

**Rapport du projet de LTW2**

***Share The Sound***



**DAITA Jordan** – M1 WIC

**DROUET Rémy** – M1 WIC

**LI Ling** – M2 DCISS RS

LTW2 – Enseignants : ATENCIA Manuel & ESPINOSA Javier

Année universitaire : 2014-2015

**Table des matières**

[**Introduction** 3](#_Toc417505571)

[**I.** **Conception de *Share the Sound*** 4](#_Toc417505572)

[**I.1. Description de l’application** 4](#_Toc417505573)

[**I.2. Architecture 3-tiers** 4](#_Toc417505574)

[**I.3. Synchronisation des chansons et système de votes** 6](#_Toc417505575)

[**I.4. Gestion des connexions** 6](#_Toc417505576)

[**II.** **Réalisation du prototype** 6](#_Toc417505577)

[**II.1. Technologies utilisées** 6](#_Toc417505578)

[**II.2. Principes ergonomiques** 8](#_Toc417505579)

[**III.** **Répartition du travail** 9](#_Toc417505580)

[**IV.** **Evolutions** 10](#_Toc417505581)

[**Conclusion** 11](#_Toc417505582)

# **Introduction**

Dans le cadre du cours LTW2, il nous a été demandé de réaliser un projet de groupe au sujet d’un *music player* collaboratif. C’est ainsi que nous nous sommes lancés dans cette aventure de la conception et le développement de l’application web *Share the Sound*.

Afin de mener à bien ce travail, nous nous sommes basés sur la description du projet fourni à son démarrage. Notre conception est issue de l’architecture 3-tiers, y compris le client, le serveur et la base de données. L’application doit être capable de fonctionner comme une radio. C’est-à-dire que la musique se lance automatiquement en fonction du nombre de votes pour chaque chanson (celle qui a le plus de votes sera la chanson suivante). En plus du système de votes, deux interfaces sont attendues : « *playlist »* et « *catalogue »*. C’est en nous basant de cela que nous avons pu émettre notre hypothèse et réaliser notre solution.

Nous allons présenter dans ce rapport les étapes de conception et de réalisation de notre travail pour aboutir à notre résultat. Pour ce faire, nous allons présenter le fonctionnement de notre *music player*, l’architecture de notre site web, comment nous avons émis l’hypothèse sur la synchronisation des chansons, avant de passer sur les aspects techniques et ergonomiques concernant la réalisation de notre site web.

# **Conception de *Share the Sound***

## **I.1. Description de l’application**

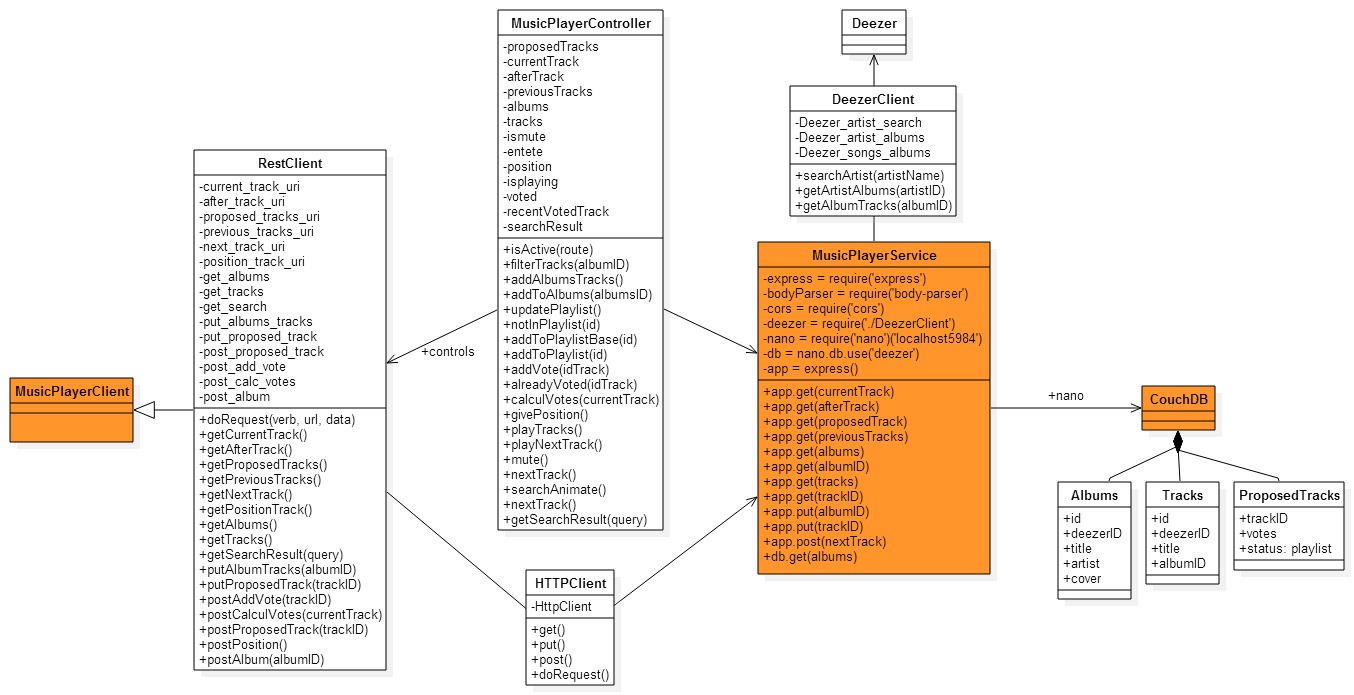
Notre application de *music player* consiste en une radio permettant aux utilisateurs d’écouter de la musique tout en participant au vote des chansons préférées. Les utilisateurs peuvent choisir des chansons dans le « catalogue », les ajouter dans la « playlist », voter les chansons dans la « playlist », écouter ou arrêter des musiques.

Fonctionnalités présentes :

* Possibilité de voter une fois pour les chansons proposées ;
* Possibilité d’ajouter une chanson dans la « playlist » présente dans le « catalogue » ;
* Possibilité de rechercher un album sur Deezer pour l’ajouter dans le « catalogue » ;
* Possibilité de rechercher des chansons dans la « playlist » ;
* Possibilité d’arrêter la radio et de couper le son ;
* Possibilité de reprendre la radio à l’endroit où se situe le serveur.

## **I.2. Architecture 3-tiers**

Notre application de *music-player* est représentée dans une architecture 3-tiers dont le client, le serveur et la base de données. Nous avons présenté cette architecture dans le diagramme suivant. Les trois grands acteurs sont : « *MusicPlayerClient* », « *MusicPlayerService* », « *CouchDB* ». Plus précisément, nous avons un « *RestClient* » qui est une instance de « *MusicPlayerClient* ». Il peut utiliser les services du style REST et il est contrôlé par « *MusicPlayerController* ». Nous pouvons nous apercevoir d’un MVC entre « *MusicPlayerController* », « *MusicPlayerService*» et « *RestClient* ». Et *« HTTPClient* » permet aux clients d’effectuer des requêtes *get/put/post/delete*. Le « *MusicPlayerService*», en tant que « *DeezerClient*», peut aller récupérer les artistes, les chansons et les albums dans Deezer. Grâce à *nano*, le « *MusicPlayerService »* peut se connecter à « *CoucheDB »* et récupérer un ensemble d’informations concernant les chansons et les albums. Notre base de données est constituée de trois grands documents : *albums, tracks, proposedTracks*.



## **I.3. Synchronisation des chansons et système de votes**

En arrivant sur la « *playlist* », le client va récupérer la position de la musique courante sur le serveur. Il va donc démarrer automatiquement la musique à cette position avec un intervalle d’erreur de cinq secondes. En effet, chaque client va envoyer sa position au serveur toutes les cinq secondes ainsi que sa musique courante pour vérifier qu’il a la même musique courante que le serveur. A la fin d’une musique, le client va prévenir le serveur qu’il a terminé. Si c’est le premier à avoir terminé cette musique, le serveur calculera la musique suivante selon les votes.

En ce qui concerne le système de votes, chaque client peut voter une seule fois pour chaque musique. Le serveur va mettre à jour le nombre de votes de chaque musique. Les clients peuvent ainsi récupérer le nombre de votes et mettre à jour l’affichage de la « *playlist* ». A chaque vote, le client est informé par un feed-back sous forme de bloc indiquant que le vote a été pris en compte.

## **I.4. Gestion des connexions**

Nous avons mis en place une simple gestion des utilisateurs non connectés. Si l’utilisateur n’est pas connecté sur Deezer, au bout de 30 secondes d’écoute, un *pop-up* va s’afficher pour lui demander de se connecter. Cependant, s’il se connecte, il sera obligé de rafraichir la page.

# **Réalisation du prototype**

## **II.1. Technologies utilisées**

Afin de rendre notre application de *music player* dynamique, interactive et collaborative, nous nous sommes servis des technologies et des langages web suivantes :

***Html5*, *css*, *bootstrap***

Les deux interfaces, « playlist » et « catalogue », ont été conçus grâce aux langages *html* et *css*, en plus du framework css *Bootstrap*. En effet, nous disposons d’une page d’index avec les codes scripts qui dirige les deux interfaces « playlist » et « catalogue ». L’utilisation du *Bootstrap* nous a permis de normaliser la présentation du site avec un ensemble homogène de styles. Il a également facilité le travail de la mise en forme des pages. C’est grâce à ce framework que nous avons mis en place des boutons esthétiques, la barre de navigation, etc.

***JavaScript, AngularJS, JQuery*** **et *Ajax***

L’utilisation du *JavaScript* est primordiale dans notre projet. Ce langage nous a permis d’écrire du code source en créant des fonctions et des évènements côté client afin de dynamiser nos pages *html*. Nous avons également utilisé un peu *jQuery* et *Ajax* qui servent à faciliter l’écriture des scripts.

Nous avons particulièrement employé le framework *JavaScript* *AngularJS*, *introduit dans le cours,* pour organiser efficacement les pages. Il nous a permis de construire une architecture MVC, et de manipuler le DOM au moyen de directives. Il a permis de créer une interface sous système d’onglets (*routeProvider*) et de faciliter la création de cette interface grâce à son système de *scope*.

***Node.js* et *nano***

Notre système est basé sur l’utilisation de *node.js* avec ses services notamment *express* et *nano*.

Nano est un outil qui nous a permis de faciliter le dialogue entre le serveur et la base de données (*CouchDB*). Il gère notamment la gestion des erreurs.

***JSon* et *CouchDB***

Une base de données est nécessaire dans notre projet pour stocker les musiques (*albums, tracks, proposed Tracks*). Nous avons choisi *CouchDB* qui a été introduit dans le cours. Il s’agit d’une nouvelle engeance de système de gestion de base de données. Nous voulions profiter de cette occasion pour découvrir cette base *NoSQL*. Pour ce faire, nous avons organisé une collection des données sous forme *Json*.

***Deezer API***

Tout au long du projet, nous nous sommes souvent rendus sur le site *Deezer developers* pour exploiter la présentation des données de musiques sur Deezer dans les onglets API et SDK. Etant donné que les musiques proposées dans le catalogue de notre application viennent de Deezer, nous devions pouvoir créer des fonctions pour récupérer et ajouter des chansons, mettre à jour la playlist, etc. Nous avons ainsi pu créer des requêtes d’actions *get/put/post/delete*.

***Github***

Nous nous sommes servis du service web *Github* pour héberger notre projet et partager les codes. Il nous a permis de travailler à distance et en collaboration. Une fois *git* installé, l’écriture des codes sont effectuée avec *WebStorm* qui demande juste quelques manipulations pour mettre à jour le projet.

## **II.2. Principes ergonomiques**

L’objectif de notre application est de créer une radio. C’est pour cette raison que nous tenons à développer un site ergonomique, dynamique et actif.

Dans un premier temps, nous avons créé un logo avec le nom de notre application « Share The Sound » et un écouteur qui rappelle le fait d’écouter de la musique sur ce site. Il a ensuite été intégré dans le *player*. Les couleurs que nous avons utilisées pour le logo sont réemployées dans l’interface de l’application afin de garantir la consistance de cette dernière.

En plus de la barre de navigation où un mini *player* se trouve en plus des boutons play-stop et volume, notre application possède deux interfaces : « playlist », « catalogue ».

**Interface « *catalogue* » :**

Nous avons eu l’idée de disposer des albums sous forme de tableau. C’est grâce à « *accordion »* de *bootstrap* que nous avons pu mettre en place ce tableau d’albums. Chaque album dispose d’une couverture, du nom d’album et du nom d’artiste. En cliquant sur le carré d’album, une liste des chansons s’affiche avec des boutons d’ajout si la chanson n’est pas présente dans la « *playlist* ».

De plus, une barre de recherche se situe au-dessus du tableau des albums. Lorsque le client entre un nom d’album ou de chanson ou d’artiste, le tableau des albums sera mis à jour avec un bouton de « Add to catalog » permettant au client de rajouter l’album s’il est absent dans le catalogue.

**Interface « *playlist* » :**

Nous avons prévu d’afficher les « *previous Tracks* » en haut du *player*. Son nombre est limité à trois et ils sont ainsi alignés. Dans le *player* se trouve la « *current track* », la « *after track* », un bouton *play-stop* et un bouton de volume. La liste des « *proposed Tracks* » est divisée en deux colonnes. Le nombre de votes est indiqué dans un rond pour chaque chanson. Lorsque le client vote pour une chanson, la couleur de « votes » change d’orange à vert.

Une louche permet de rechercher une chanson dans la *playlist*.

**Règles ergonomiques :**

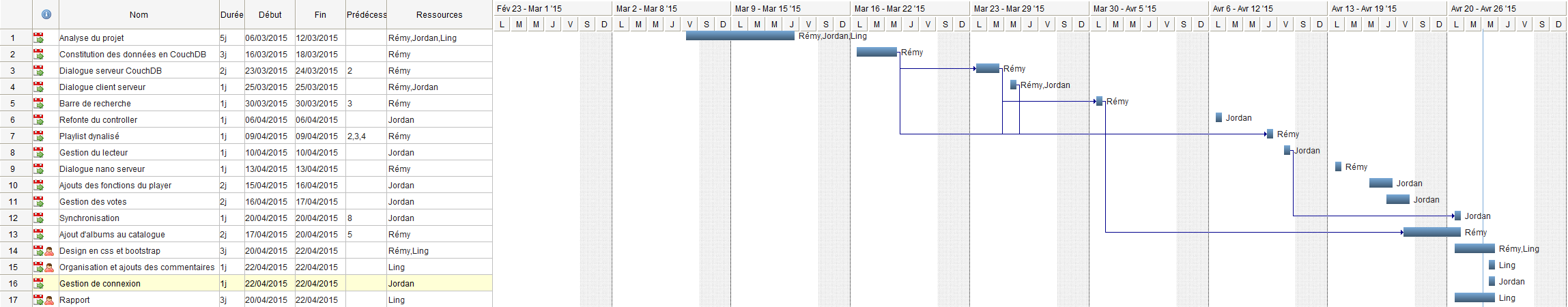
Nous tenons à créer une application qui s’attache à la consistance de l’IHM et reste facile à utiliser. Ainsi, nous avons défini des couleurs de base dans l’application. Les couleurs que nous avons privilégiées sont noir, gris, orange et blanc. La police que nous avons utilisée dans l’ensemble des pages est *Roboto (Google)*.

Au niveau des navigateurs, nous avons privilégié *Firefox* et *Chrome* lors de la réalisation de notre application.

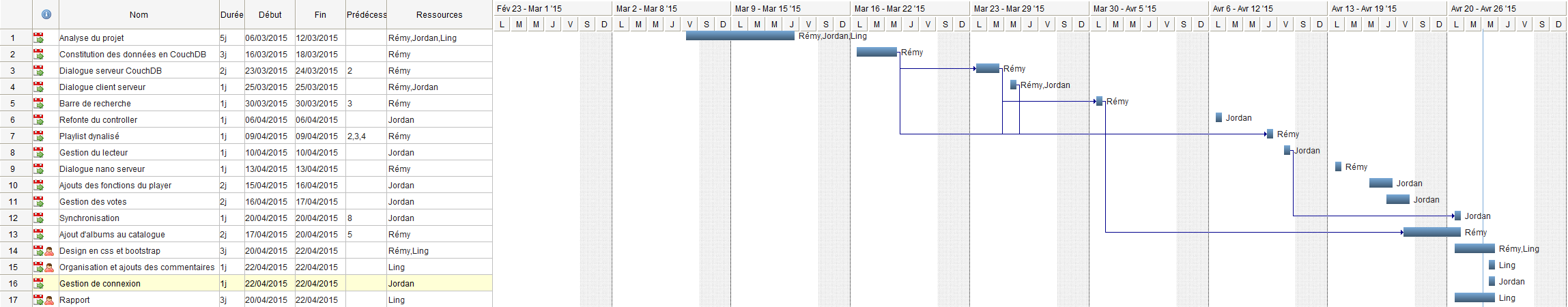
# **Répartition du travail**

Nous avons établi un planning de travail sur le diagramme de *gant*.

Voici la répartition du travail suivant :



Voici l’avancement du projet dans le temps :



# **Evolutions**

Notre projet ne s’arrête pas là. En effet, nous comptons continuer à développer cette application de *music* *player* et la sortir. Pour la future version de l’application, nous avons l’intention de fusionner la « *playlist* » et le « *catalogue*» en une seule interface. La fonction « recherche » permettra de chercher une chanson et/ou un album dans *Deezer*. Si la chanson n’est pas présente dans la « *playlist* », le client pourra la rajouter directement à ce niveau-là. De plus, nous développerons plus en profondeur sur l’ergonomie et les fonctionnalités de notre application.

Une version mobile est envisageable également.

# **Conclusion**

Ce projet nous a permis de mobiliser nos connaissances en langages web déjà acquises et découvrir des nouvelles technologies web telles que *CouchDB, JSON, REST, AngularJS*, etc. En plus de la manipulation des technologies web, nous avons été amenés à confronter les problèmes d’erreurs, à chercher des solutions et à gérer l’avancement du projet.

La conception et le développement du prototypage sont issus d’un travail d’équipe de trois personnes. La division des tâches a été importante, étant donné l’énorme volume de travail pour réaliser l’application.

Enfin faute de temps, il reste des fonctionnalités que nous souhaitions rajouter dans l’application. Nous espérons pouvoir continuer ce projet et concrétiser cette application.