Documentation du projet PING - La boîte à son

Technologies utilisées dans la STACK.

BACK-END :

Le BACK-END repose sur `Django`, il s'agit d'un framework de développement web open-source, écrit en Python. Nous avons choisi cette technologie car elle encourage le développement de code propre et bien structuré à travers un modèle MVC (Modèle-Vue-Contrôleur).

En l'état, le back-end contient 4 applications :

- Une application `frontend` qui permet de faire la jonction entre le back-end Django et le front-end React.
- Une application `users` qui permet de gérer les comptes utilisateurs.
- Une application `spotify` et une application `deezer` qui permettent respectivement de gérer les appels aux API des platforms de streaming.

Ces applications contiennent des `models` qui sont liés à la base de données située à la racine du projet ainsi qu'à des vues qui sont liées aux endpoints du serveur.

Exemple : `/users/register_user` renvoie vers l'application `user` qui renvoie elle même vers la vue `RegisterUser`.

FRONT-END :

Côté front-end, nous utilisons `React`, React est une bibliothèque JavaScript populaire qui est principalement utilisée pour construire des interfaces utilisateur pour les applications web. Nous avons choisi ce framework car il permet de construire des composants réutilisables qui peuvent être facilement composés pour créer des interfaces utilisateur complexes.

Nous utilisons de plus d'autres framework avec React :

1. `Webpack` comme module bundler.
2. `Babel` comme compilateur & loader afin de rendre notre code Javascript compatible avec n'importe quelle navigateur.

3. `MUI (Material UI)` pour simplifier la création d'interfaces modernes à travers l'utilisation de composants entièrement customisables.

Plus de détail dans `/frontend/package.json`.

`/frontend` contient l'ensemble des fichiers utilisés par les clients :

- static : contient l'ensemble des fichiers statics dont :
  - Les fichiers CSS.
  - Les images.
  - Les bundles de nos javascripts.
- template : contient les fichiers html que DJANGO va rendre dans ses vues, le seul utilisé dans ce projet est `index.html` qui correspond au point d'entré du frontend.
- src : contient les composants React et le point d'entrée de l'application `index.js`.

Principe de fonctionnement

1. Un client effectue une requête HTML `GET` sur notre serveur Web Django, cette requête est redirigée dans `/frontend/urls` puis est ensuite redirigée vers la vue `index` qui rend le fichier `index.html`.
2. Dans le template HTML, on charge les fichiers statiques dont le CSS et surtout le fichier `/frontend/static/frontend/main.js`qui est le fichier Javascript obtenu après compilation de l'ensemble des composants React par Webpack.
3. Le composant `App.js` est enfin chargé et redirige le client à la fin sur le composant souhaité, par exemple `RegisterPage.js` pour une requête sur `/register`.

Guide d'installation et de lancement du projet

FRONT-END :

Il est important de commencer par la partie front-end.

Afin d'installer toutes les dépendances dans le front, il faut se rendre dans `/frontend` et effectuer un `npm install`.

```
cd frontend
npm install
```

Après que les dépendances se soient installées il est impératif de lancer la compilation Webpack, sans ça, les modifications apportées aux fichiers .js ne seront jamais prises en compte.

Rappel : le front-end rend uniquement le fichier compilé `/frontend/static/frontend/main.js`, ce fichier est issu de la compilation Webpack

Il faut donc effectuer la commande suivante dans `/frontend` :

```
npm run dev
```

BACK-END :

Le fichier `/requirements.txt` répertorie toutes les librairies python à installer afin de faire fonctionner le back-end. Il faut donc commencer par installer ces librairies.

Il faut ensuite générer la base de données en effectuant les commandes suivantes à la racine du projet :

```
python .\manage.py makemigrations
python .\manage.py migrate
```

On peut ensuite démarrer le serveur :

```
python .\manage.py runserver
```

Le serveur est maintenant accessible en : `localhost:8000`.

Attention, le serveur ne démarrera pas sans le fichier `/credential.json` qui contient l'ensemble des credentials utilisés dans les API de Spotify et de Deezer.

Remarque : à chaque modification des modèles, il faudra relancer les commandes suivantes :

```
python .\manage.py makemigrations
python .\manage.py migrate
```

Pour résumer, une fois l'installation du projet effectuée, il faut lancer deux choses :

1. Le serveur WEB à la racine du projet :

   ```
   python .\manage.py runserver
   ```

2. La compilation du front-end (dans /frontend) :

   ```
   npm run dev
   ```

On peut accéder à l'application web en `localhost:8000`.

Création des boîtes :

Afin de créer les boîtes à son il est nécessaire de créer un compte administrateur. Pour cela, il faut taper la commande suivante à la racine du projet :

```
python manage.py createsuperuser
```

Ensuite, se rendre sur le panel d'administration en `http://localhost:8000/admin`, se connecter et gérer les boîtes.

Une fois crées, les boîtes son accessibles en `http://localhost:8000/box/nomboîte`.

Modification des clés API

Le fichier `credentials.json` contient les identifiants utilisés pour se connecter aux différentes API. Il est constitué de 3 objets :

1. Le premier objet doit contenir les informations relatives à l'API `Spotify`.
2. Le second objet doit contenir les informations relatives à l'API `Deezer` pour la recherche de musique.
3. Le troisième objet doit contenir les informations relatives à `la seconde API Deezer`, il est utilisé pour permettre aux utilisateurs de se créer un compte avec Deezer.

```
{
  "Credentials": [
    {
      "client_id": "980501e3a9f6429e8c28276feba0aba2",
      "client_secret": "85fcfc284c79446c8c31534e583a4352",
      "redirect_uri": "http://127.0.0.1:8000/spotify/redirect"
    },
    {
      "app_id": "611424",
      "app_secret": "10adac87d908e1e0e18a95795a845e7e",
      "redirect_uri": "http://127.0.0.1:8000/deezer/redirect"
    },
    {
      "app_id": "615224",
      "app_secret": "d74ef0bef10f3d40f7e81fbc75c8d8c7",
      "redirect_uri": "http://127.0.0.1:8000/oauth/complete/deezer/"
    }
  ]
}
```