



Butify

Plateforme de musique

Richard, Ebrotié, Agathe, Théo

SAE 5.01 - 3^{ème} année BUT Informatique



Table des matières

Introduction	3
Présentation du projet	3
Présentation des membres du groupe	4
Conception & Déroulement du projet.....	5
Outils utilisés en interne	5
La stratégie incrémentale	7
Les choix architecturaux et technologiques	8
Choix architecturaux :	8
Choix Technologiques :	12
Problèmes attendus et rencontrés	13
Problèmes liés à la performance du projet	13
Évolutions du projet	14
Méthodes et outils de qualités mis en place.....	14
Mesures de performance.....	15
Détail et justification de la stratégie de tests déployée	15
Dockerisation et mise en production	16
Fonctionnalités de la plateforme	17
Listing et explications.....	17
Différence sur l'application mobile.....	23
Conclusion	24
Conclusions personnelles	24
Prolongements imaginés pour le projet.....	26





Introduction

Présentation du projet

Pour ce projet nous nous sommes basés sur un constat simple. La majeure partie des applications de musiques sont soit remplies de pub, ou proposent un abonnement pour les enlever, avec un abonnement premium atteignant un prix jusqu'à 3 fois celui de l'abonnement de base.

Nous avons donc décidé de créer la nôtre avec pour l'instant une solution sans pubs et gratuite avec les fonctionnalités utiles et essentiels à la bonne utilisation d'une plateforme de musique. Tout en mettant en avant nos artistes.

Un des objectifs de notre application étant de ne pas perdre les utilisateurs ou devoir les réhabituer à un autre type de plateforme de musique, nous avons donc décidé de garder certaines similarités visuelles d'autres plateformes. Et avec comme types de données d'écoutes nos artistes émergeant et les artistes plus connus des autres plateformes grâce à l'ajout d'une source externe.

Imaginez notre application comme une ruche bondée de créativité et de diversité sonore

- Les fleurs (les morceaux de musiques) sont partout prêtes à être découvertes. Elles sont une source unique qu'il s'agisse de grands classiques ou de nouveautés émergentes.
- Les champs de fleurs (les playlists) peuvent se partager à plusieurs abeilles de par leur grandeur et leur richesse en nectar, afin de former un harmonieux bouquet sonore.
- Les abeilles (les utilisateurs) sont au cœur de notre écosystème elles ont pour rôle de contribuer à la richesse collective de la ruche en apportant du nectar aux reines
- Les reines (les artistes) ont un rôle clé dans l'évolution de la ruche et de la population. L'augmentation de cette population accroît le rayon de diversité permettant la découverte de plus de fleurs et de champs de fleurs.
- Les apiculteurs (les administrateurs) permettent de maintenir un équilibre stable et certain de l'écosystème.





Présentation des membres du groupe

Richard : Alternant au Conseil Départemental, affinités : PHP

Ebrotié : Alternant chez SofTech58, affinités avec HTML, BOOTSTRAP, JQUERY

Agathe : Alternante au technicentre SNCF, affinités avec du No Code Web.

Théo : Aucunes affinités particulières, envie de découverte des langages Mobile.



Butify

Plateforme de musique



Conception & Déroulement du projet

Outils utilisés en interne

Réalisation des maquettes => Figma



Figma est le logiciel le plus connu et nous a été recommandé en cours. De par sa facilité de collaboration et sa prise en main permettant d'obtenir une visualisation rapide.

Conception base de données => MySQL Workbench



MySQL Workbench est le logiciel le plus adapté à notre base de données (MySQL). Ayant déjà été étudié en cours et manipulés lors de précédentes SAE.

Outils de communications => Discord



Chacun de nous est familiarisé avec Discord. Il a joué un rôle clé dans ce projet en palliant l'incapacité de se voir directement pendant les périodes où les alternants étaient en entreprise. Discord a permis non seulement de regrouper les membres du groupe mais également de regrouper les informations clés du projet. Malgré la distance, nous avons pu prendre des décisions grâce à des sondages et des réunions récurrentes.

Edition du code => Visual studio code,



Visual studio code est utilisé récurrentement par tous les membres du groupe. Certains de ses points forts sont : une interface agréable une facilité d'ajouts d'extensions diverses.



Butify

Plateforme de musique



Edition du code => Android Studio,



Android Studio est un élément primordial pour le développement mobile par son intégration d'émulateur et sa place défilant toute concurrence sur le marché. Qui plus est bien maîtrisé par Théo.

Transfert de fichiers => WinSCP



WinSCP permet la manipulation des fichiers sur le serveur à distance, tout en gardant une interface d'utilisation très simple et intuitive.

Gestion des tâches => Notion



Pour la répartition des tâches nous avons décidé d'utiliser notion afin de pouvoir personnaliser nos tâches et avoir une vision plus claire de notre avancée et de l'urgence ou non des tâches restantes à accomplir.

Trello Butify

Vue d'ensemble des tâches

Tâches non commencées

Tâches en cours

Tâches faites

Tâches Richard

Tâches Elvire

Tâches Agathe

Tâches Théo

<



Butify
Plateforme de musique



Hébergement du projet => Serveur dédiée (Richard)

Pour permettre d'héberger la SAE et voir l'évolution du projet en continu avec le travail de chacun. Et réduire les risques de conflits dû la transmission du projet d'un membre à un autre. Cela facilite également la correction de bug sans avoir à remettre à jour son projet. Les membres étant connectés sur le serveur peuvent éditer directement les fichiers et donc avoir le retour instantané sur le site.

Serveur web => WampServer



WampServer est très modulaires au niveau de ses changements de versions MySQL, PHP ainsi que sa configuration de serveur web.

La stratégie incrémentale

Nous avons défini la fonctionnalité obligatoire de notre projet qui est l'écoute de musique à laquelle nous avons rajouté les fonctionnalités essentielles comme la recherche, les playlists. Puis se sont greffées les fonctionnalités complémentaires, les collaborateurs, la lecture en boucle, la lecture aléatoire...



Butify

Plateforme de musique

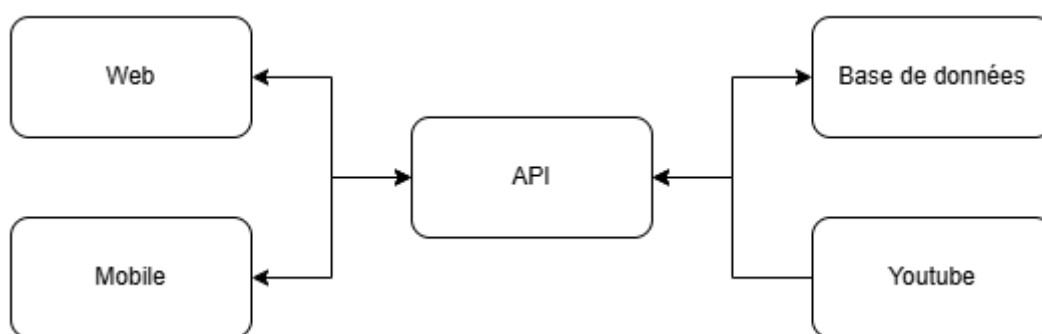


Les choix architecturaux et technologiques

Choix architecturaux :

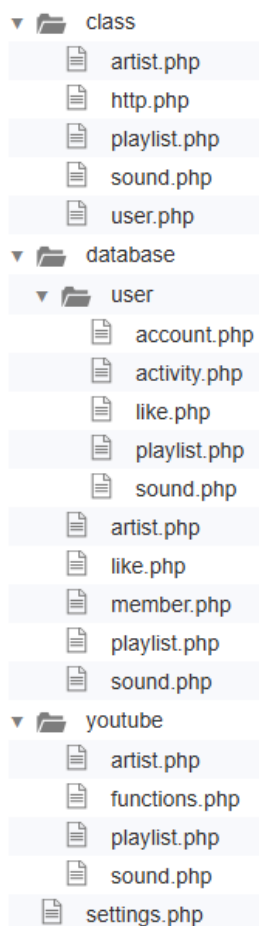
Api :

Nous avons réalisé une API pour permettre la connexion du web et du mobile aux mêmes données provenant de la base de données et de l'API YouTube sous le même format grâce à l'implémentation de classes PHP pour rendre dynamique le projet.





Structure de l'API



Le dossier class contient les classes qui permette la sérialisation des données entre notre base de données et youtube

Le dossier database contient toutes les fonctions pour la récupération des données dans notre base de données et renvoie le tout avec le format de nos classes en tableau associatif.

Le dossier YouTube contient toutes les fonctions, appelle sur l'API YouTube et renvoie le tout avec le format de nos classes en tableau associatif.

Le dossier user dans database contient toutes les fonctions pour la gestion des utilisateurs.

Le nom de chaque fichier indique quelles données sont manipulées dedans, par exemple les fichiers "artist.php" s'occuperont des données des artistes de leur partie

Le fichier "settings.php" permet la configuration de l'api exemple le type des extensions accepté, la taille des fichiers acceptée et même encore l'adresse du serveur pour le stockage des fichiers.

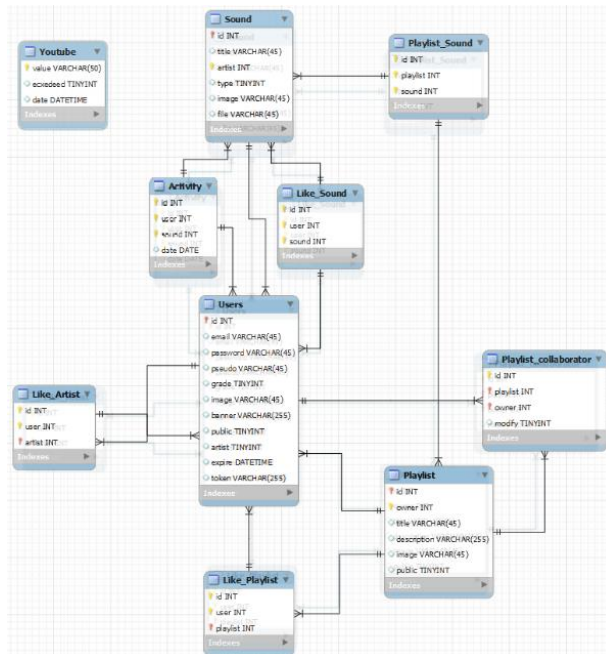
Web :

Pour garder la dynamique de l'interface un routeur a été mis en place pour le changement de page. Ce qui permet au lecteur de passer en version minimisée dans les pages afin de garder la navigation sur le site sans couper l'écoute actuelle.





BDD :



Pour permettre à nos utilisateurs de venir sur notre plateforme, ils devront renseigner leurs informations de base et s'ils sont artistes. Ils pourront également changer leur photo de profil, leur bannière et la visibilité de leur compte.

Un utilisateur peut créer une ou plusieurs playlists en leur donnant un titre, une description, une image et sa visibilité.

Un artiste peut créer une ou plusieurs musiques et vidéos qui ont un titre, une image et un fichier mp3 ou mp4.

Un utilisateur peut aimer un ou plusieurs artistes, playlists, musiques

Une playlist peut contenir aucune ou plusieurs musiques.

Une playlist peut avoir aucun ou plusieurs collaborateurs, avec ou non la permission de modifier.

Contraintes :

Tout utilisateur doit avoir une adresse mail unique, et les artistes doivent avoir en supplément un pseudonyme unique parmi ceux de notre plateforme.

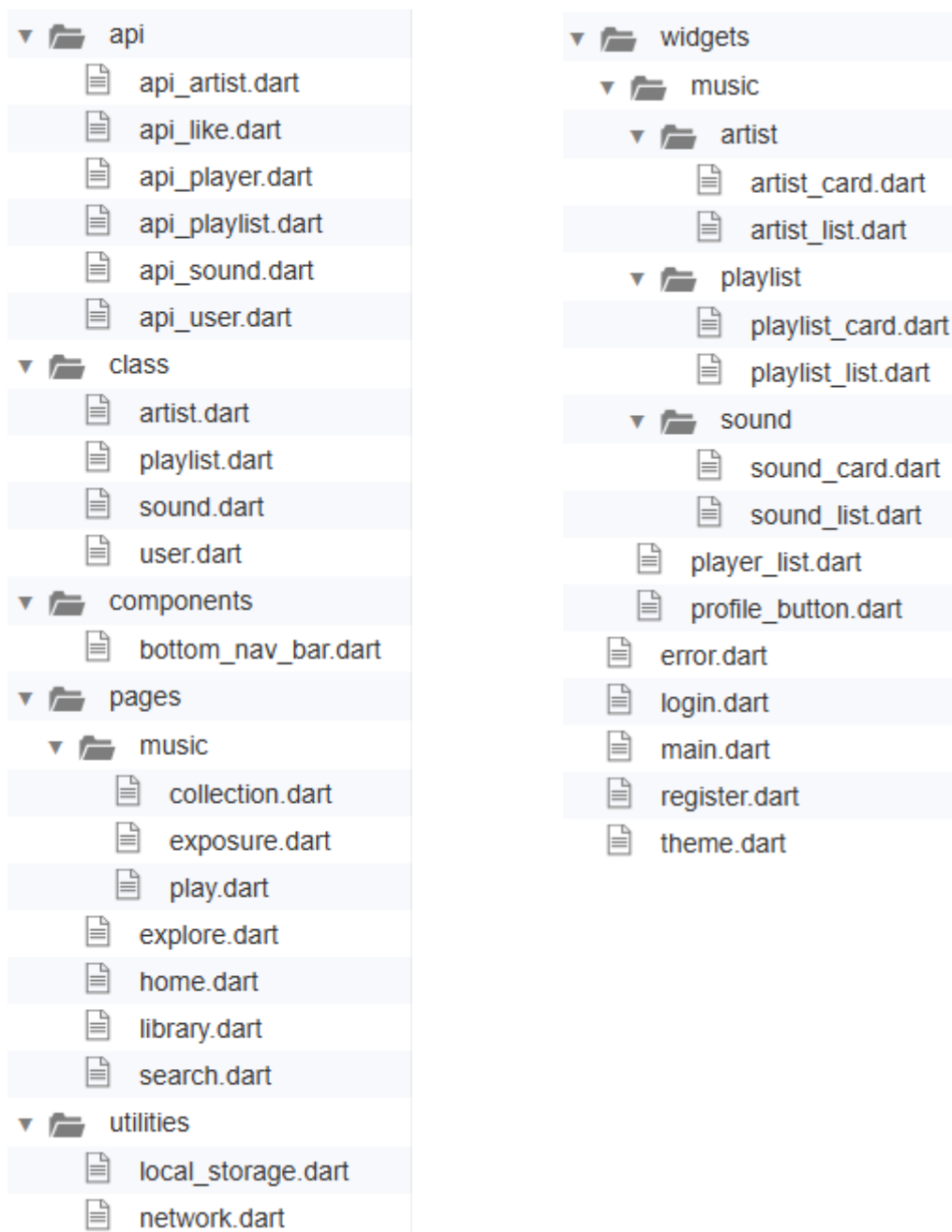
L'utilisateur ne peut pas liker son propre profil, ni sa propre playlist et/ou ses playlists collaboratives





Web et Mobile :

Nous avons mis en place une structure de code commune pour avoir une ressemblance au niveau de l'expérience utilisateur. En outre cela nous permet de faciliter la compréhension du code et par conséquent de permettre une meilleure résolution de bug d'une plateforme à l'autre.





Nous retrouvons dans cette structure de code commune :

- 6 pages qui représentent l’affichage. (Accueil, Exploration, Bibliothèque, playlist, Artiste, Lecteur de musique).
- 4 classes qui permettent de formater les données de l’API (Sound, Playlist, Artist, User)
- 3 widgets qui seront utilisés pour formaliser les données renvoyées par notre API (Carte de sons, Carte de playlist, Carte d’artiste.)
- 1 composant navigation bar qui permet la navigation dans les différentes pages

Choix Technologiques :

Api :

Richard étant amené à utiliser le PHP dans son alternance il s’est donc occupé de la partie API <=> base de données. Agathe n’ayant peu d’expérience en PHP et en développement d’API, elle a pu développer ses compétences en se chargeant de la partie API <=> YouTube tout en étant sous la supervision de Richard.

Web :

Ebrotié développe au quotidien des projets utilisant HTML, Bootstrap, JQuery. Nous avons donc utilisé ces technologies à l’exception du JQuery remplacé par du JavaScript. Richard a pu intervenir en support d’Ebrotié tout en développant le lecteur audio/vidéo.

Mobile :

Théo ayant découvert le Flutter pendant les cours de 2ème année il a voulu pousser cette compétence afin de découvrir une potentielle spécialisation en développement mobile. Tout en changeant du développement web approfondi tout au long du cursus. Le Framework Flutter permettant de développer un code hybride entre Android et iOS qui permet un déploiement sur la majorité des appareils actuels. Richard a également pu conseiller Théo grâce à son expérience personnelle. Et s’est occupé du lecteur audio/vidéo.





Problèmes attendus et rencontrés

Problème Attendu : Clés d'accès à l'api YouTube plus limité qu'espérer.

Solution trouvée : Roulement de plusieurs clés dans la base de données.

Problèmes Attendus : Implémentation et traitements des données extérieures à notre application.

Solution : Utilisation de l'api YouTube et sa documentation avec son lecteur audio/vidéo intégré.

Problèmes liés à la performance du projet

Problème rencontré : Temps de réponse de l'API YouTube assez lent

Solution trouvée : Pour éviter l'effet longue attente avant l'apparition de la carte un chargement de son squelette apparait. En plus de cela au lieu d'exécuter chaque requête et d'additionner leur temps de réponse pour obtenir un résultat final nous parallélisons les différentes requêtes, où la plus longue clôturera son résultat final.

Problème rencontré : Superposition de page

Détail du problème : Lorsque le changement de page est trop rapide, c'est-à-dire que le contenu de la première page n'a pas encore fini de se charger, il arrive donc dans la suivante. Cela est dû à notre routeur évitant le blocage de la navigation de l'utilisateur grâce aux fonctions asynchrones.

Solution trouvée : Il était totalement possible d'attendre les requêtes de la page avant d'en changer, mais cela aurait été contre notre but de dynamiser la navigation de l'utilisateur. La solution trouvée a été de supprimer le conteneur ayant les requêtes asynchrones de la première page et de le régénérer dans la suivante afin de casser la liaison entre ces dernières.





Évolutions du projet

Ajout du clonage de playlist pour améliorer l'ergonomie visuelle de la page de playlist dans la configuration où il n'y avait que 2/3 boutons.

Ajout d'une actualisation de token du côté mobile et web toutes les 30 minutes pour améliorer la sécurité.

Méthodes et outils de qualités mis en place

Pour une meilleure compréhension du code nous avons mis en place une documentation textuelle dans le code que l'on a convertis en documentation structurée grâce à Doxygen. Cette documentation permet une meilleure visualisation du fonctionnement du projet.

La structure de l'API YouTube nous forçant à faire 2 requêtes pour récupérer des informations a rapidement engendré un problème de performance qui aurait pu être évité par la simplification en une seule requête.

Par exemple la récupération d'une musique nous renvoie l'id de l'artiste ce qui nous force à refaire une deuxième requête pour obtenir les informations de l'artiste.

Pour cela nous avons créé un widget compenser par l'optimisation de l'envoi des requêtes et de l'affichage.





Mesures de performance

Nous avons utilisé l'onglet network dans le DevTools du navigateur pour cibler le temps de réponse des requêtes. Et nous utilisons l'extension Page load time pour obtenir le temps de chargement global de la page.

Détail et justification de la stratégie de tests déployée

Pour ce projet, il nous a semblé pertinent de réaliser des tests sur la partie API et la partie web du projet.

Pour l'API, nous voulons vérifier les informations transmises par la base de données et l'API Youtube. Dans un premier temps, nous testons la récupération des informations d'un artiste, d'une playlist, d'une musique et enfin d'un utilisateur.

Sur l'application globale, nous testerons les principales fonctionnalités. Tout d'abord, celle la création, modification et déconnexion d'un utilisateur. Ensuite, la création, modification et suppression de playlists. Ainsi que la création, like et suppression de musique. Enfin, nous testons les fonctionnalités qui ne sont pas disponible pour l'utilisateur comme le l'ajout de sa propre playlist en favorite.





Dockerisation et mise en production

Les applications de musique génèrent un flux de données massif, il était donc important de séparer chaque service de notre projet en plusieurs serveur web qui communiqueront entre eux.

Architecture :

<input type="checkbox"/>	Name	Container ID	Image	Port(s)
<input type="checkbox"/>	butify	-	-	-
<input type="checkbox"/>	phpmyadmin	cc66399a2e3f	phpmyadmin/phpmyadmin	8083:80 ↗
<input type="checkbox"/>	database	3fc84f7af0a3	mysql:8.0	3306:3306 ↗
<input type="checkbox"/>	web	9c13e01cedfb	php:8.0-apache	8080:80 ↗
<input type="checkbox"/>	docs	17b14da98263	nginx:alpine	8084:80 ↗
<input type="checkbox"/>	storage	3938c4780e32	php:8.0-apache	8081:80 ↗
<input type="checkbox"/>	api	de690c31b8f2	php:8.0-apache	8082:80 ↗

Si un ou plusieurs services sont surchargés nous pouvons créer un réplica de celui-ci qui recevra une partie du flux dispatché afin de pouvoir gérer encore plus de flux.





Fonctionnalités de la plateforme

Listing et explications

Nécessaires :

- Lecteur de musique ;
- File d'attente
- Recherche de musique, playlist, artiste ;
- Création de compte artiste ;
- Création de compte utilisateur ;
- Connexion et déconnexion du compte ;
- Modification des informations de compte ;
- Création de playlist ;
- Publication de musique et de clip ;
- Liker une musique, une playlist, un artiste ;
- Accès à sa bibliothèque de musique ;
- Explorer des musiques.

Optionnelles :

- Ajout de collaborateurs à une playlist ;
- Duplication de playlist ;
- Changement de thème.
- Gestion des utilisateurs (panel administrateur) ;





Lecteur de musique

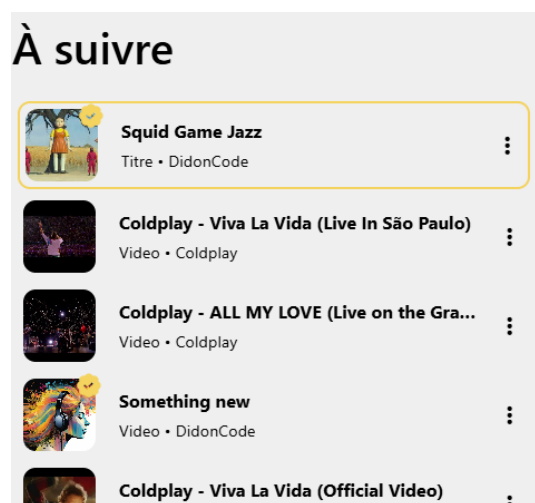
Lorsqu'une musique est lancée, un lecteur audio/vidéo est affiché, où l'utilisateur peut :

- Mettre en pause la musique ;
- La liker ;
- Revenir à la musique précédente ;
- Aller à la musique suivante ;
- Augmenter ou baisser le volume ;
- Répéter la musique en boucle ;
- Écouter des musiques aléatoirement ;
- Minimiser ou maximiser le clip ou l'image de la musique.



File d'attente

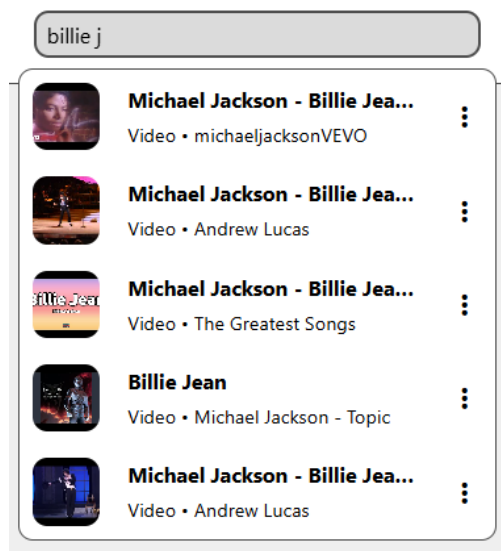
La liste d'attente d'écoute est importante pour l'utilisateur car elle lui permet de ne pas avoir besoin de changer manuellement de musique à la fin de celle-ci. Cette file d'attente va être accompagnée d'un algorithme pour proposer les meilleures musiques. L'algorithme va recevoir la musique lancée par l'utilisateur et va choisir plusieurs musiques du même artiste, des écoutes récentes d'autre musique, des musiques d'artiste récemment écoutées. Puis nous allons les mélanger uniformément et en créer une "playlist" en rapport avec la musique et l'écoute de l'utilisateur pour lui permettre une écoute fluide et diversifiée.





Recherche de musique, de playlist et d'artiste

Dans toutes les pages de la plateforme, l'utilisateur a la possibilité de rechercher une musique, une playlist ou un artiste. Pendant qu'il écrit sa recherche, il aura des suggestions correspondant à ce qu'il a déjà écrit. Lorsqu'il appuie sur "Entrée" il y aura l'affichage des musiques, playlists et des artistes qui correspondent à sa recherche.



Publication de musique ou clip

Les artistes pourront publier des musiques ou des clips en choisissant un nom, une image de couverture et un fichier audio ou vidéo. Par la suite, l'artiste pourra gérer ses musiques et clips depuis la page dédiée.

Les likes

L'utilisateur a la possibilité de liker une musique, une playlist ou un artiste. Il pourra ainsi avoir accès sur sa page d'accueil à tous les éléments likés en plus de son historique d'écoute.





Création, connexion et modification du compte

L'utilisateur pourra créer un compte en fournissant les informations suivantes dans le formulaire d'inscription : pseudonyme, e-mail, mot de passe. L'utilisateur pourra également choisir de créer un compte artiste en cochant la case "Créer un compte artiste" dans le formulaire. Une fois son compte créé, l'utilisateur sera automatiquement connecté sur la plateforme, il pourra se déconnecter à tout moment. Lorsqu'il le souhaite l'utilisateur pourra modifier les informations de son compte (photo de profil, visibilité privée ou publique, pseudonyme et mot de passe. L'e-mail est modifiable par l'utilisateur seulement si ce n'est pas un artiste.

Créer un compte


☐ Créer un compte artiste

INSCRIPTION


La bibliothèque de musique

La bibliothèque de musique recense toutes les playlists de l'utilisateur en plus des éléments likés ; que ce soient les musiques, les playlists et les artistes.


VOS PLAYLISTS



À FOND !
net 400/100



À FOND !
net 400/100



SONS DE LA NATURE
Musiques calmes et apaisan...





Création de playlist

À la création d'une playlist, il sera demandé à l'utilisateur quelle visibilité il souhaite attribuer à cette dernière. Il a le choix entre la visibilité publique et la visibilité privée. S'il choisit la visibilité publique, les autres utilisateurs pourront voir sa playlist depuis sa page de profil. Ils pourront également écouter cette dernière. Après cette étape de création, l'utilisateur pourra ajouter des musiques à sa playlist. Il pourra lorsqu'il le souhaite, modifier : le nom, la description et l'image de couverture de sa playlist. Il pourra également supprimer sa playlist s'il en a envie.

Créer une playlist

Titre *

Description

Privée

Créer la playlist

Exploration de musiques

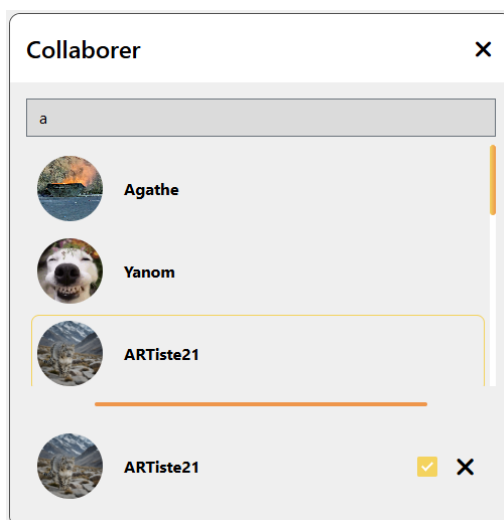
L'exploration de musique consiste ici à proposer à l'utilisateur les musiques et les playlists : les plus écoutées, les musiques et les playlists les plus aimées ; mais également les musiques et artistes émergents.





Ajout de collaborateurs à une playlist

L'utilisateur a la possibilité d'ajouter des collaborateurs à ses playlists. Ainsi ces derniers pourront tout comme le propriétaire de la playlist : ajouter ou supprimer des musiques. Cependant ils ne pourront ni changer les informations de la playlist : image de couverture, nom et description, ni ajouter à leur tour des collaborateurs à celle-ci, ni la supprimer. S'ils souhaitent ne plus être collaborateur de la playlist, ils pourront la quitter via le bouton dédié.



Changement de thème

L'utilisateur peut passer du thème clair à sombre et inversement. Le thème sombre offre une expérience plus douce pour les yeux.

Panel administrateur

Sur le panel admin, plusieurs fonctionnalités s'offrent à nous. Nous pouvons, supprimer des musiques, supprimer des playlists, supprimer des utilisateurs ainsi que changer le grade des utilisateurs afin de leur donner le rôle "Administrateur".





Différence sur l'application mobile

Etant en découverte et ayant pris le risque d'utiliser Flutter qui est encore en cours de développement je n'ai pas pu implémenter certaines fonctionnalités pour diverses raisons :

Non implémentées par manque de temps dû à la découverte de Flutter :

- Modification du compte
- Ajouts de sons à une playlist
- Lecture aléatoire (dans la pageplaylist)
- Modification de playlist
- Système de collaborateurs
- Affichage du squelette des cards pendant le chargement
- Playlist aimée crée par défaut avec toutes les musiques aimées.

Non implémentées par problèmes de package et bibliothèques flutter :

- Utilisation d'image lors de création de playlist

Non implémenté par autres raisons :

- Gestion des musiques de l'artiste : Nous avons pensée que le rendu serait beaucoup mieux sur web et moins pertinent sur mobile.
- Actualisation du token toutes les 30 mins : J'ai réalisé cette fonctionnalité mais j'ai eu un problème de déconnexion toutes les 30 minutes ce qui aurait gênée l'expérience utilisateur, je l'ai donc mise en debug et j'ai régler d'autres fonctionnalités plus importante en voyant que j'avais un temps limité et des fonctionnalités plus prioritaires.





Conclusion

Conclusions personnelles

Agathe :

J'ai réellement apprécié travailler sur ce projet. Je pense m'être énormément améliorée sur PHP, j'ai aimé apprendre et comprendre le fonctionnement de l'application.

Ces nouvelles connaissances ont pu m'aider lors de la création d'outils en alternance.

Néanmoins, je pense avoir mal géré mon temps de travail lors des périodes d'alternances.

Notamment lors de la création de l'API YouTube, où sans une maintenance et correction des bugs, les autres ne pouvaient pas travailler.

Malgré tout cela, ce projet m'aura permis d'évoluer professionnellement et je remercierais mes camarades pour leur implication et leur aide apportée lorsque j'étais bloquée.

Ebrotié :

Le projet m'a permis de consolider mes acquis au niveau des langages front-end que ce soit au niveau HTML et CSS avec et sans le framework Bootstrap.

J'ai pu également revoir l'utilisation d'API côté JavaScript en réalisant des appels à notre API Butify.

En réalisant les maquettes de plusieurs pages j'ai pu grandement m'améliorer sur l'utilisation de Figma.

J'ai également amélioré mon organisation dans la gestion de mon temps entre l'alternance, la SAE et les cours.

Au niveau du futur professionnel je me pose la question de soit me spécialiser côté front-end ou de devenir développeur full-stack.



**Théo :**

Pour ma part ce projet m'a permis de confirmer ma future voie professionnelle, je me suis vraiment plu dans le développement de cette application mobile.

Durant ce projet j'ai été confronté à plusieurs adversités ce qui m'a permis en dehors du côté technique de progresser dans certains points comme la gestion de mon temps qui m'a valu un départ plus ralenti en début de projet qui a fini par se résoudre au fur et à mesure de l'avancée du projet.

Ce projet m'a également permis de confirmer la résolution de certains problèmes durant les précédentes SAE comme le manque de conception et de réflexion.

Durant ce projet la conception a été un point clé pour ne pas être trop ralenti de mon côté.

Pour ce qui est du côté technique c'était ma première application mobile et j'ai appris énormément que ce soit au niveau architectural ou encore le Framework Flutter en lui-même.

Une des améliorations les plus importantes que m'a apporté ce projet a été l'augmentation de ma concentration pendant de longues heures.

Richard :

Étant chef d'équipe et lead technique sur le projet, j'ai pu m'améliorer dans ces compétences en faisant des points individuels et réguliers avec chaque personne du groupe pour voir leur avancée, problème rencontré, et pouvoir les aider.

J'ai pu également créer une architecture logique et commune pour les 2 plateformes de notre projet.

Ainsi, qu'une API avec une architecture logique et qui permet la combinaison de notre base de données et celle de YouTube.

Grâce à celle-ci et le lecteur de musique, mon niveau en JavaScript a pu vraiment évoluer en codant des classes, widgets, call API, callback. Sur tout le projet permettant un code fluide et de bonne qualité.





Prolongements imaginés pour le projet

L'amélioration de la gestion des collaborateurs comme les invitations serait un point clé pour une future maintenance technique du projet.

L'ajout de la création et gestion d'albums par les artistes serait intéressant.

Pour améliorer la découverte de nos artistes auprès de nos utilisateurs, une bonne idée serait de lancer l'écoute d'une musique aléatoire de l'artiste. L'utilisateur pourra soit aimer ou rejeter cette musique qui sera ajoutée ou non dans une playlist « historique » qui aura les propriétés d'une playlist avec la particularité d'être utilisée pour affiner son algorithme de recommandations.

Trouver une source de rentabilité afin de permettre l'utilisation d'une API moins limitée et avec de meilleurs temps de réponses. Comme l'ajout d'un abonnement payant permettant l'accès à de nouvelles fonctionnalités, ou encore des liens affiliés rémunérés.

Pour améliorer l'algorithme de recommandation de musique aux utilisateurs, il faudrait pouvoir rajouter une détection automatique du genre des musiques de notre plateforme

