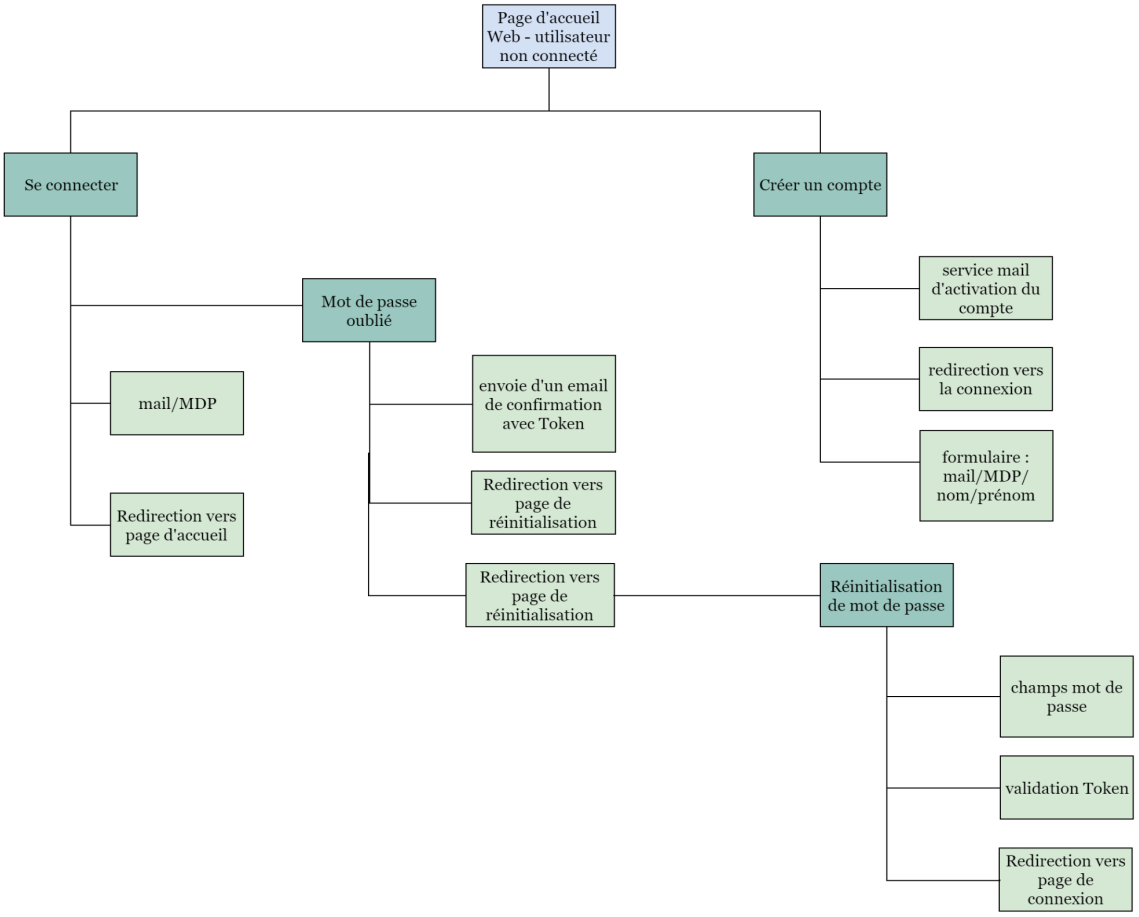
- Déploiement

    - L’application doit être déployée sur Plateforme EC2 avec accès SSH.

III. Analyse fonctionnelle

Découpage des fonctionnalités




Page d'accueil Web - utilisateur non connecté
Se connecter
Créer un compte
Mot de passe oublié
mail/MDP
Redirection vers page d'accueil
envoie d'un email de confirmation avec Token
Redirection vers page de réinitialisation
Redirection vers page de réinitialisation
service mail d'activation du compte
redirection vers la connexion
formulaire : mail/MDP/ nom/prénom
Réinitialisation de mot de passe
champs mot de passe
validation Token
Redirection vers page de connexion

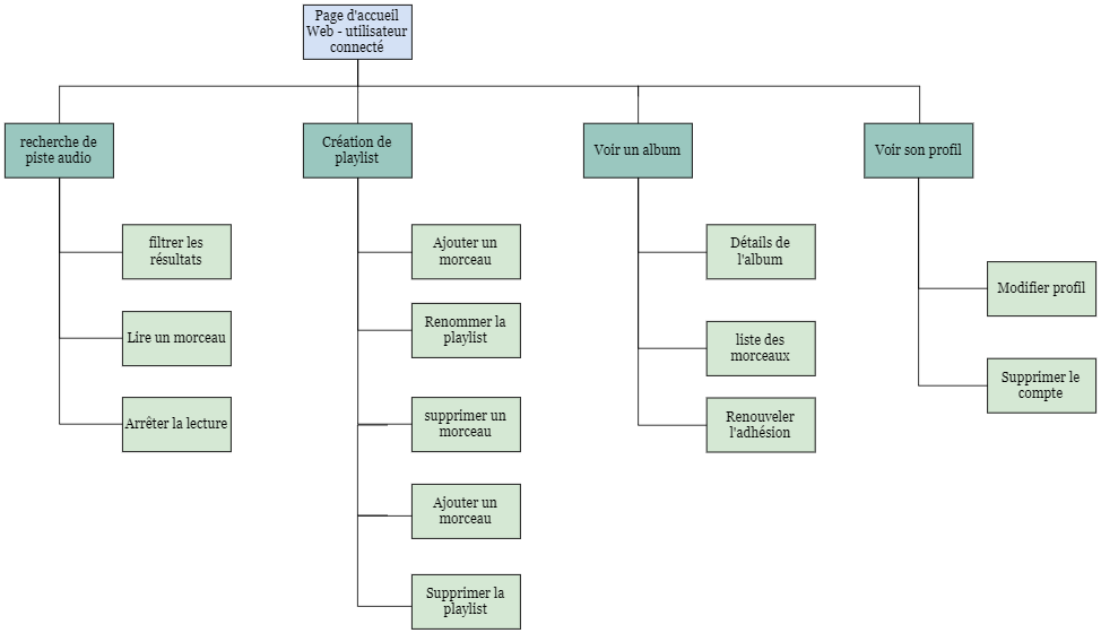

Page d'accueil Web - utilisateur connecté
recherche de piste audio
Création de playlist
Voir un album
Voir son profil
filtrer les résultats
Lire un morceau
Arrêter la lecture
Ajouter un morceau
Renommer la playlist
supprimer un morceau
Ajouter un morceau
Supprimer la playlist
Détails de l'album
liste des morceaux
Renouveler l'adhésion
Modifier profil
Supprimer le compte

Diagramme de flux

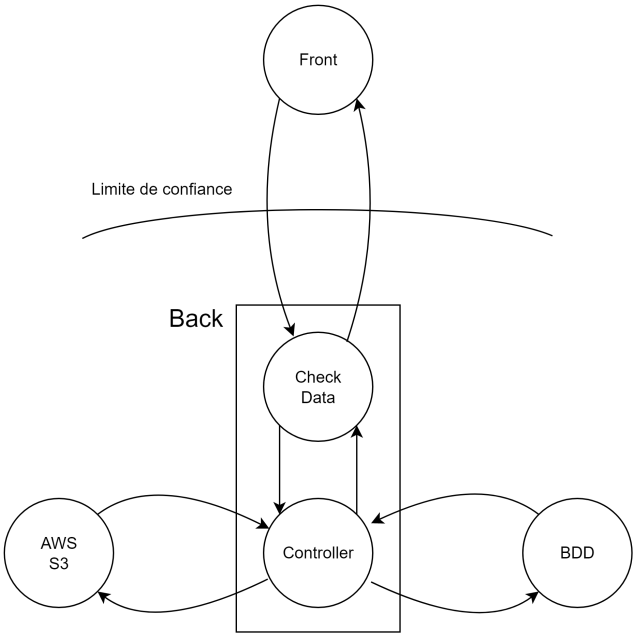
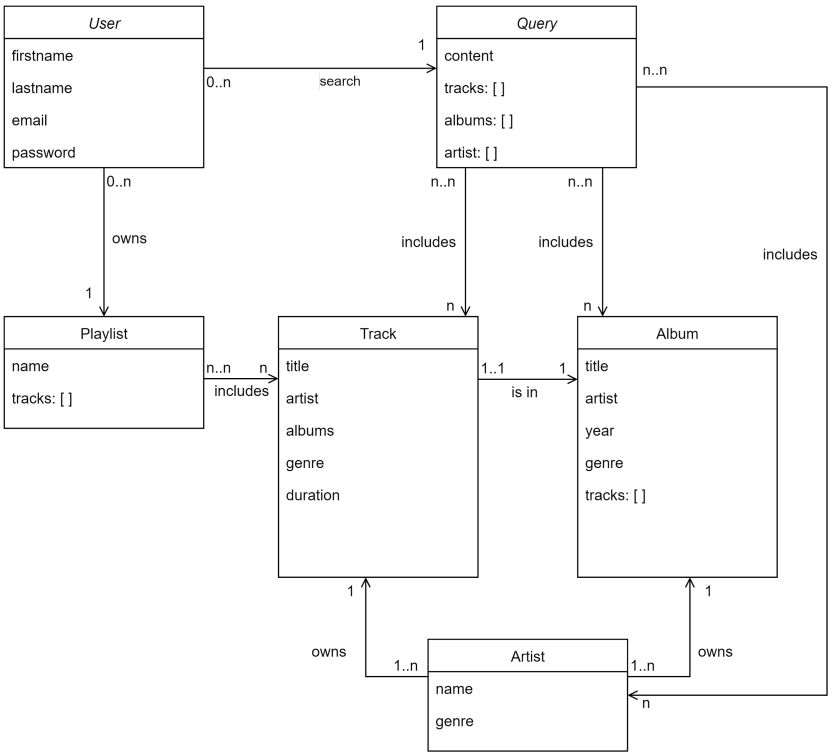


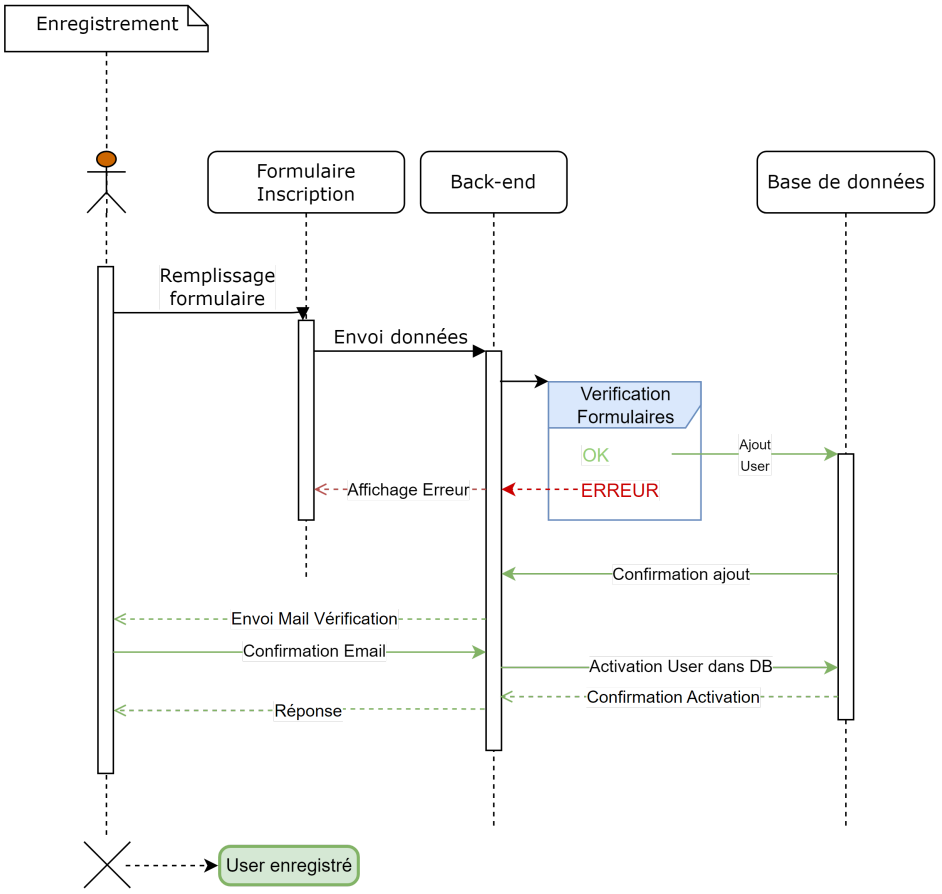
Diagramme de classes Utilisateur

Base de données

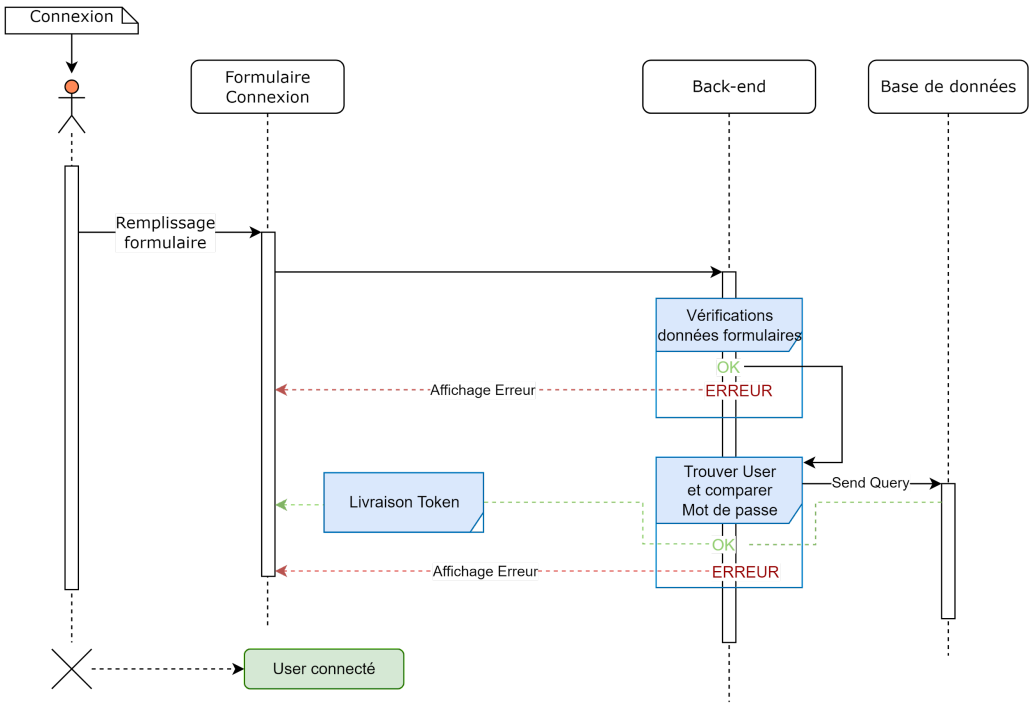


Diagrammes de séquences

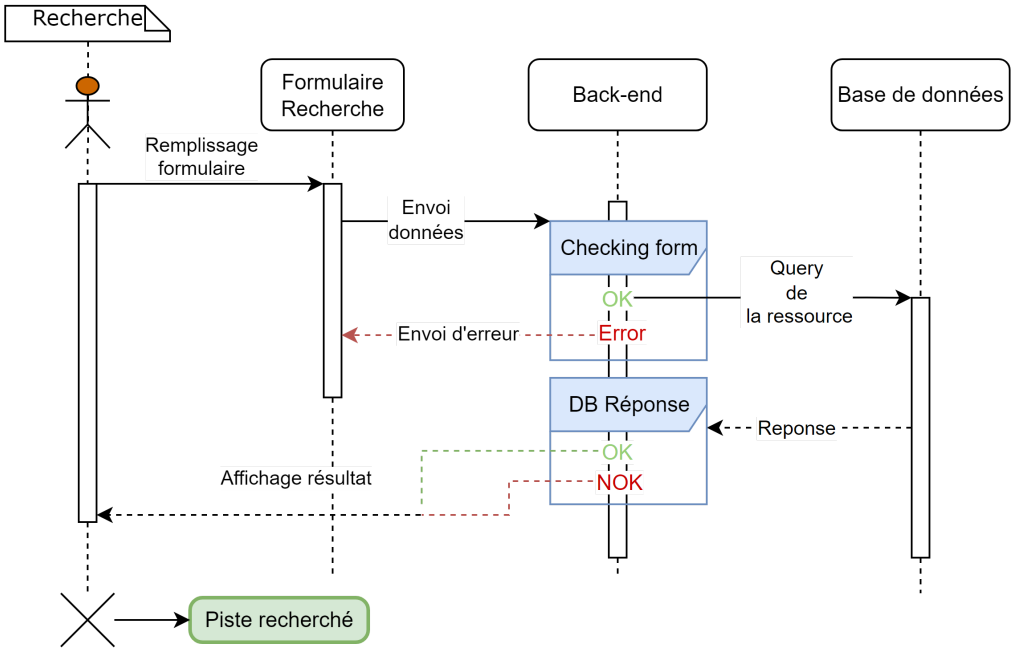
Enregistrement Utilisateur



Login Utilisateur



Recherche titre



Choix des technos

React

React est une bibliothèque JavaScript open-source développée par Facebook. Elle est utilisée pour créer des interfaces utilisateur interactives et réactives. Voici quelques raisons pour lesquelles nous avons choisi React pour notre projet :

1. Composants réutilisables : React fonctionne sur le concept de composants réutilisables. Il divise l'interface utilisateur en petits morceaux appelés composants, qui peuvent être réutilisés à plusieurs endroits de l'application.

2. Virtual DOM : React utilise un Virtual DOM (Document Object Model) léger pour représenter l'état de l'interface utilisateur.

3. Unidirectionnel et gestion efficace de l'état : React encourage l'utilisation d'un flux de données unidirectionnel, ce qui facilite le suivi de l'état de l'application.

4. Écosystème solide : React bénéficie d'un écosystème solide avec de nombreuses bibliothèques tierces, des outils de développement avancés et une communauté active.

TypeScript

TypeScript est un langage de programmation open-source développé par Microsoft. Il est basé sur JavaScript et ajoute des fonctionnalités de typage statique au langage. Voici pourquoi nous avons choisi TypeScript pour notre projet :

_ Typage statique : TypeScript permet de déclarer explicitement les types des variables, des paramètres de fonction et des retours de fonction. Cela aide à détecter les erreurs de typage avant l'exécution du code, améliorant ainsi la qualité du code et réduisant les erreurs potentielles.

_ Meilleure prise en charge des IDE : Les IDE et les éditeurs de code prennent en charge TypeScript et fournissent des fonctionnalités avancées telles que l'autocomplétion, la vérification statique des types et la documentation intégrée.

_ Évolution progressive : TypeScript est compatible avec JavaScript, ce qui signifie que nous pourront migrer progressivement notre code JavaScript existant vers TypeScript sans avoir à tout réécrire.

_ Écosystème et documentation : TypeScript bénéficie d'un vaste écosystème de bibliothèques JavaScript existantes, qui peuvent être facilement utilisées dans les projets TypeScript.

Node.js

Node.js est un environnement d'exécution JavaScript côté serveur, basé sur le moteur JavaScript V8 de Google Chrome. Il permet d'exécuter du code JavaScript en dehors d'un navigateur web. Voici quelques points clés qui expliquent pourquoi nous avons choisi Node.js pour notre projet :

_ JavaScript côté serveur : Node.js permet d'utiliser JavaScript tant du côté client que du côté serveur, ce qui simplifie le développement d'applications web complètes en utilisant un seul langage de programmation cohérent.

_ Événements et architecture orientée vers les E/S : Node.js est conçu pour gérer efficacement les opérations d'entrée/sortie (E/S) non bloquantes grâce à son modèle d'événements. Au lieu d'attendre les résultats d'une opération E/S, Node.js continue d'exécuter d'autres tâches, ce qui permet de gérer efficacement de nombreuses connexions simultanées.

_ Évolutivité : Node.js est conçu pour être évolutif, permettant de gérer facilement un grand nombre de connexions simultanées. Il utilise le modèle d'architecture orientée vers les événements et l'évolutivité horizontale pour fournir des performances élevées et une grande capacité de charge.

_ Vaste écosystème de modules : Node.js dispose d'un vaste écosystème de modules et de packages prêts à l'emploi grâce à son gestionnaire de paquets appelé npm (Node Package Manager).

Express

Express est un framework web minimaliste et flexible pour Node.js. Il simplifie le développement d'applications web en fournissant des fonctionnalités et des outils essentiels tout en laissant une grande liberté aux développeurs.

_ Facilité d'utilisation : Express est connu pour sa simplicité et sa facilité d'utilisation. Il offre une syntaxe simple et intuitive pour la définition des routes, la gestion des requêtes et des réponses, ainsi que la mise en place de middleware.

_ Routing flexible : Express facilite la définition des routes et la gestion des différentes requêtes HTTP (GET, POST, PUT, DELETE, etc.). Il permet de créer des endpoints pour répondre aux différentes actions de l'application et d'effectuer des opérations spécifiques en fonction des URL et des méthodes.

_ Middleware puissant : Express est construit autour du concept de middleware, qui permet d'ajouter des fonctionnalités supplémentaires à l'application. Il dispose d'une vaste bibliothèque de middleware prêt à l'emploi, permettant d'ajouter des fonctionnalités telles que la gestion des sessions, la compression des données, l'authentification, etc.

_ Large écosystème : Express bénéficie d'un vaste écosystème de modules et de packages développés par la communauté, grâce à son intégration étroite avec Node.js. Cela facilite l'intégration de fonctionnalités supplémentaires dans notre application en utilisant des packages existants. Aussi Express permet une personnalisation avancée et est reconnu pour sa légèreté et sa performance

MongoDB

MongoDB est une base de données NoSQL open-source, orientée documents. Elle utilise un modèle de données flexible basé sur des documents JSON (JavaScript Object Notation) pour stocker et organiser les données. Voici pourquoi nous avons choisi MongoDB pour notre projet :

_ Structure flexible des données : MongoDB permet de stocker des données de manière flexible sans schéma rigide. Cela signifie que nous pouvons stocker des données de différents types et structures dans une collection, ce qui facilite l'évolution et la modification des schémas de données au fil du temps.

_ Évolutivité horizontale : MongoDB est conçu pour fonctionner sur des architectures évolutives horizontalement, ce qui permet d'ajouter facilement de nouveaux serveurs et de répartir la charge de travail.

_ Requêtes puissantes et flexibles : MongoDB fournit un langage de requête riche qui permet d'effectuer des recherches avancées et des agrégations de données. Il prend en charge des opérations telles que les filtres, les projections, les jointures, les tris et les regroupements, offrant ainsi une grande flexibilité pour interroger et analyser les données.

_ Écosystème et intégration : MongoDB dispose d'un écosystème solide avec une grande variété d'outils, de bibliothèques et de frameworks qui facilitent l'intégration avec d'autres technologies.

_ Haute disponibilité et tolérance aux pannes : MongoDB offre des fonctionnalités de réplication et de tolérance aux pannes qui garantissent la disponibilité continue des données même en cas de défaillance d'un ou plusieurs nœuds du cluster.

Mongoose

Mongoose est une bibliothèque JavaScript pour Node.js qui facilite l'utilisation de MongoDB dans nos applications. Elle fournit une couche d'abstraction supplémentaire et des fonctionnalités supplémentaires pour interagir avec la base de données MongoDB de manière plus conviviale. Mongoose permet : une modélisation des données, une validation

des données, un mapping Objet-Document, une gestion des relations et une gestion des middlewares et des requêtes

Global

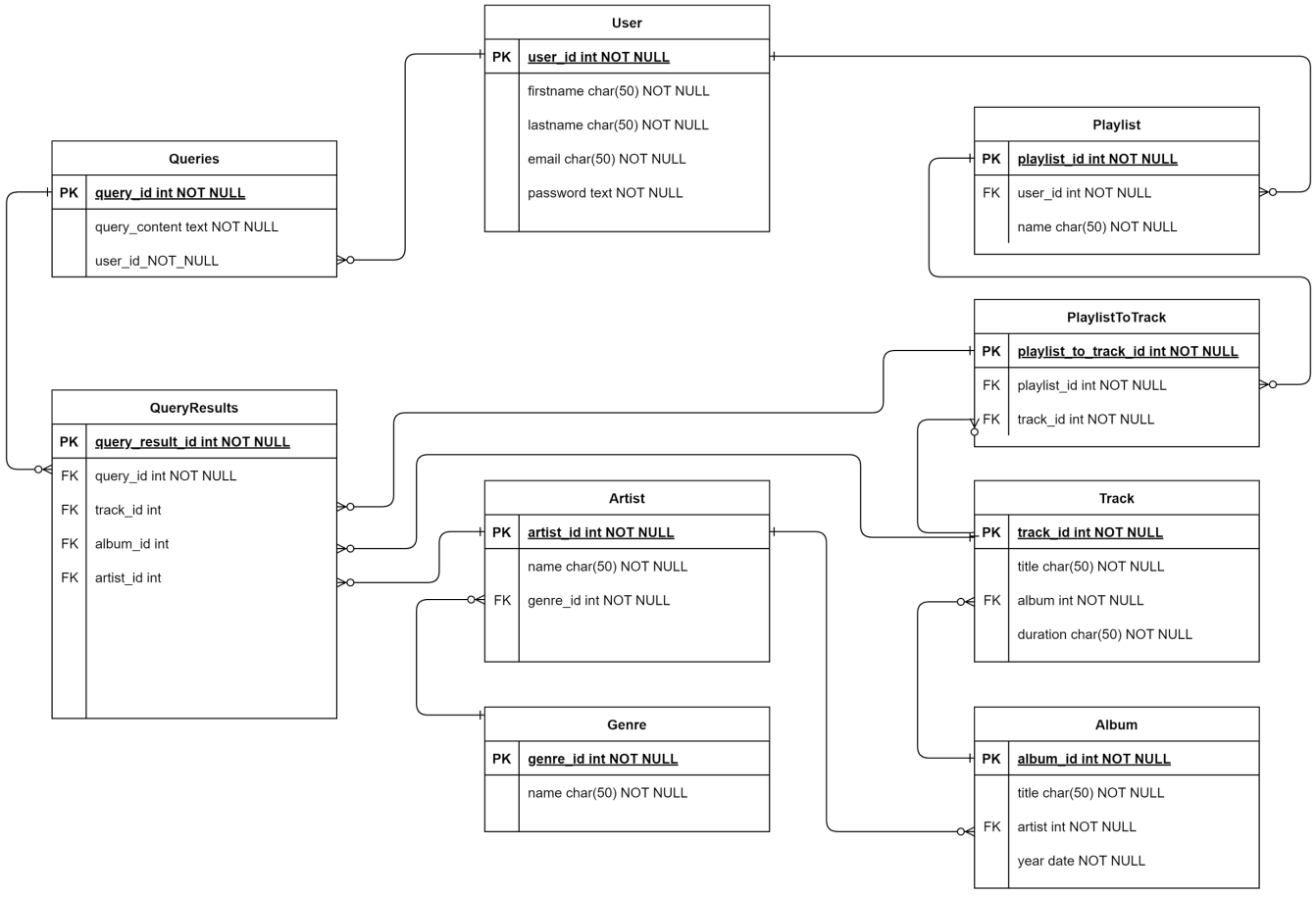
Typescript

Configuration d’un monorepo.

Base de données

ADR

relationnel :



Back

Express / Mongoose

Front

React

Sécurité :

En ce qui concerne la sécurité, nous avons pris plusieurs mesures pour assurer la protection de notre application :

- **Protection contre les failles XSS par défaut avec React.**

En ce qui concerne la sécurité avec React, nous bénéficions de mesures de protection intégrées par défaut, notamment contre les failles XSS (Cross-Site Scripting). Grâce à la configuration par défaut de notre application React, nous prévenons efficacement les attaques de ce type, ce qui renforce la sécurité de notre application.

- **Validation des données utilisateurs.**

Nous mettrons en place des validateurs pour vérifier que les données envoyées par les utilisateurs respectent le format attendu. Cela nous permet de prévenir les erreurs de formatage et de limiter les risques liés à des données incorrectes ou malveillantes.

- **Gestion des droits et authentification avec tokens.**

Nous utiliserons un système d'authentification basé sur des tokens pour gérer les droits d'accès des utilisateurs. Chaque utilisateur dispose d'un token d'authentification qui a une durée de validité d'une heure. De plus, nous utilisons également un refresh token qui a une durée de validité de 24 heures. Ce système nous permet de contrôler l'accès aux ressources de manière sécurisée et de limiter les risques d'accès non autorisés.

- **Sécurisation des routes avec un middleware.**

Nous mettrons en place un middleware qui sécurise l'accès aux routes de notre application. Ce middleware vérifie à chaque requête si l'utilisateur est autorisé à accéder à la ressource demandée. Cela nous permet de restreindre l'accès uniquement aux utilisateurs légitimes et d'éviter les attaques d'accès non autorisées.

- **Tests unitaires et fonctionnels.**

Nous avons mis en place des tests unitaires et fonctionnels en utilisant des outils tels que Postman ou Insomnia. Ces tests nous permettent de vérifier la robustesse de notre application du point de vue de la sécurité. Ils nous aident à identifier les éventuelles vulnérabilités et à les corriger avant le déploiement en production.

En prenant toutes ces mesures de sécurité, nous visons à garantir la protection des données de nos utilisateurs et la sécurité globale de notre application.

1. React : protection faille XSS par défaut.
2. Mise en place des validateurs pour s’assurer que le format des données envoyées par les utilisateurs est correct.

3. yGestion des droits et authentification avec un token (1h) et d’un refresh token (24h)
4. Les routes seront sécurisées par un middleware qui vérifiera à chaque fois si un utilisateur est autorisé pour cet accès ou non.
5. Des tests unitaires et fonctionnels (Postman / Insomnia)

IV. Gestion du projet

L’équipe

Règles et bonnes pratiques

Nous nous sommes imposés règles pour respecter de bonnes pratiques concernant les créations des branches :

- Le nom de la branche commencera soit:
    - par “feature/” : pour le développement de nouvelles fonctionnalités.
    - par “fix/” : pour les corrections d’erreurs.
    - par “refacto/” : pour les refactorisation de code.0

- Nous utiliserons le kebab-case pour séparer les mots dans le nom de la branche, ce qui signifie que les mots seront écrits en minuscules séparés par des tirets “-”.
- Les noms des branches seront le plus proches possible du titre de la tâche.
- Les branches sont rédigées en anglais pour assurer la cohérence avec le code / les commentaires.
- Les commits seront écrit eux aussi en anglais avec le format suivant

feat | fix | refacto : Short description.

**optionnel*
Long description.

    - feat | fix | refacto : obligatoire, pour indiquer si c’est une feature ou un fix du code.
    - Short description. : obligatoire, description courte du commit.
    - Long description. : optionnel, détails du commit.

Nous nous conformerons donc à ces règles et bonnes pratiques lors de la création de nouvelles branches pour maintenir un processus clair et organisé dans notre processus de développement.

Bonne pratique:

Méthodologie

Pour notre projet, nous avons choisi d'adopter la méthodologie de développement agile, plus précisément la méthode Scrum.

Nous sommes 4 développeurs full stack, cette méthodologie nous permettra :

1. Pour nous assurer du bon déroulement du sprint, nous avons défini une personne en tant que scrum master : Thierry Agnelli.
1. D'être flexible et de s'adapter : nous pourrons facilement adapter notre façon de travailler au fur et à mesure que l'on avancera dans le projet. Nous avons donc choisi des sprints de 1 semaine.
2. Niveau communication, nous avons mis en place un Jirapour une création de tâches en continu, une visibilité sur les tâches attribuées pour chaque développeur.
   Nous pensons que deux rendez-vous par semaine de deux heures à quatre nous permettra de faire un état des lieux de l'avancement, clore un sprint, et favorise ainsi l'entraide, la planification des tâches et ajuster les tâches au besoin.
3. Livrer les fonctionnalités utilisables à la suite de chaque sprint. Cela permettra de valider ou d'ajuster nos conceptions faites lors du premier sprint, réduisant ainsi les risques de dérive par rapport aux attentes initiales pour ce projet.
4. Lors de notre rendez-vous de fin de semaine, nous ferons une rapide rétrospective pour évaluer comment cela s'est passé, et comment améliorer le prochain sprint. Cela permettra d'augmenter l'efficacité de l'équipe et de ne pas reproduire les erreurs.(principe d'évaluation continue).
5. Étant donné qu'il s'agit d'un projet d'étude, il ne connaîtra pas d'évolution. Ainsi, le rôle de Product Owner ne sera pas nécessaire.

En conclusion, en adoptant la méthodologie agile , nous pensons que notre équipe pourra gérer les défis du projet de manière plus flexible, favoriser la collaboration et la communication, fournir des résultats concrets et s'améliorer continuellement. Cette approche nous permettra de mieux répondre aux attentes du projet tout en offrant une expérience de développement enrichissante pour chaque membre de l'équipe.

Liens utiles:

1. https://community.lebocal.academy/private/resources/Cahier%20des%20charges%20DW%20Nice%20S7.pdf
2. https://dptechinc.atlassian.net/jira/software/projects/SPOTI/boards/3
3. https://github.com/Le-Bocal-Academy/spotifake-rage-against-the-machine-code
4. LIEN S3 (à venir)

V. Spécifications Technique

Outils de développement :

- Docker
- Vs Code
- Insomnia / Postman
- MongoDB Compass

Configuration global :

repo Github :

Une organisation : spotifake-rage-against-the-machine-code
Un seul repo contenant les 2 projets back front.

Une pipelin de CI a été mise en place lors des pull request, elle vérifie le format du code (lint), que les tests passent et que les builds se font correctement.

Conteneurisation :

1 Docker-compose, 2 dockerfile

Commune

Configuration d'un monorepo NX :

- Simplification des pipelines
- Mutualisation des dépendances communes

Eslint : Règles de formatage de code

- indentation obligatoire
- point-virgule, obligatoire
- double-quotes
- longueur maximum d'une ligne : 120 caractères
- pas plus d'une ligne vide.
- pas de ligne vide en début de fichier
- une ligne unique et obligatoire en fin de fichier

Back

Test unitaires :

- Jest : bibliothèque de test
- supertest : pour mock des requêtes

Front

test unitaires:

- Vitest : bibliothèques de test (développée par l'équipe de vite)
- react-testing-library : simulation des actions utilisateurs

charte graphique

Code couleur :
- Mauve : #2C1E4A
- Marron : #AE5310
- Dégradé button : #8C52FF #5CE1E6
- Couleur police : #6BE5F3 et white

Font: TT Norms

Link css : `@import url('https://fonts.cdnfonts.com/css/tt-norms');`