

Dumortier Maxime  
Lacroix Théo  
Soyer Mathieu



# WIPE SOUND

PROJET DE SYNTHÈSE 2016

---

# SPECIFICATIONS DU PROJET

## Table des matières

Spécifications du projet	4
Contexte	4
Description du client	5
Description de wipessound et Objectifs	5
Description Générale	7
Description Fonctionnelle	8
Spécifications stratégique	9
Enjeux	9
Risque	10
Sprints en méthode agile	11
Prévisions – Matrice de complexité	11
Découpage initial	15
Realisation des sprints, problemes et solutions apportees	16
Sprint 1 «Recherche»	16
Sprint 2 «Application»	18
Sprint 3 – partie 1 «Côté serveur»	21
Sprint 3 – partie 2 «Finalisation»	25
Comparaison : objectifs et réalisation	27
Spécifications organisationnelles	28
La méthode scrum	28
L'équipe	29
Les rôles	30
Spécifications des tests	31
Tests fonctionnels	31

# SPECIFICATIONS DU PROJET

Sprint 1:	31
Sprint 2:	31
Sprint 3 – partie 1) :	32
Sprint 3 – partie 2):	32
Tests unitaires	33
Evolutions futures	34
Conclusion	34

## Spécifications du projet

### CONTEXTE

En premier lieu, il s'agit de présenter le contexte du projet. Au mois de mars 2015, nous avons décidé de participer à un concours d'idées nommé la « foire aux neurones » organisé par le VIPE (structure qui accompagne l'implantation d'entreprise innovante sur le territoire vannetais). Nous avons choisi de présenter l'idée de WipeSound, une application mobile citadine (nous y reviendrons par la suite). Nous avons eu la chance de remporter l'idée d'or.

Ensuite, nous présentions WipeSound à deux reprises, d'abord lors de la soirée des talents organisée dans le cadre du printemps de l'entreprise, puis à l'IUT lors d'une conférence dédiée à l'entrepreneuriat. Ces trois oraux nous ont permis de confronter notre projet face à différents publics et nous avons remarqué un engouement autour de WipeSound. Qui plus est, nous voulions réellement créer l'application dans un futur proche.

C'est pourquoi, nous avons pensé à réaliser WipeSound en tant que projet de synthèse. Une fois notre demande acceptée, nous avons bénéficié d'un client fictif (puisque nous sommes en réalité les principaux intéressés). Nous en sommes arrivés à la conclusion que notre maître d'œuvre (Monsieur Lelain) jouerait également le rôle de client.

A noter que nous avons également présenté WipeSound pour le module de création d'entreprise avec Madame Mannevy, Monsieur Baudont et Monsieur Etrillard. En définitive, nous avons travaillé sur notre projet sous différents angles (économique, commercial et technique), ce qui nous a davantage motivés afin de mener à bien notre projet de synthèse.

# SPECIFICATIONS DU PROJET

## DESCRIPTION DU CLIENT

Comme expliqué ci-dessus, notre client était également notre maitre d'œuvre, en loccurence Mathieu Lelain. Nous lui avons envoyé nos rendus comme si il s'agissait d'un client extérieur. Ainsi, nous avons échangé avec notre client, maitre d'œuvre de surcroît, tout au long du projet pour ré-orienter notre travail lorsque nous nous détachions de notre objectif initial.

Bien que nous n'avions pas de réel client extérieur, notre projet s'est parfaitement déroulé. En outre, nous sollicitons monsieur Lelain afin de discuter des problèmes rencontrés. Ainsi, lors du Sprint 3, nous le rencontrions trois fois par semaine.

## DESCRIPTION DE WIPESOUND ET OBJECTIFS

Nous allons ici expliquer notre projet WipeSound, mais avant toute chose, établissons une liste de définitions des mots que nous serons amenés à employer à de nombreuses reprises au cours de la présentation de notre projet.

**WipeSound** : WipeSound est le nom de l'application mobile que nous avons développée.

**Genres musicaux** : Nous parlons ici des genres musicaux classiques que sont la Pop, la Folk, la Country, le Rock, le Métal, le Rap, le Jazz, le Blues, etc...

**Profil musical** : Le profil musical désigne les informations d'un utilisateur que nous récupérons en analysant sa bibliothèque musicale Spotify. Nous obtenons ainsi les trois genres préférés de l'utilisateur.

# SPECIFICATIONS DU PROJET

**Moyen localisation :** Les moyens de localisation désignent ici les technologies utilisées afin de localiser les utilisateurs de l'application WipeSound. Nous avons créé la première version de WipeSound en utilisant le GPS, la 3G, et le wifi.

**PlayList du lieu:** La PlayList du lieu compatible désigne les musiques présentes dans la Playlist du magasin, et qui sont donc susceptibles de passer dans le magasin.

**Transition musicale :** La transition musicale fait référence au passage d'une musique à une autre dans la PlayList du lieu compatible.

**Lieu compatible :** Le lieu compatible (ou lieu reconnaissant l'application) désigne une zone délimitée dans laquelle l'utilisateur de notre application pourra être localisé afin que son profil musical soit utilisé pour affiner la transition musicale. Un utilisateur n'a d'impact sur la transition musicale d'un lieu compatible uniquement lorsqu'il y est localisé.

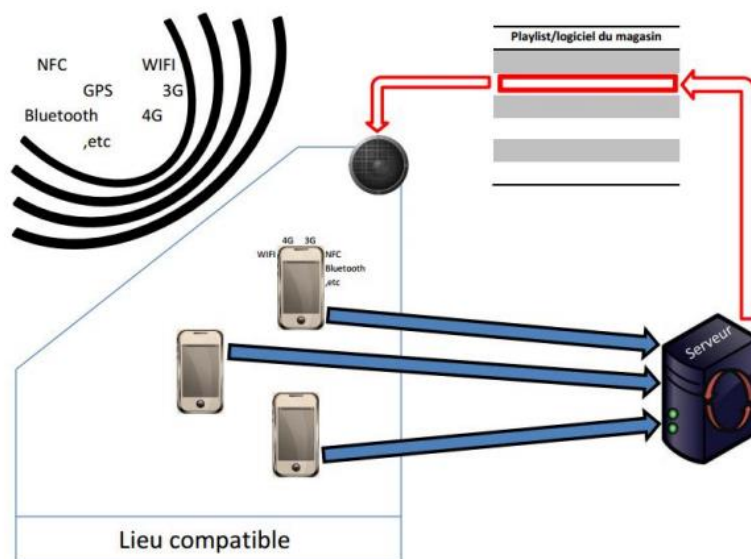
# SPECIFICATIONS DU PROJET

Nous avons défini les termes que nous serons amenés à utiliser afin de décrire notre système. Nous pouvons maintenant établir une description générale de Wipesound.

## DESCRIPTION GENERALE

Wipesound propose d'adapter la musique au profil musical des individus présents en temps réel dans un lieu compatible avec notre application. Lorsqu'une personne est localisé dans un lieu compatible, son profil est pris en compte dans le choix de la (ou des) prochaine(s) musique(s) et ce, tant qu'il est localisé dans ce lieu. Ainsi, nous connaissons les goûts musicaux des individus présents et permettons la transition vers une musique adaptée.

Ainsi notre système peut être représenté de la manière suivante :



Transfert du profil musical, des données de géolocalisation (latitude et longitude) et d'une donnée propre à ce client (pseudo ou adresse mail par exemple)



Calcul du profil moyen



Emission de signaux



Enceintes

# SPECIFICATIONS DU PROJET

## DESCRIPTION FONCTIONNELLE

Notre solution comprend donc dans son ensemble :

- ➔ Une **application mobile** qui permet aux utilisateurs d'afficher leur profil musical.
- ➔ Un **moyen de localisation** nous permettant de localiser les individus lorsqu'ils entrent dans un lieu reconnaissant l'application.
- ➔ Une **interface** pour le lieu compatible, permettant de garder la main sur les transitions musicales.
- ➔ Un **serveur** qui calcule, en fonction des profils musicaux des individus présents dans le lieu, la musique de la PlayList qui plaira à un maximum de personnes.



## Spécifications stratégique

### ENJEUX

Afin de préparer au mieux nos présentations de WipeSound, nous nous sommes renseignés à plusieurs reprises sur les enjeux liés à la mise en œuvre d'un tel projet.

L'essentiel des enjeux liés à notre projet s'axe sur le marketing. En effet, la musique constitue depuis longtemps un instrument largement intégré dans les pratiques et les politiques marketings. Selon Richard F. Yalch, professeur à l'université Washington, « Les consommateurs passent plus longtemps dans le magasin et ne se soucient plus autant du montant dépensé lorsqu'ils sont soumis à une musique qui correspond à leur goûts ». Suite à la diffusion d'une musique familière, il se crée une envie chez le consommateur de revenir sur le lieu de vente.

D'autres études ont montré que pour une personne entrant dans un magasin, la musique est un élément aussi important que les caractéristiques plus classiques comme la compétence des vendeurs, leur disponibilité ou même le prix.

Ainsi, WipeSound, en adaptant la musique aux consommateurs présents dans un magasin, joue un rôle déterminant sur l'expérience client. Notre système constitue un argument marketing différenciant pour les magasins compatibles.

## RISQUE

Si les enjeux d'un tel projet sont clairs et définis, les risques sont eux, plus abstraits.

En effet, le risque majeur lié à la mise en œuvre de WipeSound repose sur la compatibilité des technologies, bibliothèques et langages que nous utilisons.

Ainsi, pour sonder la bibliothèque Spotify de nos utilisateurs, nous récupérons les id's des musiques présents dans la PlayList et nous réalisons une série d'appel à l'API Echo Nest. Spotify et Echo Nest ont développé un partenariat qui permet de croiser leurs données. Dans le cas où leur partenariat venait à s'interrompre, notre système ne serait plus fonctionnel. C'est pourquoi nous avons identifié cette dépendance comme un risque pour notre système puisque rien n'indique que leur partenariat perdure dans le temps.

## Sprints en méthode agile

Nous détaillons ici les sprints du projet et leurs dates. Dans un premier temps, nous reviendrons sur nos objectifs initiaux, en analysant notre matrice de complexité répertoriant l'essentiel des fonctionnalités de notre système. Ensuite, une explication générale du découpage réel de notre projet. Puis, une explication plus détaillée de la mission réalisée lors de chaque sprint, des problèmes rencontrés et des solutions apportées. Enfin nous comparerons nos objectifs initiaux et nos rendus finaux.

### PREVISIONS – MATRICE DE COMPLEXITE

Nous expliquons ici nos prévisions initiales. Nous avons découpé le travail en tâches et fonctionnalités à effectuer. Ci-dessous, la liste des tâches que nous voulions effectuer et les fonctionnalités à développer lors de notre projet :

(T) Tâches (F) Fonctionnalités

En rouge, les fonctionnalités qui étaient attendues à la fin de notre projet.

En violet, les fonctionnalités que nous considérons comme optionnelles.

#### A. (T) Développement de l'application Android.

(F1) Création de profils utilisateurs.

(F2) Se connecter avec Spotify lors de la création du compte WipeSound (voir API)

(F3) Renseigner ses goûts musicaux manuellement (sans compte Spotify)

(F4) Pré-remplir son profil WipeSound via la musique de son téléphone.

(F5) Activer / Désactiver WipeSound.

(F6) Affiner son profil (Artiste, albums, titres)

(F7) Visibilité des lieux compatibles autour de l'utilisateur.

(F8) Visibilité de la musique à l'écoute.

#### B. (T) Mise en place d'une Base de données (ajout/suppression d'utilisateurs).

#### C. (T) Mise en place d'un serveur.

# SPRINTS EN METHODE AGILE

(F9) Algorithme simple de choix de musique dans une playlist donnée.  
(Algorithme qui fonctionne côté serveur). L'objectif ici est de réaliser un algorithme permettant un choix de musique basique à partir de deux profils musicaux et d'une liste donnée de musiques.

D. (T) Délimiter des zones géographiques afin que le profil musical d'un utilisateur ait un impact sur la musique du lieu dans lequel il se trouve.

(F10) Géolocalisation des utilisateurs de WipeSound.  
(F11) Algorithme plus complexe de choix de musique qui prend en compte les utilisateurs présents dans la zone géographique concernée.

E. (T) Développement de l'application Web pour le lieu compatible (interface).

(F12) Possibilité d'établir plusieurs playlist pour le magasin (une pour chaque genre par exemple)  
(F13) Obtenir des statistiques sur les utilisateurs.  
(F14) Suggestion de titres à rajouter pour le magasin.  
(F15) Établir des restrictions pour le passage de musique.  
(F16) Établir des niveaux de priorités aux différents profils utilisateurs

# SPRINTS EN METHODE AGILE

Niveau de priorité <hr/> Complexité	4	3	2	1
1	A (F3)			A (F5)
2	A (F1, F2) B			A (F8)
3	C (F9)	D (F10, F11)	E (F12, F13)	A (F4, F6, F7) E (F14, F15, F16)
4				

*Matrice de complexité (Tâches & fonctionnalités)*

Afin d'identifier avec précision l'ordre dans lequel nous devons réaliser les tâches de notre projet, nous avons utilisé une matrice de complexité. La matrice ci-dessus classe donc les différentes tâches du projet (et les fonctionnalités induites) en fonction de leur complexité et de leur niveau de priorité.

Ainsi, nous observions que certaines tâches nous paraissaient moins complexes que les autres tout en étant fortement prioritaires : Création de profil utilisateur et connexion (A (F1, F2, F3), B).

Il s'agissait donc de réaliser ces tâches-ci dans un premier temps. Ensuite, il a fallu réaliser les tâches prioritaires plus complexes : Mise en place du serveur et délimitation de zone géographique pour géo localiser l'utilisateur (C (F9), D (F10, F11)).

Arrivé à ce stade, nous aurions réalisé l'ensemble des fonctionnalités que nous avons considérées comme obligatoires (en rouge). Si le temps nous le permettait, nous

# SPRINTS EN METHODE AGILE

pourrions alors réaliser les tâches moins prioritaires et peu complexes telles que l'activation de son profil ou la possibilité de voir la musique à l'écoute (A (F5, F8)). Enfin, nous pourrions réaliser d'autres tâches moins prioritaires et plus complexes telles qu'affiner son profil (A (F6)) ou encore une suggestion de titres à ajouter à la playlist du lieu compatible et diffuseur de musique (E (F14)).

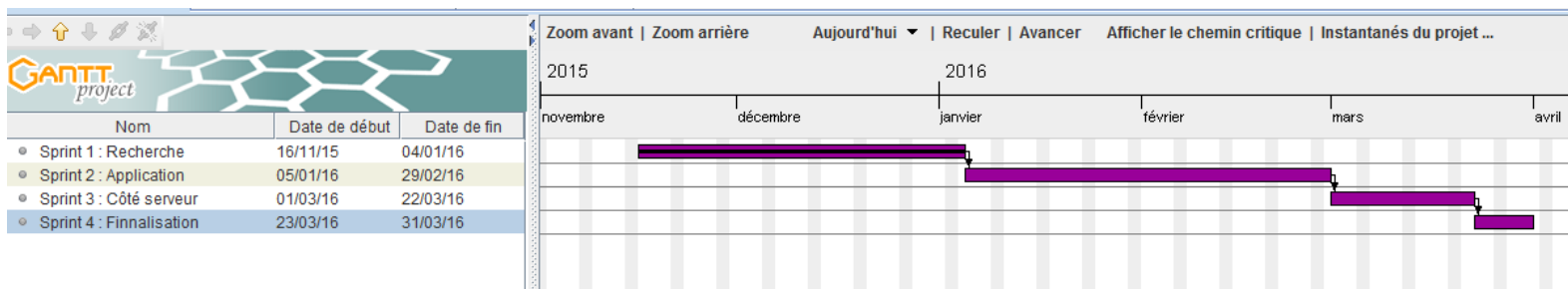
Si nous parvenions à réaliser l'ensemble de ces fonctionnalités, nous aurions à la fois réalisé les fonctionnalités obligatoires mais également celles que nous considérons comme optionnelles (en violet).

A noter que nous procéderions dans ce projet en méthode agile afin de nous permettre une certaine flexibilité au niveau des fonctionnalités que nous allions développer. L'idée principale était donc de construire « brique par brique » notre projet tout en testant au fur et à mesure nos fonctionnalités. Nous pensions déjà par exemple à deux versions pour notre algorithme ; une première assez « simple » que nous testerions et ferons fonctionner avec des profils basiques, puis une seconde, plus complexe capable de prendre d'autres paramètres en compte pour le choix des musiques.

# SPRINTS EN METHODE AGILE

## DECOUPAGE INITIAL

Nous avons expliqué ci-dessus nos objectifs pour ce projet. Nous allons maintenant décrire le découpage réel des tâches et le planning qui en découle.



*Planning Gant de nos 3 sprints*

Ainsi, nous avons identifié 4 grands axes de travail qui déterminent nos 3 sprints :

- Sprint 1 (15/11/15 → 04/01/16) «Recherche»
  - Recherche, installation et prise en main de l'ensemble des outils nécessaires. Réalisation d'une première ébauche de l'application exploitant uniquement les données publiques liées au compte Spotify de l'utilisateur.
- Sprint 2 (04/01/16 → 27/02/16) «Application»
  - Création de l'application (Connexion avec Spotify, récupération des trois genres préférés de l'utilisateur), utilisation de SonarQube, tests fonctionnels.
- Sprint 3 – partie 1 (27/02/16 → 21/03/16) «Côté serveur»
  - Création des Web Service, envoi des genres musicaux au serveur, envoi de la géolocalisation, algorithme côté serveur pour le changement de musique.
- Sprint 3 – partie 2 (21/03/16 → 28/03/16) «Finalisation»
  - Création du .apk via Cordova, «Assemblage» de toutes les parties du projet (application mobile, serveur, transition musicale sur Spotify), tests fonctionnels divers, design application, création d'une page web pour mettre en évidence la transition musicale qui s'effectue.

# SPRINTS EN METHODE AGILE

## REALISATION DES SPRINTS, PROBLEMES ET SOLUTIONS APPORTEES

### Sprint 1 «Recherche»

(15/11/15 → 04/01/16)

Lorsque nous avons commencé à réfléchir sur les moyens à utiliser pour notre projet, nous nous sommes rendus compte qu'il allait falloir se former à de nombreuses technologies. Nous avons établi la liste des outils/technologies qu'il était nécessaire de prendre en main.

Nous avons donc approfondi nos connaissances en Javascript, appris le fonctionnement de Cordova, AngularJS, jQuery et NodeJs (Back-end). Ces technologies nous ont permis d'obtenir des premiers résultats rapidement. Néanmoins, nous avons parfois dû éclaircir certains points. Notamment lorsque nous devions choisir quelle technologie utiliser pour certains problèmes spécifiques.

A noter que nous avons également travaillé sur le Framework Ionic, que nous avons finalement laissé de côté pour le sprint 1, puisqu'il rajoutait une couche technique supplémentaire. Néanmoins, nous avons utilisé Ionic plus tard dans le projet (sprint3).

Durant cette période, nous avons également analysé les API's dont nous avons besoin. Par exemple, il a fallu détailler sur papier le fonctionnement de l'API Spotify afin de penser à la meilleure solution pour récupérer les genres préférés de l'utilisateur.



# SPRINTS EN METHODE AGILE

## SPRINT 1 : PROBLEMES ET SOLUTIONS

PROBLEMES RECONTRES	SOLUTION APPORTEE	SPRINT	DATE
(1)	A	1	27/11/15 au 30/11/15
(2)	B	1	15/12/15
(3)	C	1	18/12/15 au 04/01/16

(1) PROBLEME : Le premier problème bloquant que nous avons rencontré lors du Sprint 1 concernait les librairies proposées par Spotify. En effet, de nombreuses bibliothèques portaient des noms similaires et semblaient correspondre à nos attentes.

**A** - La solution que nous avons apportée a été de trouver la bonne bibliothèque. En examinant l'ensemble des bibliothèques à notre disposition, nous avons pu identifier angular-spotify.js comme étant une bibliothèque répondant à nos besoins.

(2) PROBLEME : Ensuite, en programmant une première ébauche de l'application en javascript et angular JS, nous avons rencontré des problèmes liés à la superposition avec cordova.

**B** – Nous avons résolu ce problème notamment en nous documentant sur internet. Il a fallu revoir notre manière de programmer en angularJS et ainsi se concentrer d'abord sur angularJS puis incorporer notre code dans cordova.

(3) PROBLEME : Enfin, nous avons rencontré un problème lié au Token lors de la connexion Spotify. Accéder aux informations privées d'un compte Spotify nécessite des autorisations particulières. Pour obtenir ces autorisations, l'application doit récupérer un token. Nous n'arrivions pas à le récupérer.

# SPRINTS EN METHODE AGILE

**C** – Nous n'avons pas trouvé de solution à ce problème lors du sprint 1, nous avons donc réalisé notre première application en exploitant uniquement les données publiques des comptes Spotify utilisateurs (ces informations ne nécessitent pas de Token)

## Sprint 2 «Application»

(04/01/16 → 27/02/16)

A partir de janvier, nous avons donc commencé la création de l'application (Nous considérons la version précédente comme une simple ébauche). En réalisant nos premiers appels à L'API Spotify, nous nous sommes rapidement rendu compte que nous serions limités puisque certaines méthodes nécessitaient un Token (identification de connexion – voir Sprint1). Nous avons donc décidé de découper les tâches en deux. Mathieu se concentrait sur la récupération du Token (nous permettant de réaliser l'ensemble des appels nécessaires), tandis que Théo et Maxime précisaient le chaînage des méthodes d'appel afin de récupérer les genres préférés de l'utilisateur.

Ainsi, nous avons créé un algorithme (remanié à plusieurs reprises) afin de parcourir les Playlists de l'utilisateur (désormais bien identifié grâce à son Token) puis identifier chaque artiste. Il a ensuite fallu requêter Spotify pour connaître les genres de chaque artiste.

Cependant, une fois ce travail achevé, nous nous sommes rendus compte lors d'une phase de tests fonctionnels, que la bibliothèque de genres de Spotify était peu fournie. En effet, des artistes tels que Eminem, Rihanna ou encore Adèle, ne possédaient pas de genre ! Il était donc compliqué de fonctionner avec si peu d'information.

C'est pourquoi nous avons lancé un travail de recherche (qui n'était pas prévu à la base) afin de trouver une API plus complète renseignant les genres des artistes. A ce moment-là, nous avons saisi l'importance de travailler en méthode agile : Etre capable de faire évoluer ses besoins en respectant le planning défini. Nous avons donc découvert

# SPRINTS EN METHODE AGILE

EchoNest, qui fait le lien entre les données de Spotify et leur propre base de données de genres. Une fois cette modification terminée, nous avons de nouveau réalisé une batterie de tests fonctionnels qui se sont avérés justes (la quasi-totalité des genres était désormais renseignée). A noter que nous avons rencontré un problème lié aux nombres d'appels autorisés par cet API Echonest (120 par minute). C'est pourquoi, il a fallu repenser l'algorithme en utilisant des méthodes permettant de factoriser le nombre d'appels à l'API.

A noter que nous avons utilisé SonarQube afin de d'examiner notre code. SonarQube nous a permis d'avoir un retour visuel quant à la complexité des fonctions, la duplication du code, le temps estimé pour les corrections à apporter et les erreurs à corriger.

## SPRINT 2 : PROBLEMES ET SOLUTIONS

PROBLEMES RECONTRES	SOLUTION APPORTEE	SPRINT	DATE
(1)	A	2	15/01/16 au 25/01/16
(2)	B	2	28/01/16 au 05/02/16
(3)	C	2 (et 3)	20/01/16 au 25/02/16

(1) PROBLEME : Lorsque nous avons testé notre algorithme de récupération de genre (en demandant à l'API quel est le genre de chaque artiste présent dans la playlist utilisateur), nous nous sommes rendus compte que les artistes étaient très peu documentés, notamment au niveau de leurs genres.

A – Pour palier ce manque d'information concernant les genres des artistes, nous avons trouvé une autre bibliothèque (Echo Nest) bien plus fournie. En sondant de nouveau les bibliothèques Spotify des utilisateurs, nous avons eu des résultats bien meilleurs avec cette nouvelle API.

(2) En réalisant plusieurs tests fonctionnels, nous avons compris que lorsque la PlayList de l'utilisateur comprenait trop de morceaux, les appels réalisés (chaque morceau possède un attribut artiste que nous récupérons, puis nous interrogeons Echo Nest) dépassaient le quota autorisé. En effet, Echo Nest limite le nombre d'appels à leur API à 120/ minutes.

B – La solution que nous avons trouvée est algorithmique. Au lieu d'appeler Echo Nest pour chaque artiste à chaque musique sondée dans la playlist utilisateur, nous récupérons tous les artistes puis utilisons une autre méthode d'appel permettant de nous renvoyer la réponse globale une seule fois. En factorisant ce code, nous factorisons le nombre d'appel à l'API.

(3) Enfin, nous avons eu un dernier problème lors de ce sprint lié à l'asynchronisation du langage javascript. Nous étions habitués à travailler avec Java ou Php qui sont des langages synchrones. La manière d'appréhender les choses est totalement différente avec un langage asynchrone, c'est pourquoi il représentait une réelle difficulté pour nous. Ce problème est un problème récurrent que nous avons rencontré à plusieurs reprises, c'est pourquoi nous avons trouvé plusieurs manières d'y remédier.

C – Dans un premier temps, ce problème était bloquant. A tel point que nous ne pouvions continuer nos tests. C'est pourquoi la première solution que nous avons trouvée était d'utiliser la méthode « `window.setTimeout()` » qui nous permettait d'attendre que la tâche s'effectue avant de lancer une nouvelle opération. Cependant, cette façon de faire n'était que provisoire et ralentissait notre application. C'est pourquoi nous avons affiné nos connaissances des callbacks afin de découper les tâches et ainsi lutter contre nos problèmes asynchrone en se passant des `SetTimeout()`. Plus tard dans le projet, nous utilisons également les directive angular (`.then`) et la librairie `async` afin de lutter contre ce phénomène.

# SPRINTS EN METHODE AGILE

## Sprint 3 – partie 1 «Côté serveur»

(27/02/16 → 21/03/16)

Au cours du sprint 3, nos premiers efforts se sont concentrés sur les web service, nous faisons tourner notre client sur localhost :8000 et un serveur sur localhost :8080, nous essayions de faire transiter des informations du client au serveur. Il s'agissait là des premières ébauches de nos futurs Web Service.

Ensuite, nous avons mis en place une base de données via un script de création de table. Cette base de données contient tous les magasins qui sont compatibles avec WipeSound. Elle est composée des champs suivants :

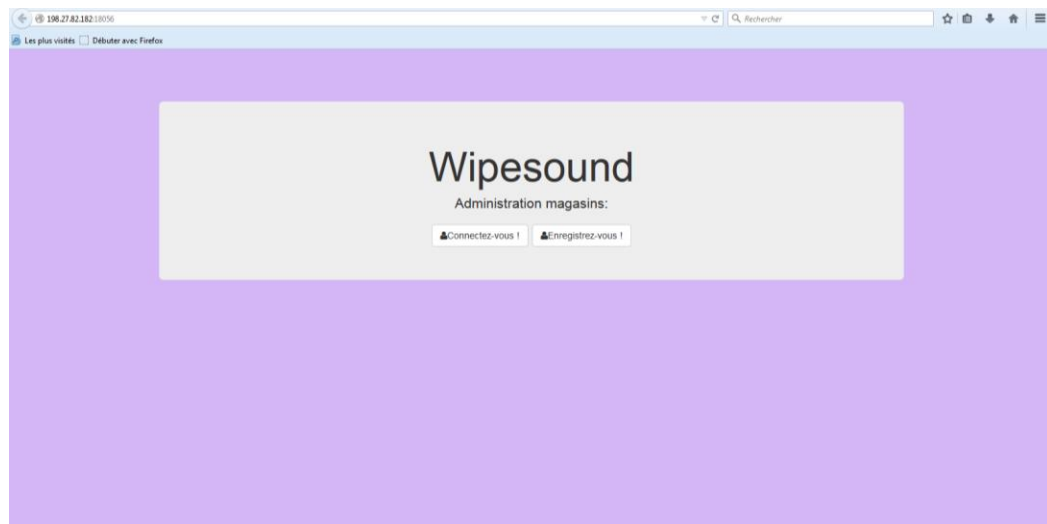
- Un Id qui s'auto incrémente, il s'agit d'une clef unique pour un magasin
- Un username
- Un password (crypté avec bcrypt)
- Sa position (latitude et longitude)
- Un booléen isLinkedSpotify (qui passe à « true » lors de la première connexion d'un magasin avec son compte)

id	username	password	lat	lon	isLinkedSpotify
1	superu	\$2a\$10\$NPDBYTrB87UkWwcjqA0bcexf20ZYpl2cFTftRZUct6LIthiKI/G3S	47.6448	-2.77784	1
2	Mathieu Home	\$2a\$10\$8FqQMCZbmmW6r58CP/ey..p9IOv2B8LBDuRZiMLDx8hwbT7oja9LC	47.6518	-2.78698	1
3	test	\$2a\$10\$mEuorXNqlzpmh.HX5ECj.dFeEH7wrtFD1pUWh/AIUz/sRRtVcYTW	48.1127	-2.34711	0
5	theo	\$2a\$10\$w9JsN7Ek.111LQF66a9h90Io38jFxCJJwF9g4CMVv9ErVxxw6l04K	48.1127	-2.34711	1

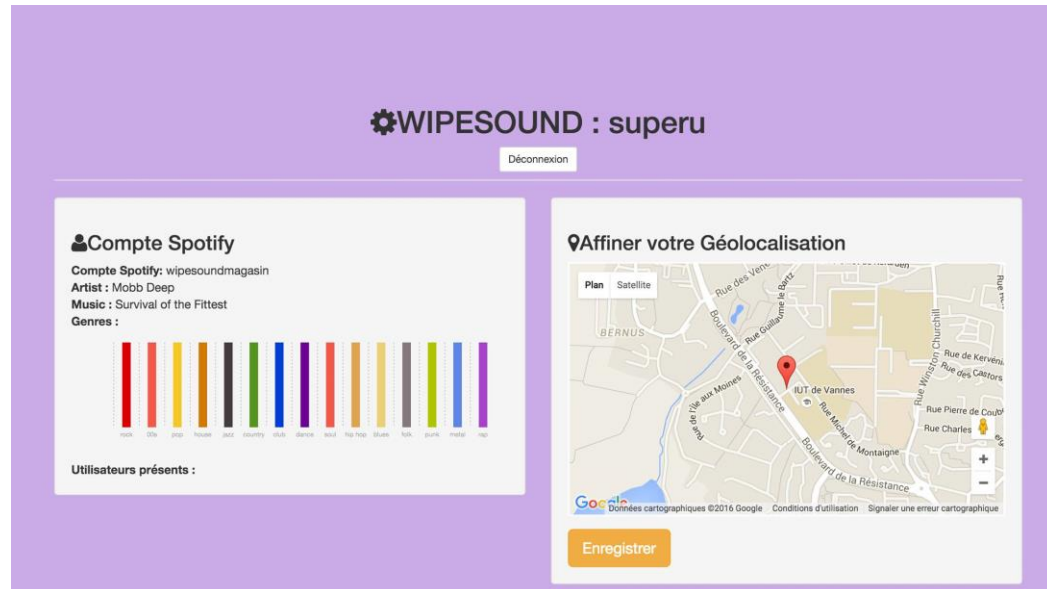
*La table « Places » qui contient les magasins compatibles*

# SPRINTS EN METHODE AGILE

D'autre part, nous avons créé une interface de connexion (avec Bootstrap) pour les magasins. Le magasin s'inscrit directement sur l'interface et peut accéder à la musique qui passe dans son magasin (voir ci-dessous).



*Connexion / Inscription Wipesound*



*Interface magasin*

# SPRINTS EN METHODE AGILE

Au cours de ce sprint, nous avons également créé un Web Service permettant au client d'obtenir la liste des magasins aux alentours (dans lequel il est susceptible de rentrer). Ainsi, nous réalisons un GET sur le serveur qui prend en paramètre la latitude et la longitude du client et qui renvoie une liste de magasins à proximité.

Nous avons créé des classes DAO, afin de rajouter une couche abstraite dans l'optique de faciliter et sécuriser la manipulation de nos tables en base de donnée.

D'autre part, un de nos rendez-vous avec notre maitre d'œuvre nous a permis de repenser un mécanisme de notre système. En effet, nous pensions que les clients enverraient constamment leur géolocalisation au serveur qui lui ferait le test pour savoir lesquels sont présents dans les magasins de la base de données. Cependant, nous avons décidé de fonctionner autrement. Grâce au web service expliqué plus haut, nous récupérons la liste des magasins côté client et c'est le client qui teste si il est présent dans l'un de ces magasins. L'envoi des genres musicaux du client au serveur ne se fait que lorsqu'il est géo localisé dans un des magasins.

Côté serveur, nous avons également besoin de connaître l'état des genres préférés de chaque magasin en temps réel. C'est pourquoi nous avons créé un tableau « Magasins » qui contient un tableau de genre pour chaque magasin. Lorsqu'aucun client n'est localisé dans ces magasins, tous les tableaux de genre sont à 0. Suite à cela, nous devons créer des web services permettant d'envoyer les genres des utilisateurs à ajouter au bon magasin (côté serveur) au moment où ils entrent dans le magasin, et d'envoyer les genres des utilisateurs à enlever au bon magasin (côté serveur) au moment où ils sortent du magasin. C'est pourquoi, nous avons créé « entremag » et « sortmag » qui répondent exactement à ces besoins.

# SPRINTS EN METHODE AGILE

## SPRINT 3 – PARTIE 1 : PROBLEMES ET SOLUTIONS

PROBLEMES RECONTRES	SOLUTION APPORTEE	SPRINT	DATE
(1)	A	3	02/03/16 au 08/03/16
(2)	B	3	06/03/16
(3)	C	3	18/03/16

(1) PROBLEME : Lorsque nous avons réalisé les premiers tests de géolocalisation avec l'application android, nous nous sommes rendus compte que la géolocalisation était imprécise.

A – Il a fallu utiliser au maximum le GPS, qui reste le moyen le plus sûr de localiser un appareil mobile. Nous avons essayé de forcer la localisation via le GPS, sans utiliser la localisation de la 3G ou du wifi. Suite à cela, nous avons obtenu des résultats bien plus satisfaisants.

(2) PROBLEME : Lorsque nous avons réalisé nos web service (notamment les premiers POST en Ajax) nous avons rencontré des problèmes lors de l'activation de ripple (ripple permet de simuler un téléphone depuis notre PC afin de simuler le rendu de notre application cordova).

B – La solution que nous avons trouvée afin de remédier à ce problème est de changer nos Post en ajax en directives http d'angularJS.

(3) PROBLEME : Enfin, nous avons des difficultés à calculer la distance entre deux points à la main, nos tests révélaient parfois des imprécisions.

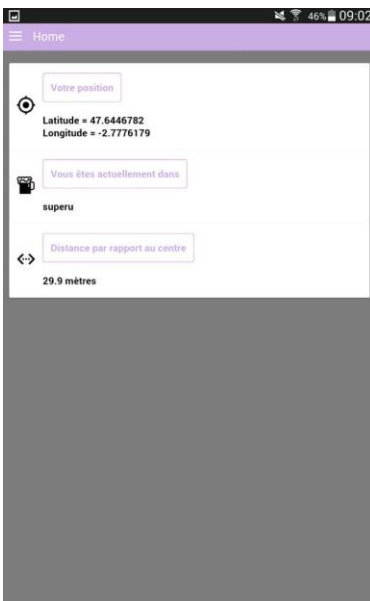
C – La solution a été trouvée en développant une méthode spécifique nommée `GetDistance()`.



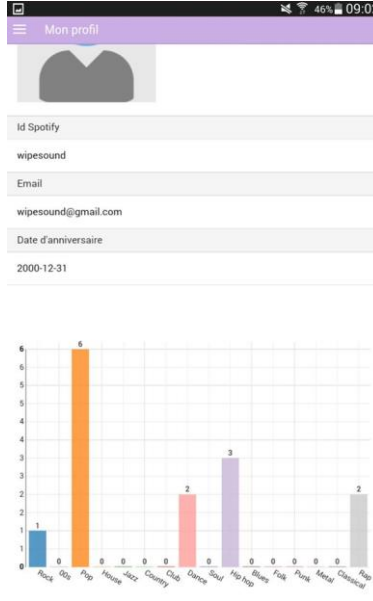
# SPRINTS EN METHODE AGILE

## Sprint 3 – partie 2 «Finalisation»

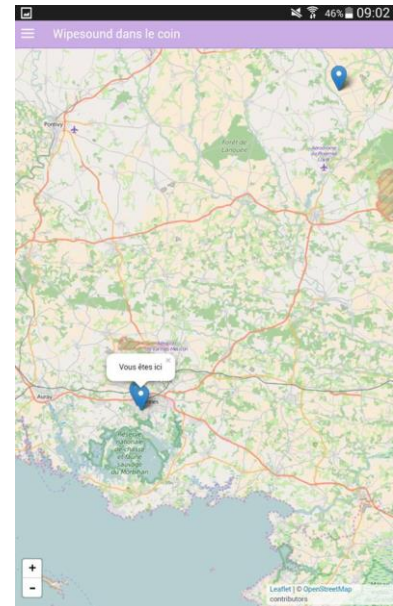
(16/03/16 → 28/03/16)



Home



Mon profil



Magasin dans le coin

Au cours de cette seconde partie de Sprint 3 destinée à la « finalisation » du projet, nous avons amélioré l'application mobile. En effet, on remarque ci-dessus qu'il est désormais possible via le menu « home » d'afficher sa position, le magasin courant et la distance séparant l'utilisateur du centre du magasin.

Via la rubrique « mon profil », il est possible d'accéder à un diagramme à bâtons, qui détermine la répartition des genres dans la playlist de l'utilisateur. On remarque facilement les genres qui prédominent, et nous pouvons analyser son top 3. Enfin, il est possible d'afficher une carte (construite avec Leaflet) des magasins dans le coin et de sa position actuelle.

Côté serveur, nous avons amélioré le CSS (conçu avec bootstrap), conçu des algorithmes permettant la transition musicale (en utilisant l'objet audio d'html 5 et des preview de musique Spotify) et mis en forme les genres à l'écoute dans un magasin via un diagramme à bâtons. Nous voulions retrouver la même charte graphique dans l'application et sur le serveur.

# SPRINTS EN METHODE AGILE

## SPRINT 3 – PARTIE 2 : PROBLEMES ET SOLUTIONS

PROBLEMES RECONTRES	SOLUTION APPORTEE	SPRINT	DATE
(1)	A	3 (partie 2)	25/03/16
(2)	B	3 (partie 2)	26/03/16
(3)	C	3 (partie 2)	27/03/16

(1) PROBLEME : Nous avons rencontré un problème concernant la transition musicale. Il s'agissait de trouver un moyen efficace de jouer la musique lorsque le magasin se connecte.

A - Pour répondre à ce besoin, nous avons utilisé l'objet audio d'html 5 au lieu du module Spotify spécifique. Nous utilisons les preview des musiques Spotify que l'objet HTML charge toute les trente secondes (Une fois la connexion établie).

(2) PROBLEME : Au cours de cette deuxième partie destinée à la finalisation, nous avons également été perturbés par un problème de routage, différent sur émulateur et sur tablette. Nous avons rencontré un problème de compatibilité, puisque cordova inséraient un « # » dans l'url de l'application émulée sur le navigateur. Or l'application Spotify précise qu'il ne faut pas de « # » dans l'URL pour pouvoir réceptionner le token de connexion. Il était donc impossible de récupérer le token de connexion.

B – La solution a été de récupérer le token via le serveur en réalisant un nouveau Web Service. Pour ce faire, on ouvre une page du serveur depuis l'application.

(3) PROBLEME : Lorsqu'un magasin s'enregistre sur le site wipesound, lors de sa première connexion à Spotify, nous voulions créer un algorithme automatique de création de playlist afin que chaque magasin possède les 16 playlist correspondant à chaque genre musical. Néanmoins, Spotify empêche le bon déroulement de cet algorithme puisqu'il renvoie des erreurs lors de la création automatique de PlayList. Nous sommes en présence d'un problème connu de Spotify qui n'a toujours pas été réglé.

# SPRINTS EN METHODE AGILE

C – Pour palier ce manque, nous n'avons pas trouvé de solution efficace si ce n'est créer manuellement les playlist manquantes.

## COMPARAISON : OBJECTIFS ET REALISATION

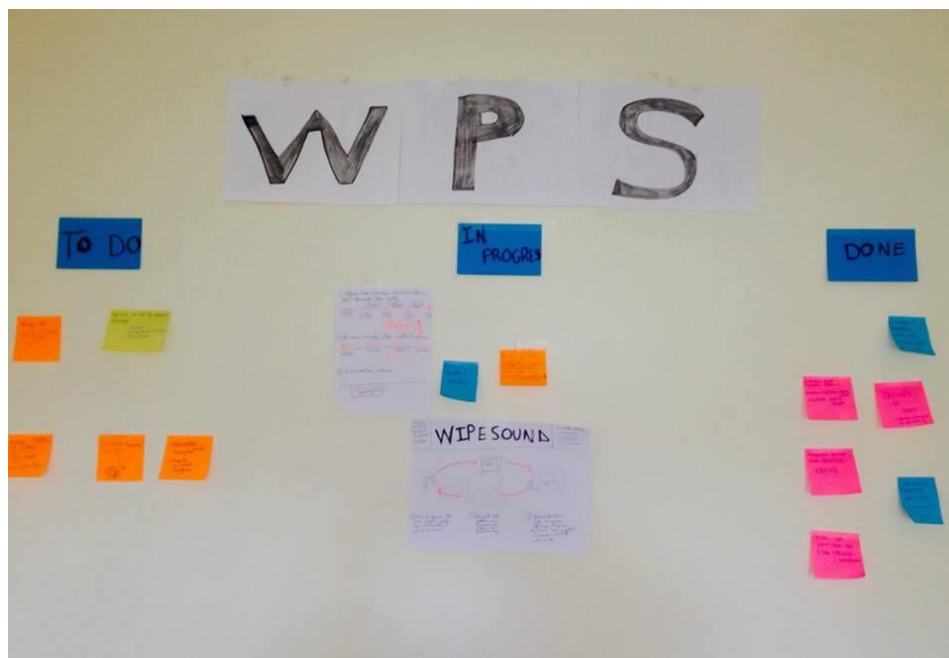
Si nous observons la matrice de complexité exposée précédemment on observe que nous avons en fin de compte réalisé la quasi-totalité des fonctionnalités que nous estimions obligatoires : la Tâche A) avec les fonctionnalités F1 et F2, la tâche B), la tâche C) avec F9) puis la tâche D) avec F10 et F11 (voir matrice de complexité ci-dessus). Nous avons donc réalisé toutes ces tâches et fonctionnalités qui sont citées, à l'exception de la fonctionnalité A) 3) qui était de « permettre aux utilisateurs de rentrer leur profil à la main ». En effet, nous avons voulu que notre système soit 100% automatisé, c'est pourquoi nous avons fait le choix d'obliger la connexion avec Spotify afin de sonder uniquement des bibliothèques Spotify sans permettre parallèlement d'indiquer manuellement ses genres préférés.

A noter également que nous avons réalisé certaines fonctionnalités que nous estimions optionnelles (en violet – cf. matrice de complexité). Ainsi, nous avons réalisé la fonctionnalité A) F7) (visualiser sur l'application les magasins autour de soi). Et également la tâche E) puisque nous avons développé une interface pour le lieu compatible qui peut établir plusieurs PlayList (une de chaque genre musical) pour son magasin (F12).

## Spécifications organisationnelles

### LA METHODE SCRUM

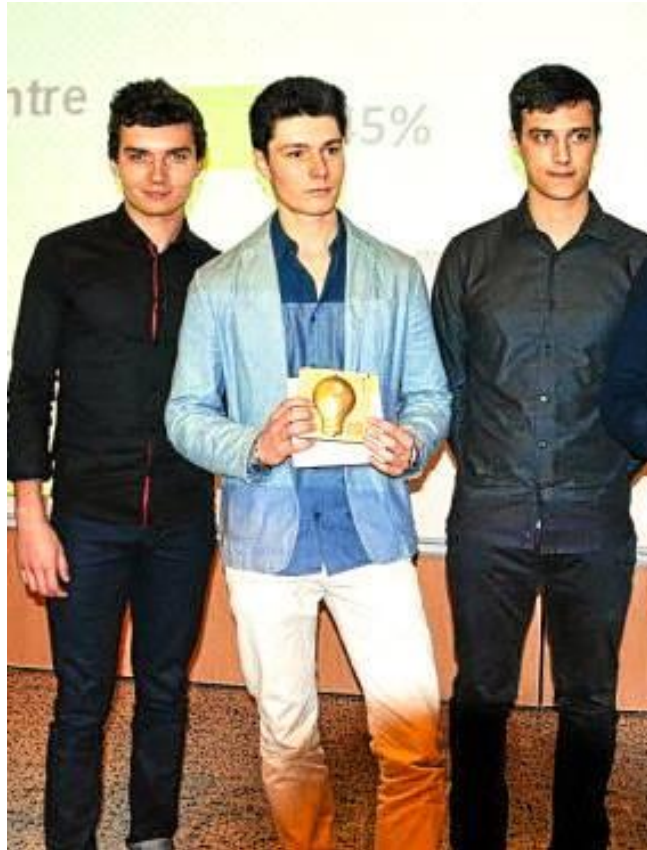
Afin de mener à bien notre projet, nous avons opté pour la méthode SCRUM. Cette méthode de projet agile permet non seulement un découpage des tâches précis mais également une certaine souplesse lors de repositionnements stratégiques au cours du projet. Ainsi, nous listions l'ensemble des tâches à effectuer sur un mur SCRUM installé chez Théo (chambre étudiant à côté de l'IUT) pour des raisons pratiques. Nous déplaçons les post-its en fonction de l'avancement de la tâche en question. Nous avons à plusieurs reprises, en concertation avec Monsieur Lelain, notre maître d'œuvre, changé d'idée quant à l'architecture à mettre en place. En effet, nous nous rendions compte qu'il existait des méthodes plus simples afin de réaliser des tâches identiques. Travailler en agile nous permettait de constamment redéfinir les tâches à effectuer lors des prochains sprints (back logs) et ainsi gagner en souplesse quant au déroulement du projet dans sa globalité.



SCRUM Task Board » Wipesound (WPS)

# SPECIFICATIONS ORGANISATIONNELLES

## L'EQUIPE



*L'équipe WipeSound lors de la « Foire aux neurones 2015 »*

DUMORTIER  
MAXIME

Chef de projet,  
responsable  
documentation.

SOYER  
MATHIEU

Développeur, responsable  
de communication.

LACROIX  
THEO

Développeur, responsable  
des tests.

# SPECIFICATIONS ORGANISATIONNELLES

## LES ROLES

Au cours du projet, nous avons eu chacun des rôles différents. Lors de la partie développement du projet, notre équipe a su se répartir efficacement les tâches. En effet, Mathieu s'est plus concentré sur la conception de l'application mobile, son design, le choix des technologies et son architecture. Théo a quant à lui, développé l'application côté serveur, et comme Mathieu, décidé du design et de son l'architecture. Maxime a fait office d'électron libre, aidant à l'élaboration de tous les algorithmes nécessaires au bon fonctionnement de chaque partie de l'application Wipesound, serveur comme mobile. Il a, en qualité de chef de projet, discuté avec chaque membre pour décider des voies à suivre et des choix de développement pour lesquels nous devions opter.

A noter que lorsqu'un membre de l'équipe avait besoin d'aide, les autres n'hésitaient pas à arrêter leur travail en cours pour apporter leur aide. Nous avons également dû travailler à deux sur des portions de code quand le niveau des algorithmes était plus élevé. Ainsi, nous avons construit certains algorithmes, comme celui qui teste si un utilisateur est présent un des magasins, que nous avons réalisé à trois.

## Spécifications des tests

### TESTS FONCTIONNELS

#### **Sprint 1:**

Au Sprint 1, nous avons réalisé des tests concernant la connexion avec Spotify, nous cherchions à obtenir des données comme le nom de l'utilisateur, en faisant apparaître ces informations en console.

#### **Sprint 2:**

Lors du Sprint 2, les vrais tests fonctionnels ont débutés. Pendant les phases de programmation, nous étions toujours à proximité de la console chrome ou Firefox et nous mettions en évidence le cheminement des algorithmes. Ainsi, nous pouvions voir tout ce qu'il se passait lors du clic sur « Connexion avec Spotify ». Nous avons ensuite testé le bon fonctionnement de la récupération des genres préférés de chaque utilisateur qui se connecte sur l'application. Les différentes étapes ont été :

- Test de la bonne récupération des playlists de chaque utilisateur via l'API Spotify
- Test des appels à l'API EchoNest pour la récupération des genres de chaque artiste non répertorié dans les données retournées par Spotify
- Test de notre algorithme qui calcul et renvoi les 3 genres les plus représentés dans les playlists des utilisateurs

# SPECIFICATIONS DES TESTS

## **Sprint 3 – partie 1) :**

Les tests fonctionnels du Sprint 3 ont davantage été axés sur la bonne transmission des données entre l'application mobile et notre serveur (grâce aux WebServices REST) et sur les tests fonctionnels sur la réactivité de l'application mobile et le serveur. Nous avons effectué les premiers tests sur la géolocalisation de l'application mobile. Exemple :

- Nous enregistrons le magasin dans la Bibliothèque Universitaire de l'IUT (nommé BU).
- Une tablette où est installé l'application Wipesound se connecte dans la BU pour vérifier qu'elle est bien prise en compte dans le magasin BU.
- La tablette sort de l'IUT pour se rendre au Jardin Universitaire. La tablette ne se trouve plus dans le magasin BU.
- La tablette se connecte au Jardin Universitaire. Elle ne se trouve pas dans le magasin BU.
- La tablette se trouvant au JU rentre dans l'IUT et se dirige vers la BU. La tablette se trouve dans la BU, les genres de l'utilisateur sont alors pris en compte.

## **Sprint 3 – partie 2):**

Tests plus poussés sur la géolocalisation pour déterminer avec précision dans quel rayon l'utilisateur ne se trouve plus dans un magasin. Tests d'ajout et de suppression d'un utilisateur dans le tableau des genres de chaque magasin, exemple : • Le magasin SuperU se connecte à notre application WEB puis se lie avec Spotify

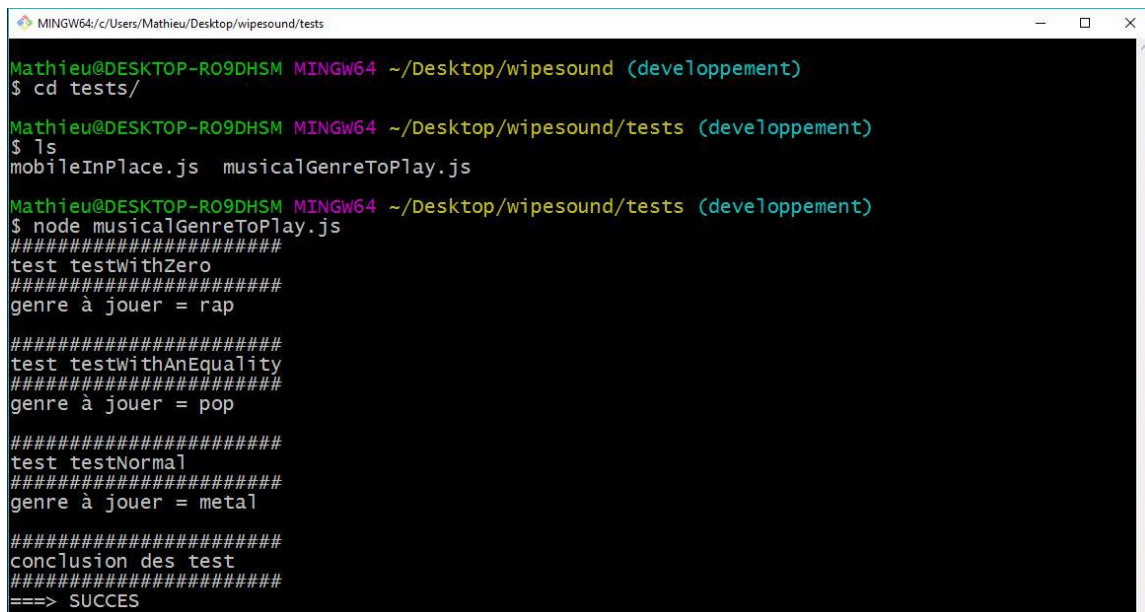
- Un premier utilisateur se trouvant dans ce magasin lance son application mobile et envoie ses données (le genre POP est son préféré, le ROCK est en second). La musique transite vers la playlist POP.
- Un second utilisateur se connecte toujours dans ce même magasin SuperU avec un genre préféré ROCK, la transition se fait vers la playlist ROCK du magasin.



# SPECIFICATIONS DES TESTS

## TESTS UNITAIRES

Nos tests unitaires ont été créés de façon à tester des portions de code, ou des méthodes en particulier. Ainsi, nous testions si notre algorithme de choix du genre prépondérant renvoyait le bon genre à partir d'un tableau de genre. Nous attendions par exemple avec une playlist contenant plus de chansons de rap, un retour « rap » pour que le test soit réussi.



```
MINGW64/c/Users/Mathieu/Desktop/wipesound/tests
Mathieu@DESKTOP-RO9DHSM MINGW64 ~/Desktop/wipesound (developpement)
$ cd tests/

Mathieu@DESKTOP-RO9DHSM MINGW64 ~/Desktop/wipesound/tests (developpement)
$ ls
mobileInPlace.js  musicalGenreToPlay.js

Mathieu@DESKTOP-RO9DHSM MINGW64 ~/Desktop/wipesound/tests (developpement)
$ node musicalGenreToPlay.js
#####
test testWithZero
#####
genre à jouer = rap

#####
test testWithAnEquality
#####
genre à jouer = pop

#####
test testNormal
#####
genre à jouer = metal

#####
conclusion des test
#####
==> SUCCES
```

## Evolutions futures

Au cours de notre projet, nous avons identifié de nombreuses évolutions futures. Ainsi, nous pourrions développer une compatibilité avec d'autre géant de la musique comme Deezer ou Sound Cloud. Pour améliorer notre système, nous pourrions également proposer de changer la musique manuellement côté administrateur dans le cas où la musique à l'écoute ne plait pas. Nous pourrions également à terme, changer les moyens de géolocalisation. En effet, le GPS n'est pas suffisamment précis pour délimiter deux bars voisins. C'est pourquoi nous pourrions utiliser les « Beacon », qui permettent de géo localiser via le Bluetooth Low Energy précisément des clients dans un magasin.

## Conclusion

Grâce au projet Wipesound, nous avons acquis de solides connaissances techniques. Nous nous sommes formés à NodeJS, AngularJS ou encore au Framework Ionic. Ces enseignements techniques nous ont permis de développer un système très proche de notre objectif initial. D'autre part, nous avons acquis un réel savoir-être en gestion de projet. Il a en effet fallu que chacun respecte son rôle et les tâches qui lui était donné. En travaillant en méthode agile, nous avons pu redéfinir avec plus de précision nos grands axes de travail.

Nous tenons aussi à remercier Monsieur Lelain, notre maitre d'œuvre et client, pour ses conseils précieux tout au long du projet. Nul doutes que cette étroite collaboration nous a permis de mener à bien le projet Wipesound.