## ligne horizontale

Module M4103C – [***Playlist Travel****]*

**─**

Noms/Prénoms du binôme

Comte Gabriel

Crusson Cyprien

# 

# Introduction

Ce projet a été réalisé par Gabriel Comte et Cyprien Comte, dans le cadre d’un module de Client Riche encadré par Jean-François Berjdugin . Étant donné que nous sommes tous deux passionnés par la musique, nous avons décidé de nous orienter vers une API musicale. Nous avons ensuite réfléchi à une plus-value original pour notre site et après plusieurs idées, nous avons décidé de faire une application web, qui à partir d’un itinéraire de voyage entre deux points et d’un genre de musique, nous créer une playlist avec des musiques dans ce genre, de la durée du temps de trajet de l’itinéraire.

# Vision fonctionnelle

* Présentation de l’application
* Présentation des fonctionnalités
* **POUR ALLER PLUS LOIN** : Maquettes d’écrans de l’application. Quelques outils ici -> <https://korben.info/18-outils-gratuits-mockups.html>

## Présentation l’application

Comme expliqué ci-dessus, notre application permet donc de créer une playlist de musique, de la durée d’un itinéraire, entre deux points telles que des villes et d’un genre de musique donné. Pour toute la partie musicale, nous avons utilisé l’api de Spotify car elle nous permettait de récupérer des musiques, créer des playlists ainsi que de gérer nos playlists en ajoutant des morceaux. Pour la gestion de l’itinéraire nous avons utilisé l’API GéoService, nous permettant de calculer un itinéraire entre deux positions et ainsi récupérer le temps de trajet (dans notre cas en voiture). Enfin pour facilité l’expérience de l’utilisateur, nous avons également utilisé l’api Adresse de Data.gouv.fr dans le but de trouver plus facilement les villes à entrer pour l’itinéraire. Nous pouvons donc lire les playlists une fois créées et également consulter toutes les playlists de l’utilisateur présentent sur son compte.

## Présentation des fonctionnalités

Nous avons donc une première page principale dans laquelle nous avons un formulaire avec 4 champs de saisi : départ, qui correspond à la ville de départ du trajet ; arrivée qui est donc la ville de destination ; genre, pour sélectionner le style de musique de la playlist et le nom de notre playlist. Nous utilisons l’API de Spotify dès la création de la page, afin de récupérer les styles de musiques les plus populaires en France afin de les assigner pour l’autocomplétion du champ de style. Les deux champs des villes déclenche un évenement à chaque fois qu’on écrit du texte dedans, afin d’appeler l’API d’adresse de data.gouv pour trouver les villes qui corresponde au champ de texte et les assigner également à l’autocomplete.

Quand on appuie sur le bouton créer, notre application va dans une premier temps générer un itinéraire à partir de la ville de départ et d’arrivée (étant donné que nous utilisons une API gratuite, il faut savoir qu’elle ne reconnaît pas certaines positions. Il existait d’autres API d’itinéraires mais la plupart étaient payantes aux nombres de requêtes). Une fois l’itinéraire calculé, on récupère le temps de trajet total ainsi que le style de musique choisi et le nom de la playlist. Avec ces informations, nous pouvons donc faire appel à l’API de Spotify pour faire une requête en post et créer une nouvelle playlist sur le compte de l’utilisateur.

Nous récupérons d’abord une playlist créée par Spotify contenant les musiques les plus populaires du genre sélectionné puis nous sélectionnons aléatoirement des morceaux dedans et les ajoutons à la nouvelle playlist créée, jusqu’à ce que le temps total des morceaux ajoutés soit égal à celui de l’itinéraire. Les playlists ainsi créées sont également stockées via le stockage local afin d’avoir un historique persistant sur la machine. Nous pouvons ainsi voir les dernières playlists avec un lecteur intégré pour la lancer et écouter les sons. Nous avons également une deuxième page qui nous permet de charger toutes les playlists présentent sur le compte utilisateur pour les afficher avec également un lecteur intégré.

# Vision technique

* Schéma d'architecture générale (points d’entrée, découpage, communication externe avec l’api …)
* Explication des choix techniques (framework ...)
* **POUR ALLER PLUS LOIN :** Diagrammes de séquences (par exemple sur le stockage et la réutilisation des données du LocalStorage, sur les interactions de votre code avec l’api ...)

Nous avons plusieurs points d’entrée :

* Spotify API :
  +  : cette route nous permet de récupérer un token Spotify. Nous mettrons ce token dans le header des différentes requêtes pour que nous puissions utiliser l’API.
  +  : cette route nous permet de créer une playlist sur notre compte Spotify. Il nous faut mettre notre id user dans l’url et notre token dans le header de la requête, pour notre url nous la remarquons entre /users et /playlists. Nous obtenons cette id grâce au lien « https://developer.spotify.com/console/get-current-user/ ». Nous obtenons un json où nous allons nous servir de l’id de cette nouvelle playlist. Mais nous obtiendrons aussi son lien son nom et d’autres informations.
  +  : cette route nous permet de récupérer toutes les playlists du compte Spotify. Nous récupérons alors un json qui comporte les id de chaque playlist, leurs images et d’autres informations. Pour appeler cette route nous avons besoins de mettre notre token dans le header de la requête.
  +  : cette route nous permet d’effectuer une recherche sur Spotify. Nous utilisons la query « playlist : ${catégorie} » qui nous permet de trouver une playlist qui a pour mot clé le nom qu’on met, dans notre cas la catégorie. De plus nous devons mettre un type qui peut être « track » pour une music. Dans notre cas nous utilisons le type « playlist ». Nous avons toujours besoin de mettre le token dans le header. Cette route nous permet de récupérer toutes les playlists avec cette catégorie. Nous récupérons pour chaque playlist son id, son image et sa description et d’autres informations.
  +  : Cette route nous permet de récupérer toute l’information nécessaire d’une playlist. Pour sélectionner la playlist dont nous voulons les informations il faut passer son id par la route. Il faut alors mettre notre token dans le header de la requête. Cette url nous permet d’accéder à l’id et les musiques que contiennent une playlist.
  +  : Cette route nous permet l’ajout d’une musique dans la playlist sélectionner. L’url requiert l’id de la musique que nous voulons ajouter et l’id de la playlist où nous voulons la musique. Cette route utilise la méthode POST il faut alors renseigner dans la requête qu’il faut utiliser la méthode POST et renseigné dans le header notre token.
  +  : Cette route nous permet de récupérer les 50 premières catégories de Spotify. Il nous suffit de renseigner le pays dans notre cas FR et le nombre de catégories que nous voulons c’est-à-dire 50. Nous récupérons alors le nom des 50 premières catégories. Il nous faut toujours renseigner notre token dans le header de la requête.
* GéoService API :
  +  : Cette route nous permet de récupérer les coordonnées GPS de la ville indiqué dans la requête. Pour cette route aucun token n’est à mettre.
  +  : Cette route prend un point d’entrée avec une longitude et une lattitude et pareil pour le point d’arriver. Il calcul alors l’itinéraire en voiture et nous pouvons récupérer le kilométrage, le temps de trajet et d’autres informations. Pour notre application nous n’aurons besoins que du temps de trajet. Aucun token c’est à mettre pour que l’application soit fonctionnelle.
* API gouvernementale
  +  : Cette route nous permet d’avoir le nom des villes et leur code postal. Nous avons besoin pour cela de renseigner dans la querry le nom de la ville rechercher. Il nous faut aussi renseigner une information liée à l’autocomplete. Aucun token n’est à renseigner.

NB : Pour l’utilisation de l’API Spotify il est nécessaire de changer la valeur de la variable clientId et clientSecret par vos propres valeurs. De plus dans le getToken à la ligne 17 du fichier spotify.js il nécessaire de changer la valeur après « &refresh\_token= » en allant sur ce site : <https://getyourspotifyrefreshtoken.herokuapp.com/>

Et en selectionnant « Select All ». Avant cela vous devrez aller dans la DashBoard de Spotify aller dans « edit Settings » et mettre ce lien <https://getyourspotifyrefreshtoken.herokuapp.com/callback>

**Dans redirect uris. Apres cela vous pourrez alors générer votre refresh token et le mettre dans le code à la ligne 17 apres le « &refresh\_token= ».**

## Explication des choix techniques :

Nous avons donc choisi le Framework VueJs pour notre projet afin de faciliter la construction et la gestion des différents éléments de notre application. En effet, grâce à VueJs, cela a été plus rapide de mettre en forme l’architecture de notre site, à travers les différentes fonctionnalités du framework. Dans un premier temps cela nous à permis de créer des « composants » c’est à dire des extensions d’éléments HTML, auxquels nous associons des comportements spécifiques et que l’on peut donc réutiliser partout. Par exemple dans notre site nous avons un composant Playlist qui nous permet d’afficher les différentes playlists, avec leurs informations, un lecteur de sons intégré etc. Du fait que nous utilisons et affichons beaucoup de playlists, il était très pratique de créer une fois ce composant afin de le réutiliser dès qu’une playlist est créée. De plus avec VueJs nous pouvons utiliser des boucles directement dans le code HTML, à travers les « templates » et les « models ». Les templates représentent les balises HTML des composants et les nous pouvons déclarer des boucles, des conditions etc. à l’intérieur. Nous pouvons également utiliser des variables à travers un système de modèle, que nous pouvons aussi bien utiliser dans les templates ou les méthodes des composants. Dans notre cas nous avons pu utiliser des boucles pour afficher toutes les composants playlists ainsi créés.

Nous avons également utilisé le framewrok materialize, afin de créer des champs de saisie qui bénéficie de l’autocomplétion, afin de faciliter les recherches et aider les utilisateurs à cibler leurs attentes. Pour la mise en page et la mise en forme de notre site, nous avons utilisé le framework bootsrap notamment pour faciliter le positionnement et améliorer l’ergonomie de notre site.

# Conclusion

Un bilan de votre module client riche. Ce que vous en avez retenu, ce qui vous a marqué …

Le fait de réaliser cette application web, sur un temps imparti relativement court, à l’aide d’API REST, a été un exercice enrichissant qui nous a permis de mettre en pratique les connaissances acquises lors des séances de TD. En effet, nous avons pu reprendre les points étudiés, dans un cas concrets pour les étudier sous un angle nouveau. Au départ, nous avons eu des difficultés avec l’API de Spoitfy, car elle nécessite un niveau d’authentification et l’utilisation de token pour accéder aux différentes requêtes, mais une fois cette étapes surmontée, nous avons pu avancer effacement et apporter nos solutions techniques. C’était formateur de faire une API à l’aide d’une vraie API, comme celle de Spotify qui est une application largement utilisée et de manipuler ainsi de vraies données.