2.1 - Préparation des données

Afin de pouvoir travailler sur les données de manière sereine, nous avons choisi de créer une structure de données enregistrant toute les données à note disposition et étant itérable afin de pouvoir parcourir les données de cette manière, sans avoir à parcourir le tableau à chaque fois.

Pour cela, nous avons tout d'abord envisagé et mis en place un système de Hashmap, mais nous avons fini par nous rendre compte que cela poserait problème avec le système de serveur que nous envisagions de mettre en place. En effet, la transmission des données entre le client et le serveur ne peut s'effectuer que sous format JSON ; nous avons donc décidé d'utiliser le même idée de Hashmap, mais avec une structure basée sur des objets natifs, ce qui nous donnes la structure de données suivantes :

```
{
	"fr": {
	"https://www.spotontrack.com/track/julien/41177241": {
		"title": "Julien",
		...,
		"positions": [{
			"date": "2018-09-26",
			"position": 3
		}, {
			"date": "2018-10-03",
			"position": 4
		}, ...
		]
	},
	"https://www.spotontrack.com/track/baisers-voles/40595594": {
		...
	},
	...
	},
	"jazz": {
	...
	},
	...
}
```

Cette stratégie induit le fait que ces données seront dupliquées dans le cas où une musique est présent dans plusieurs playlist (puisque les données relatives à la musique sont associée à une playlist spécifique). Toutefois, après une étude de données, nous n'avons trouvé aucune occurrence d'une telle musique. Si elle existe, elle semble n'arriver que très rarement, et le nombre de duplications engendrées serra alors négligeable (et ne mettra pas en péril les opérations effectuées sur les données).

A ce stade, nous nous sommes dit que cette structure nous permettait d'y ajouter les données de tracks/ En effet, chacune est spécifique à un url et peut donc s'ajouter avec au coté du title. Nous avons donc ajouté les données de Dancabilité, valence...

```
{
      "fr": {
      "https://www.spotontrack.com/track/julien/41177241": {
            "title": "Julien",
            "bpm": 96,
            ...,
            "positions": [{
                  "date": "2018-09-26",
                  "position": 3
            }, {
                  "date": "2018-10-03",
                  "position": 4
            }, ...
            ]
      },
      "https://www.spotontrack.com/track/baisers-voles/40595594": {
            ...
      },
      ...
      },
      "jazz": {
      ...
      },
      ...
}
```

CODE La création des cette structure est comprise dans les fonctions **parseData2**

Position-pic

Afin de calculer la position-pic de chaque chanson, il nous suffit de parcourir la structure de données évoquées via des itérateurs natifs de javascript

Indicateur si la position-pic est 15

Pour associer un booléen à chaque chanson indiquant s'il a été au moins une fois au dessus de la 15ème position dans une playlist, nous parcourons simplement le

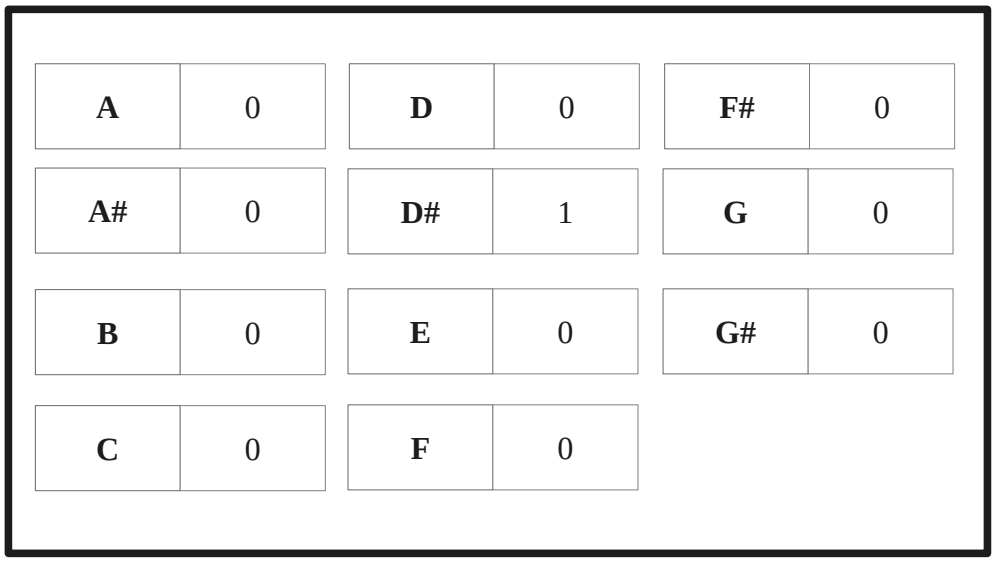
2.1 – Analyse exploratoire

Certains données ne sont pas adaptées pour l'analyse : il s'agit de le key et le mode. En effet, ces derniers son des variables catégorique, il faut donc les transformer.

Transformation du mode : le mode ne peut prendre que deux valeurs : Minor ou Major. Nous avons donc pu **transformer cette variable en booléen** : si le mode est

Transformation de la key : la key en revanche peut prendre énormément de valeurs (12). Pour régler ce problème, il nous faut **créer une colonne pour chaque valeur possible** ; colonne pouvant prendre comme valeur soit true (si la key est bien de cette valeur), soit false (si elle ne l'est). Chaque musique possédera donc une colonne à true, et 11 colonnes à faux.

key	D#



A	0	**D**	0	**F#**	0
A#	0	**D#**	1	**G**	0
B	0	**E**	0	**G#**	0
C	0	**F**	0		

Normalisation

A ce stade, les variables n'était pas utilisables pour faire des statistiques avancées car elles n'étaient pas comparables : le BPM, par exemple, peut aisément monter à plus de 100 alors que les variables définies si dessous ne peuvent intrinsèquement pas dépasser 1. Il nous faut donc réaliser une normalisation sur les variables.

Méthode de normalisation : Pour chaque valeur, nous lui soustrayons la moyenne afin qu'elles soient toutes centrées autour de 0, puis nous divisons par l'écart type afin que toutes les variables soient dans un interval à la même échelle.

Nous avons donc dû pour cela créer des bouts de codes s'occupant de calculer la moyenne pour chaque valeur (sommes toute les valeurs et divise par la taille du tableau associé) et l'écart type (epxlications), puis normalisant les valeurs grâce aux statistiques trouvées.