Analyse du code de Frédéric Auger

Kevin Varnier – 06/04/2018

Caractéristiques générales

Le code source analysé dans ce document est celui envoyé par Mr Tétu Martin à la date du 23 Mars 2018.

Cette analyse peut servir de complément à la documentation technique fournie auparavant.

Le programme d'application de recherche automatisée est très bien documenté ce qui facilite son utilisation par un autre développeur. On y retrouve un fichier readme.md à la racine du projet qui explique les prérequis et les méthodes pour utiliser les différentes fonctionnalités.

Le programme est écrit en JavaScript et fonctionne sur la plateforme Node Version 8. A l'intérieur du code source, de nombreuses parties sont également commentées cela facilite grandement la compréhension générale.

Cette solution est gratuite, et peut fonctionner sur plusieurs plateformes (MacOSX, Windows, Linux). Il peut être envisagé par la suite d'automatiser le fonctionnement du programme sur une machine allumée en permanence et dédiée à cette tache afin d'extraire des résultats en continues.

Le logiciel est actuellement en cours de développement, des problèmes peuvent survenir en cours de route si l'API interrogé ou le site web ont subi des changements.

Actuellement, le logiciel permet d'interroger les plateformes avec les paramètres suivants :

- Google Play Music: Album et Track
- YouTube: Movie, Album et Track
- Spotify: Album
- iTunes: Movie et Album
- *Netflix: Movie **

**N'ayant pas de compte Netflix, ce test n'a pas pu être effectué.*

Extrait du fichier index.JS

```
//Toutes les plateformes avec les types de médias exploitables
const CATALOGS = {
    "google_play_music": {
        catalog: GoogleMusicFetcher,
        types: ["album","track"]
    },
    "youtube": {
        catalog: YoutubeFetcher,
        types: ["movie","album","track"]
    },
    "spotify": {
        catalog: SpotifyFetcher,
        types: ["album"/*TODO:"track"*/]
    },
    "itunes": {
        catalog: ITunesFetcher,
        types: ["movie","album"]
    },
    /* TODO : Netflix à corriger/optimiser avant de réactiver*/
    "netflix": {
        catalog: NetflixFetcher,
        types: ["movie"]
    }
};
```

Authentification

Afin d'interroger les plateformes Youtube et Spotify, un compte développeur gratuit est nécessaire, j'ai utilisé mes propres identifiants pour les tests.
Netflix nécessite un compte utilisateur payant.

Google et iTunes ne nécessitent pas d'authentification.

Ces paramètres doivent être saisis dans le fichier ./credentials/index.js avant d'utiliser le programme

```
module.exports = {
    "netflix": {
        username: "",
        password: ""
    },
    "youtube": {
        key: ""
    },
    "spotify": {
        id: "",
        secret: ""
    }
}
```

Extrait credentials/index.js

Fichier de critères

Un fichier de critères au format .csv devant respecter une structure bien spécifique en fonction du type de média recherché est utilisée en entrer du programme.

Si ce fichier n'est pas au bon format ou que des éléments sont manquants, le programme retournera une erreur.

Movie nécessite le champ titre :

```
id;movie
1;"1:54"
2;"3 p'tits cochons 2"
```

Extrait Readme.md

Album nécessite les champs Artiste et album :

```
id;artist;album
1;"celine dion";"Loved Me Back To Life"
```

Extrait Readme.md

Track nécessite les champs Artiste, album et titre

```
id;artist;album;track
1;"celine dion";"Loved Me Back To Life";"Somebody Loves Somebody"
```

Extrait Readme.md

Spécificités Spotify :

- Les champs enregistrés pour Spotify sont :

```
const CSV_FIELDS_ALBUM = [
    "idCriterion",
    "totalResults",
    "items.id",
    "items.name",
    "items.album_type",
    "items.release_date",
    "items.artists.name",
    "items.artists.id"];
```

Extrait spotifyFetcher.js

- Si le champ "artiste" est égal à « Artistes variés » il est remplacé par « * »
- Si le champ "album" est égal à « EP » il est remplacé par « * »
- Le nombre maximum de résultats par requête est de 50
- Le paramètre « market » est pris en compte, mais n'influe pas sur les résultats

Spécificités Youtube :

- Les champs enregistrés pour YouTube sont :

Extrait youtubeFetcher.js

- Les catégories movie et track rechercheront des vidéos
- La catégorie album recherche des playlists

```
if (type === "album") {
    path = "playlists";
    listIds = parsedData.items.map((item) => item.id.playlistId);
}
else //movie et track
{
    path = "videos";
    listIds = parsedData.items.map((item) => item.id.videoId);
}
```

Extrait youtubeFetcher.js

- Les résultats observés sont très proches d'une recherche effectuée physiquement sur Youtube
- Si le champ "artiste" est égal à « EP » il est remplacé par une chaine de caractères vide
- Le paramètre « market » est pris en compte et influe sur les résultats
- Le nombre maximum de résultats par requêtes est de 50, pour le paramètre Track il est défini à 5

Spécificités ITunes

- Les champs enregistrés pour iTunes sont :

```
  "idCriterion",
    "term",
    "totalResults",
    "items.artistId",
    "items.artistName",
    "items.collectionCensoredName",
    "items.collectionExplicitness",
    "items.collectionHdPrice",
    "items.collectionId",
    "items.collectionName",
    "items.collectionPrice",
    "items.contentAdvisoryRating",
    "items.country",
    "items.currency",
    "items.kind",
    "items.longDescription",
    "items.primaryGenreName",
    "items.releaseDate",
    "items.trackCensoredName",
    "items.trackCount",
    "items.trackExplicitness",
    "items.trackHdPrice",
    "items.trackHdRentalPrice",
    "items.trackId",
    "items.trackName",
    "items.trackNumber",
    "items.trackPrice",
    "items.trackRentalPrice",
    "items.trackTimeMillis",
    "items.wrapperType"
```

Extrait itunesFetcher.js

- Le paramètre « market » est pris en compte et influe sur les résultats
- Si le champ "artiste" est égal à « Artistes variés » il est remplacé par une chaine de caractères vide
- Le nombre de résultats maximum par requête est de 200
- L'API est limitée à 20 requêtes/minute

Google Play Music

- Les champs enregistrés pour Google Play Music sont :

```
const CSV_FIELDS = [
    "idCriterion",
    "term",
    "url",
    "title",
    "artist",
    "price",
    "rating"
];
```

Extrait googleMusicFetcher.js

- Le paramètre market n'est pas pris en compte, il faudra utiliser un VPN pour simuler la présence dans un autre pays
- On récupère uniquement le premier article trouvé
- Si le champ "artiste" est égal à « Artistes variés » il est remplacé par «Various artists »
- Pour éviter les blocages, il est nécessaire de simuler une navigation « humaine »

```
        //Attendre 1 seconde après chaque critère pour éviter de se faire bloquer...
        await page.waitFor(1000);
    } else {
        console.error("Page de recherche bloquée, attendre 5 secondes avant de ré-essayer...");
        await page.waitFor(5000);
        throw Error('Page de recherche bloquée.');
    }
```

Extrait googleMusicFetcher.js

Notes personnelles

@param

Je retrouve dans le code l'utilisation de « @param », je ne connais pas cette syntaxe, est-ce une spécificité du langage ? Différentes variables passent apparemment par là. C'est notamment via ce paramètre que le logiciel peut être lancé avec différentes options.

```
    /**
     * Initialise baseUrl pour les appel d'API selon la spécificité de chaque sous-classe
     * @param {String} baseUrl Domaine URL pour les appels d'API ou options
     * @param {String} catalogName ID du fetcher
     * @param {Object} opts Autres options à ajouter à chaque appel (ex. { headers : {auth : { bearer: token }} }) (facultatif)
     */
```

_nomDeClasse

Que signifie cette syntaxe ? A priori, cela permet de différencier la classe de son instance

```
this._catalogName = catalogName;
```

Module export

Utiliser dans de nombreux fichiers, surement pour définir que ce fichier est un module à importer. Comment fonctionnent les dépendances entre les différents fichiers du programme ? Les « require » fonctionnent comme des imports ?

```
module.exports = GoogleMusicFetcher;
```

Gestionnaire de code source

Un gestionnaire de code source comme Bitbucket sera-t-il utilisé par la suite ? Cela me permettrait d'avoir toujours la dernière version du code et d'échanger plus facilement avec l'auteur.

Mes modifications pourraient également être validées et intégrées plus facilement plutôt que par mail.