

PROJET DÉVELOPPEMENT MOBILE

[Développement Mobile]

GADEAU Juliette - NDIAYE Mariam - PERRIER ALBAN

BORDEAUX INP

ENSC 2A – Groupe 1 & 2

SOMMAIRE

INTRODUCTION	2
NOTICE D'INSTALLATION	3
Installation de Node.js	3
Installation d'Expo	3
Installation de l'application	3
INTERFACE HOMME-SYSTÈME	6
Fonctionnalités	6
Tops	6
Profil musical	7
Recommandations	9
Ergonomie	10
IMPLÉMENTATION DE L'API	11
Architecture	11
Appel à l'API	13
Structure des données reçues	14
GESTION DE PROJET	15
Répartition des tâches	15
Organisation	16
BILAN	17
Difficultés rencontrées	17
Améliorations	17
Conclusion	17

INTRODUCTION

Ce rapport tient compte du travail fourni dans le cadre du projet de développement mobile 2021. L'objectif de ce projet est de créer une application mobile présentant des données issues d'une API web externe.

Notre choix d'API s'est porté sur l'**API de Spotify**, plateforme de streaming musical. L'application que nous avons développée permet à un utilisateur Spotify de déterminer son "**profil musical**". Il peut consulter ses tops artistes et titres, obtenir des recommandations personnalisées et consulter son profil musical, déterminé à partir des caractéristiques audio, fournies par Spotify, de ses titres les plus écoutés. Il peut également créer une playlist "profil musical" depuis l'application et y ajouter des titres.

L'API nécessite la possession d'un compte ainsi qu'une authentification.

NOTICE D'INSTALLATION

Installation de Node.js

1. Rendez-vous à l'adresse suivante : [Download](#) et téléchargez la version correspondante à votre installation.
2. Installez Node.js

Installation d'Expo

1. Ouvrez un terminal WIndows puis tapez la commande ci-dessous pour installer Expo.
`npm install -g expo-cli`
2. Sur votre smartphone, installez ExpoGo à l'aide du lien correspondant :
 - iOS : [Apple Store](#)
 - Android : [Play Store](#)

Installation de l'application

1. Pour installer le projet, commencez par cloner le projet au lien Github suivant :
https://github.com/ensc-mobi/projet-2021-gadeau_ndiaye_perrier.git
2. Ensuite, ouvrez à l'intérieur du dossier ProfilMusical un invite de commandes.
Tapez alors `npm install`, puis lancez l'application avec `npm start`.
3. Un QR code apparaît alors. Ouvrez l'application “Expo Go” et scannez ce QR code : l'application se lance automatiquement.
4. Vous arrivez sur la page de connexion. Cliquez sur le bouton “Se connecter” (*attention, ne cochez pas “se souvenir” si vous souhaitez essayer avec différents comptes Spotify, la déconnexion n'est pas fonctionnelle*), puis connectez-vous soit à l'aide des identifiants de votre compte personnel, soit avec les identifiants suivants :
Email : ndl.mariam@icloud.com
Mot de passe : Profilmusical2022
5. Vous êtes ensuite redirigés sur notre application ProfilMusical.

Que faire si la connexion à Spotify ne fonctionne pas ?

C'est-à-dire soit l'application ne vous propose pas de vous connecter à Spotify, soit le lien de redirection ne fonctionne pas (invalid client, invalid redirect uri...).

Il faut copier l'adresse qui se trouve au dessus du QR code dans l'application Expo Go comme suivant :

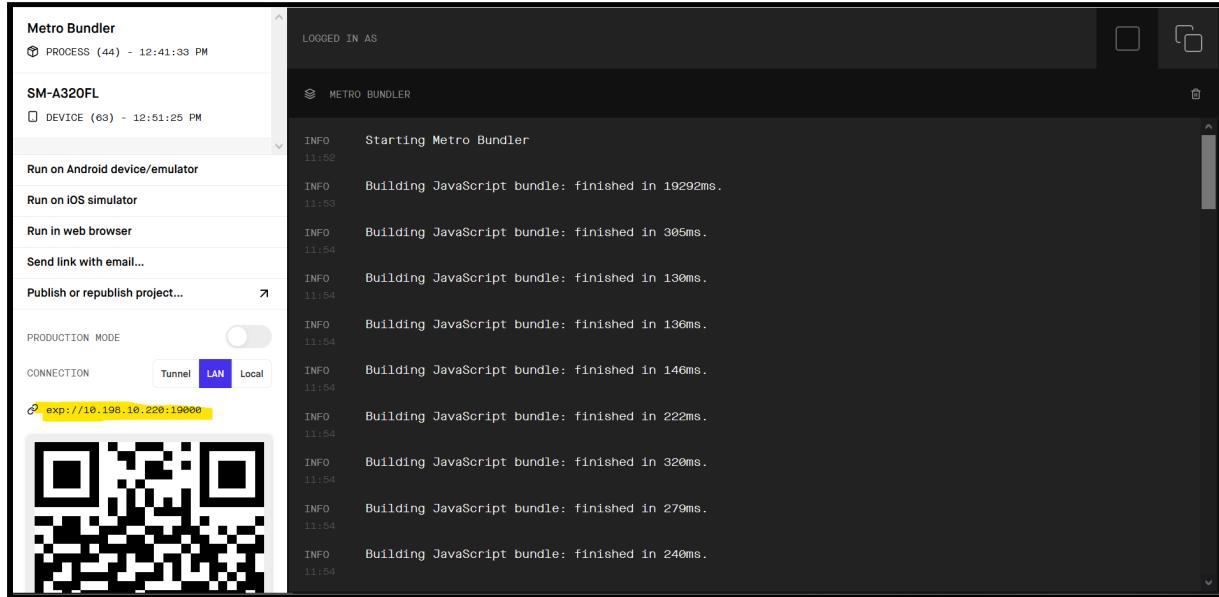


Figure 1 - Capture d'écran de l'URI

NB : Certaines connexions Wi-Fi (celle de l'école notamment) nécessitent le passage du mode de connexion LAN au mode Tunnel

Dans le code, dans le fichier spotify.token.ts, il faut coller cette adresse à côté de redirectUri (ligne 33) en ajoutant à la suite "/--/".

```

13 import {
14   makeRedirectUri,
15   RefreshTokenRequest,
16   ResponseType,
17   useAuthRequest,
18 } from "expo-auth-session"; //pour faire implicit flow
19 WebBrowser.maybeCompleteAuthSession();
20 const discovery = {
21   authorizationEndpoint: "https://accounts.spotify.com/authorize",
22   tokenEndpoint: "https://accounts.spotify.com/api/token",
23 };
24 let redirectUri = "exp://localhost:19000/--/";
25 let accessToken = null;
26 export default function getToken() {
27   debugger;
28
29   if (Platform.OS == "web") {
30     redirectUri = "http://localhost:19006/";
31   }
32   if (Platform.OS == "android") {
33     redirectUri = "exp://10.198.10.220:19000/--/"; ←
34   }
35   const [request, response, promptAsync] = useAuthRequest(
36     {
37       responseType: ResponseType.Token, //pour faire implicit flow
38       clientId: "5ab19fc39ae4615950c77502815e69c",
39       scopes: [
40         "user-read-email",
41         "user-top-read",
  
```

Figure 2 - Capture d'écran de l'URI à remplacer

Ensuite, il faut copier l'adresse complète avec "/--/" pour l'ajouter à Spotify.

Pour cela, connectez vous à l'adresse <https://developer.spotify.com/dashboard/login> avec le compte Spotify suivant :

Email : ndl.mariam@icloud.com

Mot de passe : Profilmusical2022

Id : profilmusicalgnp

Cliquez sur ProfilMusical, puis “Edit Settings”. Ensuite, collez l'adresse de *redirectUri* dans la partie Redirect URIs.

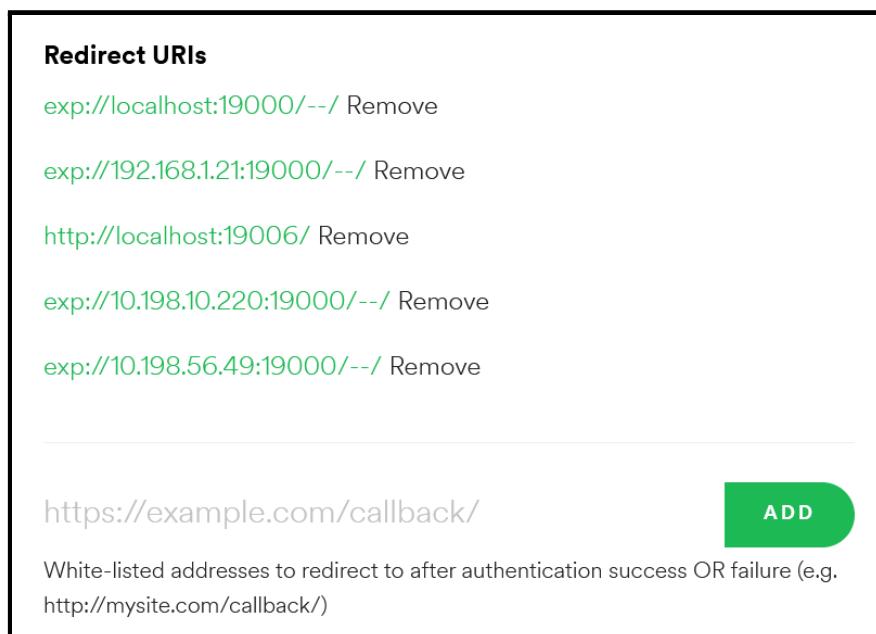


Figure 3 - Capture d'écran des settings des redirect URIs du projet

Faites “ADD” puis “Save” en bas de page. Vous pouvez maintenant vous connecter.

INTERFACE HOMME-SYSTÈME

Fonctionnalités

Les fonctionnalités développées sont regroupées dans trois catégories : **tops**, **profil musical et recommandations**.

Tops

À l'ouverture de l'application, l'utilisateur peut consulter son top titre et son top artiste, c'est-à-dire les titres et les artistes qu'il a le plus écouté. Il peut choisir d'afficher le nombre de titres ou d'artistes qu'il souhaite. L'utilisateur peut écouter une preview de 30 secondes des chansons lorsqu'elles sont disponibles.

Vous trouverez ci-dessous l'écran d'accueil où sont affichés les tops.

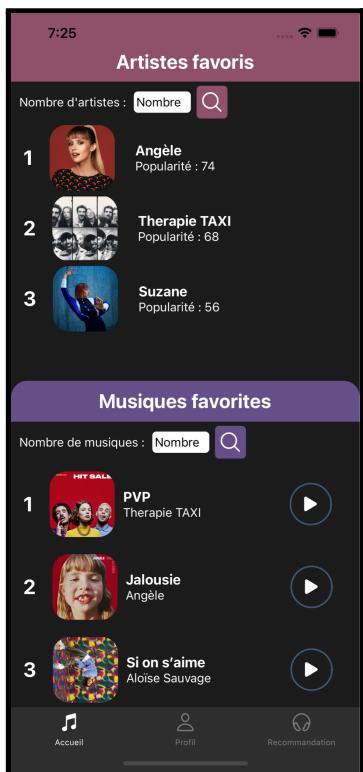


Figure 4 - Écran d'accueil

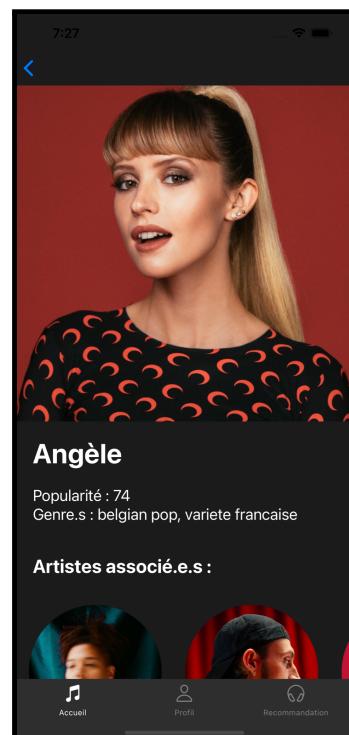


Figure 5 - Écran de détail d'un artiste

Comme on peut le voir sur les figures 5 et 6, l'utilisateur peut cliquer sur un artiste ou un titre afin d'afficher plus de détails.

De plus, il peut créer la playlist ProfilMusical sur son compte Spotify (figure 6), puis ajouter un titre à cette playlist (comme montré sur la figure 7).



Figure 6 - Ecran de détail d'un titre



Figure 7 - Playlist créée

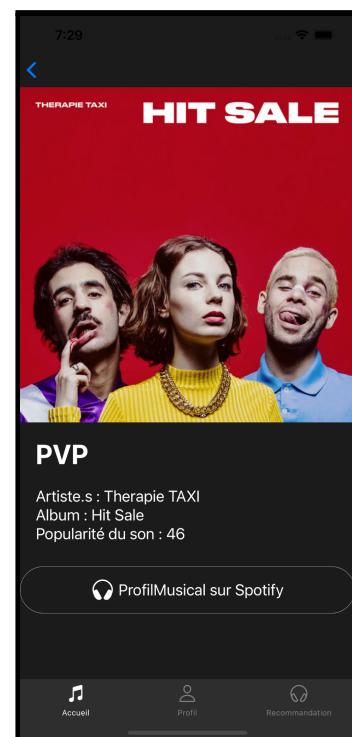


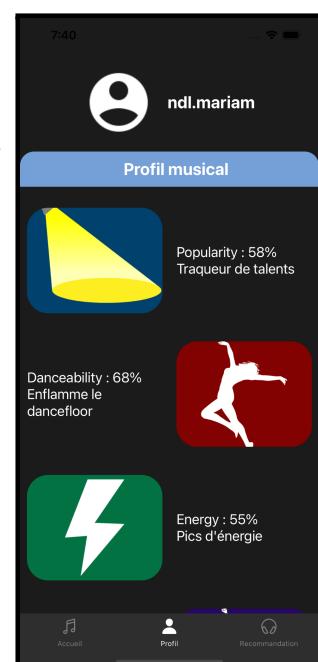
Figure 8 - Titre ajouté dans une playlist

Figure 9 - Écran de profil

Profil musical

Fonctionnalité phare de l'application, le profil musical de l'utilisateur est déterminé à partir de caractéristiques audio, ou *audio feature*, fournies par Spotify.

Ci-contre se trouve l'écran de profil musical.



Nous avons choisi d'utiliser les caractéristiques suivantes :

- **Danceability** : décrit dans quelle mesure un morceau est adapté à la danse sur la base d'une combinaison d'éléments musicaux, notamment le tempo, la stabilité du rythme, la force du battement et la régularité générale.
- **Valence** : décrit la positivité musicale véhiculée par un titre. Les morceaux à valence élevée sont plus positifs (par exemple, heureux, gais, euphoriques), tandis que les morceaux à valence faible sont plus négatifs (par exemple, tristes, déprimés, en colère).
- **Energy** : représente une mesure perceptive de l'intensité et de l'activité. En général, les morceaux énergétiques sont rapides, forts et bruyants. Par exemple, le death metal a une énergie élevée, tandis qu'un prélude de Bach obtient un score faible sur l'échelle.
- **Acousticness** : une mesure de confiance de 0,0 à 1,0 pour déterminer si le titre est acoustique, c'est-à-dire si des instruments électroniques modernes ont été employés.
- **Popularity** : un indice de 0 à 100 pour indiquer la popularité d'un artiste ou d'un titre (100 étant le plus populaire possible). Nous affichons la moyenne entre la moyenne des popularités des 10 titres les plus écoutés, et la moyenne des 5 artistes les plus écoutés.

Il est possible de consulter la description des ces caractéristiques depuis l'écran de profil musical par le bouton "Détails".

À droite se trouve cet écran. (*Figure 10 - Détails des caractéristiques*)

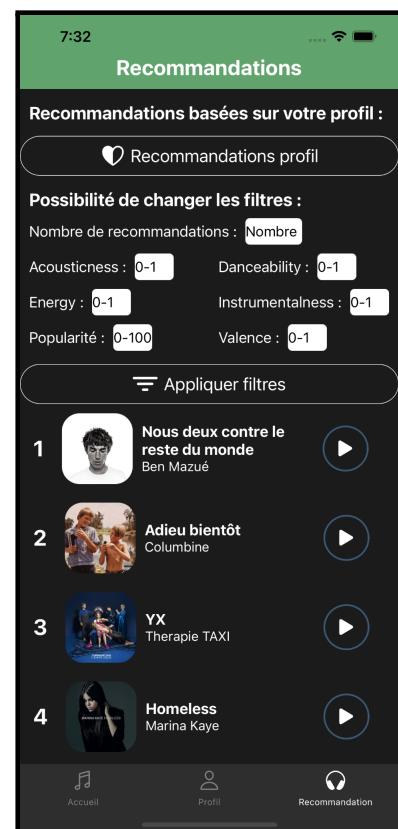


Pour chacune des caractéristiques, l'utilisateur fait partie d'une catégorie.

Audio feature	Catégories associées
Danceability	<ul style="list-style-type: none"> ○ entre 0 et 0.5 : Danseur.euse du dimanche ○ entre 0.5 et 0.8 : Enflamme le dancefloor ○ entre 0.8 et 1 : No limit
Valence	<ul style="list-style-type: none"> ○ entre 0 et 0.4 : Mélancolique ○ entre 0.4 et 0.8 : Extraverti.e ○ entre 0.8 et 1 : Ambianceur.euse
Energy	<ul style="list-style-type: none"> ○ entre 0 et 0.5 : Tranquille ○ entre 0.5 et 0.8 : Pics d'énergie ○ entre 0.8 et 1 : Inarrêtable
Acousticness	<ul style="list-style-type: none"> ○ entre 0 et 0.2 : Autotune lover ○ entre 0.2 et 0.7 : Plus électro que guitare ○ entre 0.7 et 1 : Ambiance feu de camp
Popularity	<ul style="list-style-type: none"> ○ entre 0 et 40 : Traqueur de talents ○ entre 40 et 80 : Étoile montante ○ entre 80 et 100 : Sous les projecteurs

Figure 11 - Écran de recommandations

Dans cet onglet sont affichées les recommandations de chansons à écouter selon le profil musical de l'utilisateur. Par défaut, les recommandations sont basées sur les valeurs de caractéristiques audio (ou *audio features*) de l'utilisateur, disponibles dans l'onglet "Profil". Il est possible de changer ces valeurs par défaut à l'aide des filtres (les filtres étant initialisés à 10 pour les recommandations et 0 pour le reste). Là encore, des preview de 30 secondes des recommandations sont disponibles.



Ergonomie

À propos de l'interface de l'application, nous avons tout de suite fait le choix d'une barre de navigation avec des icônes et un libellé situés en bas de l'écran pour respecter les guidelines courantes. Dans la même lignée, nos icônes, situées sur nos boutons, sont toutes accompagnées de labels afin de préciser leur usage pour l'utilisateur. Nos champs d'entrées utilisateurs (inputs) sont eux aussi accompagnés de labels, en tant que placeholder.

Concernant la navigation entre les écrans, l'onglet dans la barre de navigation est mis en valeur lorsque l'on se situe sur l'écran correspondant. De plus, si l'on va en profondeur dans un onglet, comme Profil par exemple avec l'écran de détails, un bouton de retour est disponible, afin que l'utilisateur puisse toujours revenir à l'écran précédent. Au sujet de cet écran, il permet à l'utilisateur de comprendre le fonctionnement de l'application et les caractéristiques sur lequel son profil est évalué.

Finalement, notre choix de couleur s'est basé sur la charte graphique de Spotify, en gardant une thématique sombre et des touches de couleurs.

IMPLÉMENTATION DE L'API

Architecture

L'architecture de notre application correspond à celle apprise en cours avec une séparation des responsabilités du code en *components*, *services* et *screens* ainsi qu'une partie pour la navigation dans l'application.

Concernant les *screens*, il s'agit de fichiers décrivant ce à quoi va ressembler l'écran et ce avec quoi l'utilisateur peut interagir.

Pour les services, on retrouve le fichier *spotify.token* qui permet de gérer l'authentification à l'API. Le fichier *spotifyapi.service* contient les requêtes qui récupèrent les données de l'API. Le fichier *feature.service* contient uniquement un tableau contenant les descriptions de chaque caractéristique audio. Enfin, nous avons des fichiers d'*objets.model* qui permet de définir un constructeur pour l'objet utilisé dans *spotifyapi.service*.

D'autre part, nous avons des *components*, qui permettent d'afficher des éléments graphiques que l'on pourra utiliser dans les screens.

Ce tableau récapitule les différents fichiers de notre application selon la fonctionnalité dans laquelle le fichier est utilisé.

Fonctionnalité	Screens	Services utilisés	Components utilisés
Connexion	ConnexionScreen Écran pour se connecter à son compte Spotify	<i>spotify.token</i>	
Affichage des tops (accueil)	HomeScreen Écran contenant les top artistes et top titres DetailsArtistScreen Ecran d'affichage des informations d'un artiste (nom, popularité, genre) DetailsTrackScreen Ecran d'affichage des informations d'un titre	<i>spotifyapi.service</i> <i>artist.model</i> <i>track.model</i>	TrackItem TrackList ImageTrack ArtistList ArtistItem ImageArtist ArtistListH ArtistItemH Inputs CircularProgress
Profil musical	ProfilScreen Ecran pour visualiser son profil musical, c'est-à-dire les caractéristiques audio DetailsAudioFeature Ecran d'explication de chaque caractéristique audio	<i>spotifyapi.service</i> <i>artist.model</i> <i>track.model</i> <i>audiofeatures.model</i> <i>feature.service</i>	ImageArtist ImageProfil Playlist FeatureList FeatureItem
Recommandations	RecommandationScreen Ecran des recommandations de titres selon le profil musical	<i>artist.model</i> <i>track.model</i> <i>audiofeatures.model</i> <i>spotifyapi.service</i>	TrackItem TrackList ImageTrack Inputs

Appel à l'API

L'API nécessite une authentification. Pour cela, nous utilisons la méthode Implicit Flow, dans le fichier spotify.token.ts. La méthode que nous utilisons correspond à un *authorization flow* nommé "Authorization code"

(<https://developer.spotify.com/documentation/general/guides/authorization-guide/>). Nous avons fait ce choix car nous souhaitions avoir un accès à l'API actualisable pour l'utilisateur. Ce flux convient à notre application car elle est ce qu'on appelle "de longue durée", dans laquelle l'utilisateur n'accorde une autorisation qu'une seule fois. Il fournit un jeton d'accès qui peut être actualisé.

Une fois que l'utilisateur a entré ses identifiants à l'aide de l'authentification Spotify, un *access_token* est généré. Un *token* est un jeton d'authentification qui est propre à la session d'un utilisateur. Il va nous permettre d'effectuer des requêtes par la suite et d'utiliser les méthodes de l'API pour récupérer des données.

```
class SpotifyApi {
  private ...fetchFromApi(query: string): Promise<Array<any>> {
    const userAccessToken = getAccess_token().access_token;
    return fetch(query, {
      method: "GET",
      headers: {
        Authorization: `Bearer ${userAccessToken}`,
      },
    })
      .then((response) => response.json())
      .then((jsonResponse) => jsonResponse.items || [])
      .catch((error) => {
        console.error(error);
      });
  }
}
```

Figure 12 - Appel à l'API

Ici, on va chercher l'accès à l'API à l'aide du *token* de l'utilisateur.

```
getArtistebyid(id: number) {
  const userAccessToken = getAccess_token().access_token;
  return fetch(`${rootEndpoint}/v1/artists/${id}`, {
    method: "GET",
    headers: {
      Authorization: `Bearer ${userAccessToken}`,
    },
  })
    .then((response) => response.json())
    .catch();
}
```

Figure 13 - Requête pour récupérer un artiste avec son identifiant

Et à l'aide de fonctions comme celle-ci, on peut utiliser les requêtes de l'API pour récupérer des données.

Structure des données reçues

La structure des données retournées par l'API est consistante sur toutes les requêtes, permettant un usage générique des JSON obtenus sans nécessité trop de traitement au préalable.

Par exemple, pour la requête “Get an Artist”, il suffira de fournir à l'API un id en lui indiquant que l'on cherche parmi les artistes :

```
GET https://api.spotify.com/v1/artists/{id}
```

Figure 14 - Méthode GET pour récupérer un artiste par son id défini par l'API

Cela nous retournera ce résultat qui est un tableau d'objets JSON :

```
{
  "external_urls": {
    "spotify": "https://open.spotify.com/artist/0OdUWJ0sBjDrqHygGUxeCF"
  },
  "followers": {
    "href": null,
    "total": 871490
  },
  "genres": [
    "indie folk",
    "indie pop",
    "indie rock",
    "modern rock",
    "stomp and holler"
  ],
  "href": "https://api.spotify.com/v1/artists/0OdUWJ0sBjDrqHygGUxeCF",
  "id": "0OdUWJ0sBjDrqHygGUxeCF",
  "images": [
    {
      "height": 640,
      "url": "https://i.scdn.co/image/0f9a5013134de288af7d49a962417f4200539b47",
      "width": 640
    },
    {
      "height": 320,
      "url": "https://i.scdn.co/image/8ae35be1043f330173de198c35a49161337e829c",
      "width": 320
    },
    {
      "height": 160,
      "url": "https://i.scdn.co/image/602dd7b3a2ee3f3fd86c6c4f50ab9b5a82e23c59",
      "width": 160
    }
  ],
  "name": "Band of Horses",
  "popularity": 64,
  "type": "artist",
  "uri": "spotify:artist:0OdUWJ0sBjDrqHygGUxeCF"
}
```

Figure 15 - Tableau d'objets JSON récupérés via l'API

On peut ainsi récupérer le nom de l'artiste, sa popularité, ainsi que d'autres informations.

GESTION DE PROJET

Répartition des tâches

La tableau suivant récapitule les principales tâches effectuées au cours de ce projet, avec la durée estimée de la tâche en heures et la personne qui l'a exécutée.

Durée en heures	Tâches	Exécutant
2h	Initialisation du projet, barre de navigation et création de la page de connexion	Mariam
8h	Authentification à l'API	Alban, Juliette et Mariam
2h	Affichage du top artistes	Mariam
5h	Affichage du top titres	Mariam
2h	Détails artiste et titre	Mariam
2h	Recommandations selon les données Spotify	Mariam
2h	Recommandations selon nos critères	Mariam
2h	Ajout de filtres de recommandations	Alban
3h	Profil musical selon les critères de danceability, valence, energy et acousticness	Juliette
1h	Profil musical selon le critère de popularity	Juliette
2h	Détails des caractéristiques audio du profil musical	Alban
3h	Création d'une playlist et ajout de titres à la playlist	Mariam

Organisation

Au sein de ce projet, nous avons dû nous organiser très vite étant donné que nous appartenions à des groupes de TD différents. Ainsi, une fois notre choix d'API et sujet déterminés, nous avons fait un brouillon de comment nous imaginions l'application en termes d'interface graphique, ainsi que les fonctionnalités présentes.

Cela nous a permis de dresser une feuille de route sur le travail à mener et de tenir un document d'avancement pour que chacun soit au courant du travail effectué par les autres. Nous n'avons pas utilisé le système de branches sur Git car nous n'avions pas suffisamment de fonctionnalités pour que ce soit pertinent. De plus, nous avons travaillé en pair programming lorsque cela était possible afin de partager nos compétences techniques et nous aider mutuellement sur nos difficultés.

BILAN

Difficultés rencontrées

Une des premières difficultés que nous avons rencontrées concerne l'authentification à l'API à l'aide d'un compte utilisateur.

En effet, nous avions pris comme source la documentation d'expo (cf.

<https://docs.expo.io/guides/authentication/#spotify>) présentant la méthode à suivre directement pour l'API Spotify, mais la méthode que nous avons choisie (Implicit Flow) ne fonctionnait pas comme prévu. Nous avions des problèmes sur l'URI de redirection (URL de réponse qui redirige l'utilisateur sur l'application mobile en fournissant un *token* une fois connecté). Ce problème venait en fait de l'URI qui est propre à chaque machine et correspondait à l'URL utilisé par expo pour lancer l'application.

Un autre problème est survenu à la fin de notre projet concernant la déconnexion à l'application et à son compte Spotify. Nous souhaitions faire un bouton sur l'onglet profil permettant de revenir à l'écran de Connexion et qui permettrait de se connecter à un potentiel compte utilisateur Spotify différent, mais cela n'a pas été possible. Pourtant, une documentation était existante, mais l'utilisation de la méthode *revoke* n'étant pas expliquée et sans démonstration, même sur divers forums, de la façon de la mettre en place, nous n'avons pas réussi à la rendre fonctionnelle.

Améliorations

En considérant l'état actuel de l'application, quelques améliorations pourraient être effectuées.

Tout d'abord, lorsqu'un titre est affiché, il est possible de cliquer sur sa preview (bouton play) pour en écouter un extrait. Cependant, il est actuellement possible d'en écouter plusieurs en même temps, ce qui n'est pas forcément souhaitable. Il serait préférable alors que la preview d'un titre s'arrête lorsqu'une autre commence. Une autre amélioration serait de réussir à réaliser la déconnexion à l'application.

Conclusion

En conclusion, ce projet de développement mobile nous a permis de réaliser et d'avoir à façonner en autonomie une IHM mobile. Ainsi, nous avons dû mettre en place nos acquis techniques et différentes connaissances en choix ergonomiques. De plus, au sein de notre application, nous avons pu mettre en pratique un échange de données avec l'API Spotify, ce qui nous a également appris à mettre en œuvre des requêtes selon une documentation précise.

Enfin, nous avons réussi à développer toutes les fonctionnalités définies en début de projet (tops, profil musical et recommandations) et nous avons pu ajouter des fonctionnalités supplémentaires comme la création d'une playlist et la possibilité d'écouter un extrait d'un titre.