
Rapport d'élève ingénieur
Projet de 2ème année
Filière : Génie Logiciel et Systèmes Informatiques

# Projet Fullstack de Révision de Parole

Présenté par : **BALLEJOS Lilian et CHARPIN Etienne**

**Responsable ISIMA : YON Loic**
**Soutenance : Jeudi 23 Mars**

**Projet de 60 heures**
**Année Scolaire 2022 - 2023**

Campus des Cézeaux. 1 rue de la Chebarde. TSA 60125. 63178 Aubière CEDEX

# Table des matières

# I Contexte du projet

## 1 Introduction au Full Stack

Pour commencer l'explication du projet, nous allons faire une brève introduction au **Full Stack**.

Lors d'un projet informatique de Développement Web, l'architecture du logiciel va être séparé en plusieurs parties distinctes : la partie côté serveur dit le "back" puis la partie côté utilisateur dite le "front".

La partie front va être chargé sur le moteur de recherche de l'utilisateur et donc le code sera disponible et exploitable par n'importe quel utilisateur du site web (par exemple le code HTML ainsi que le code Javascript).

A l'inverse la partie back va géré tous les comportements communs que l'on doit garder côté serveur (interagir avec la base de donnée, contacter une API ....).

Lorsqu'on parle de Full Stack, on parle donc du développement de ces deux aspects. D'où le terme Full Stack qui peut se comprendre comme "travailler tout les aspects du projet".

Dans ce projet nous allons vous montrer comment nous avons implémenter un projet Full Stack avec un back en NodeJs à l'aide du module "Express" et un front à l'aide du framework Angular.

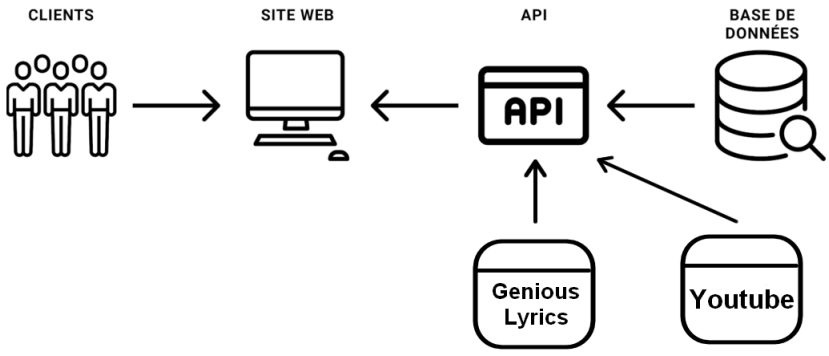


FIGURE 1 – Organigramme de notre Application Fullstack

# II Réalisation et conception

## 1 Nos choix

### 1.1 Le Back de notre site

Pour la partie back de notre projet nous avons choisit de prendre NodeJs avec le module "Express".

L'un de nous avait déjà travaillé sur cette technologie donc cela permettait pour l'un, de ne pas partir de zéro et d'améliorer ses connaissances et pour l'autre de découvrir une nouvelle technologie possiblement très utile pour notre filière de développement à l'ISIMA.

De plus, NodeJs est une technologie plus récente que d'autres langages qui peuvent gérer le côté serveur (comme PHP par exemple). Ainsi le module Express que nous utilisons est toujours maintenu à jour.

Avec Express, nous pouvons en quelques minutes implémenter une API assez facilement et parfaitement fonctionnelle. Cela nous a permit de bien scinder la partie front et back de notre projet. Nous pouvons par exemple citer l'exemple de PHP où la partie serveur et client est parfois mélanger dans les mêmes fichier.

Ici, nous avons le back dans un dossier et le front dans autre et ces deux aspects sont parfaitement indépendant ! On peut par exemple lancer le back pour faire des tests dessus sans toucher au front.

La partie front va pouvoir communiquer avec la partie front à l'aide de requête envoyé à celui-ci auquel il va répondre.

Au niveau de l'organisation de notre back, nous avons séparé celui-ci en différentes couches.

— **Couche "Controller"** : réceptionne les requêtes et appelles les fonctions nécessaires en fonction de la demande. Répond ensuite aux requêtes. Elle correspond au fichier *listen.js*

— **Couche "Business"** : effectue tous les calculs de notre API, on implémente dans celle-ci toutes les fonctions nécessaires au bon fonctionnement de l'API (communication avec d'autres API, création de trou à compléter dans les paroles etc...). Elle correspond à tous les fichiers *gestion_*.js* sauf "*gestion_database.js*"

— **Couche "Base de donnée"** : effectue toutes les requêtes à la base de donnée et renvoie les données récupérées. Elle est appelées par la couche business. Elle correspond au fichier *gestion_database.js*.

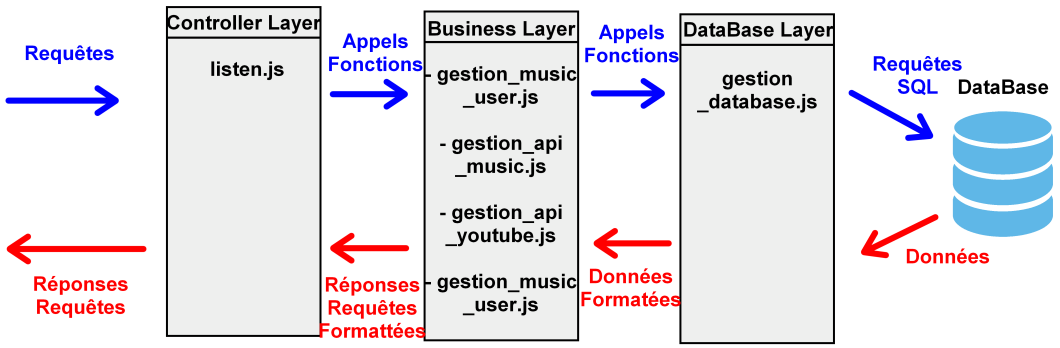


FIGURE 2 – L'organisation de l'API

Pour démarrer notre API, on lance le fichier **server.js** qui place notre API sur le port voulu de notre machine Ensuite, il se connecte à la base de donnée. Enfin, après que tout cela soit fait, on active les différents contrôleur de l'API et on est prêt à recevoir des requêtes !

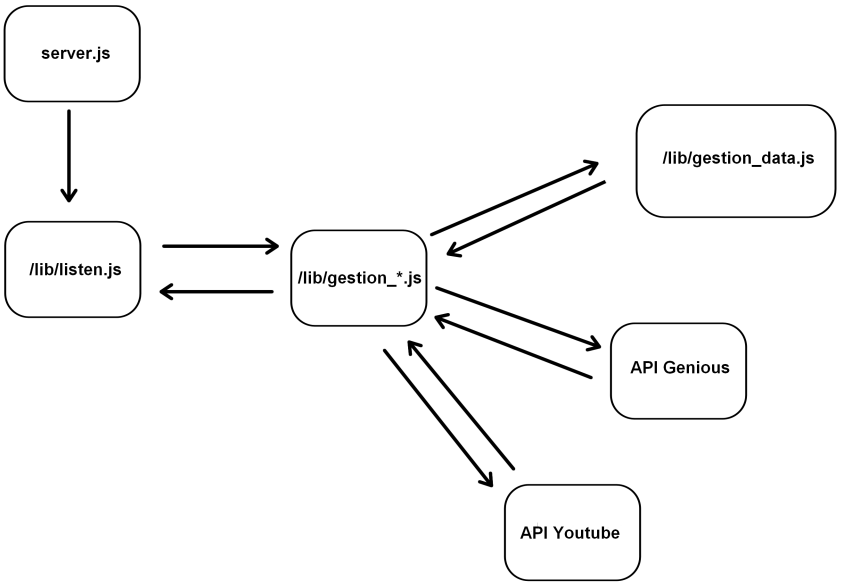


FIGURE 3 – Architecture de l'API

On peut voir ici que l'on communique avec d'autres APIs publiques ! On vous explique cela plus en détail après.

### 1.2 Le Front

Pour la partie Front nous avons prit le choix de prendre un Framework car ces technologies sont plus évolués pour effectuer des comportements complexes.

L'un des gros avantages de ces technologies est de pouvoir séparer notre application en plusieurs composants que l'on va pouvoir faire interagir les un avec les autres ou encore inclure les un dans les autres !
Cela permet de structurer plus proprement et efficacement une application.

De plus les Framework sont des outils très puissant avec de nombreuses fonctionnalités pré-implémentées Cela nous permet de ne pas avoir besoin de réinventer la roue est de nous concentrer sur nos implémentations personnelles.

Ainsi nous avons hésité entre le Framework Angular et ReactJS qui sont les plus courants et les plus utilisés de nos jours.
Notre choix c'est finalement porté sur Angular car nous étions familiarisés avec ce Framework suite à des projets personnelles. Le deuxième avantage de ce Framework est que sur la version que nous utilisons, il est nativement utilisé avec Typescript qui permet de faire des applications plus clair en typant les variables et objets utilisés. De plus, chaque composant généré est composé d'un fichier HTML, un fichier SCSS et un fichier Typescript ce qui permet de bien scinder les 3 différents aspects importants du front qui sont respectivement dans l'ordre le corps d'une page, son style et son comportement.

En plus d'Angular, nous avons utilisé le framework **Bootstrap** pour avoir des éléments graphiques classes et au goût du jour. Par exemple, notre barre de navigation a été stylisé grâce à ce framework.

De plus, nous avons utilisé **PrimeNg** qui est une bibliothèque de composants Angular. Nous y retrouvons dedans plein d'éléments déjà créés très pratique comme des dossiers (que nous avons utilisé pour la partie utilisateur) ou encore des boutons déjà stylisés.

### 1.3 Choix des APIs

Nous avons dû utilisé pour ce projet deux APIs publiques de deux entreprises différentes.

Ces APIs nous permettent de récupérer des informations comme par exemple des paroles de musique et nous ont été très utiles. Pour chacune d'elle nous avons dû nous inscrire sur le site de la documentation afin d'obtenir un token pour communiquer avec elles.

#### Gestion des lecteurs vidéos avec paroles de musique

La première est l'API de **Youtube** de l'entreprise **Google**. Nous lui demandons via une requête le meilleur résultat pour une recherche avec les mots clefs "*nom musique*" + "*nom artiste*" + "lyrics".

L'API nous répond en envoyant une liste des 5 vidéos sur sa plateforme qui corresponde le mieux à la recherche et nous, nous renvoyons le résultat numéro 1 à la partie utilisateur afin d'afficher le bon URL dans le lecteur.

#### Récupération des paroles de musique

Notre second objectif était de trouver une API qui pourrait nous fournir les paroles de musique et qui couvre donc un très grand nombre de celle-ci.

Nous nous sommes tout d'abord penché sur l'API **ChartLyrics** puis nous nous sommes rendu compte que celle-ci était trop incomplète. L'API étant américaine nous avions dessus très peu de musique française ce qui est très dommage au vu de notre projet.

Ainsi, nous nous sommes ensuite penché sur une autre API d'un site très connu qui couvre de très nombreuses musiques très diversifiées : **Genious Lyrics**.

Le problème de cette API est que dû à des problèmes de droit sur les paroles de musique (gérés en France par la **SACEM**), nous ne pouvons pas obtenir les paroles de musique directement via des requêtes.

La solution est la suivante : l'API nous fournit le lien vers la page web du site **Genious-Lyrics**. Nous récupérons ce lien et grâce à du Web scraping nous arrivons a récupérer les paroles.

Dans l'ordre la méthode pour récupérer les paroles d'une musique est donc la suivante :

— On contacte l'API Genious Lyrics et on lui demande l'url de la page du site qui contient les paroles
— On aspire ensuite tout le code HTML de la page web du site

— Les paroles étant toujours dans la même balise HTML, on les récupère tout en ignorant le reste du contenu de la page
— Enfin on formate les paroles pour qu'elle soit plus adapter à de l'apprentissage.

## 2 Le déroulement du projet

Notre façon de travailler était comparable à une méthode agile de développement.

C'est à dire que nous avons séparé le projet en différentes parties comme vous l'avez vu plus tôt (partie parole, partie user ...)

A chaque partie du projet, nous avons effectué un sprint durant lequel on suivait toujours le même schéma de travail :

— On implémente ou modifie la base de donnée
— On relie la partie serveur (back) du projet avec la base de donnée.
— On implémente les fonctionnalités nécessaires côté serveur
— On relie la partie user (front) à la partie serveur (back)
— On teste la fonctionnalité implémentée

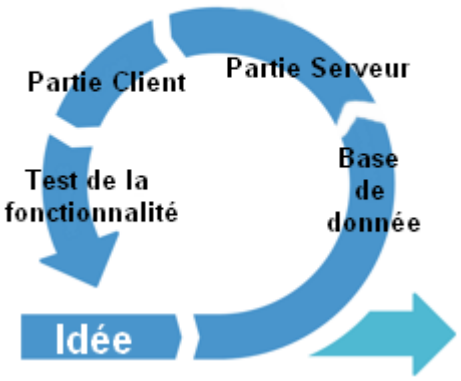


FIGURE 4 – Les différentes étapes d'ajout d'une fonctionnalité

Au final, on a effectué à la toutes fin un embellissement de la partie utilisateur (front) car il est très important d'avoir une interface utilisateur très agréable d'utilisation pour faciliter l'interaction avec notre application.

Voyons maintenant en précisions ce qu'impliquait chaque partie d'un sprint de notre projet.

### 2.1 Exemple complet d'un Sprint

#### Modification de la base de donnée

A chaque nouveau sprint on regardait tout d'abord si une modification de la base de donnée était nécessaire.

Si c'est le cas, on reconstruisait notre base de donnée en rajoutant les nouvelles tables et colonnes nécessaires.

Cela implique de modifier le script **setupDatabase.js** pour que nous ayons chacun de notre côté, en local, exactement la même base de donnée.

Lorsque cela était fait, on effectuait chacun de notre côté des tests en local sur nos machines afin de vérifier que nos bases de données avait le même comportement.

#### Mise en lien de la base de donnée avec notre partie serveur

Ensuite, on reliait la base de donnée à la partie serveur.

Le principe est le suivant : on créait des requêtes SQL qui permettent de récupérer les données dont on a besoin pour les futurs implémentations.

On inclut ensuite ces requêtes dans le code Javascript de notre back dans le fichier *gestion_database.js*

Enfin on crée dans la "couche business" toutes les fonctions nécessaires pour formater les données comme il nous faut.

C'est durant cette partie aussi que l'on a créé les fonctions qui envoie des requêtes HTTP à des APIs si nécessaire. Comme avec la base donnée, après avoir reçu les données on les formate afin de mieux les exploiter dans le front.

#### Mise en lien de la partie serveur avec la partie utilisateur

Au final, il ne nous restait plus qu'à créer côté client les nouveaux composants Angular nécessaires pour implémenter la nouvelle fonctionnalité. Ensuite, on avait juste à créer les requêtes à envoyer à notre partie serveur pour récupérer les éléments nécessaires côté client.

### 2.2 Nos différents Sprints effectués

Nous avons effectué ainsi 3 sprints durant le développement de notre application web.

— Le développement de la partie entrainement avec les paroles
— Le développement de la partie inscription/connexion
— Le développement de la partie partie stockage des musiques avec les dossiers utilisateurs

### 2.3 Embellissement du front

Au départ, nous avons cherché à développer une application fonctionnelle sans nous pencher de trop sur l'apparence. Nous avons essayé de développer toutes les fonctionnalités que nous voulions ajouter en nous assurant que celle-ci n'avait aucun bug.

Après s'être assuré que tous fonctionné comme nous le souhaitions, nous avons décidé à la toute fin d'embellir le front. Nous avons pour cela utilisé nos connaissances en CSS ainsi qu'en Bootstrap.

Ayant quelques notions d'IHM (interface homme machine), nous savons que l'apparence d'un site web est une étape primordiale du développement logiciel. En effet il ne faut pas par exemple utiliser des contrastes de couleur qui pourrait gêner certaines personnes comme les daltoniens.

De plus chaque éléments du site doit être intuitif et c'est ainsi que nous avons essayé de faire des fonctionnalités très simple d'utilisation (bouton simple, pas trop de détails ...).

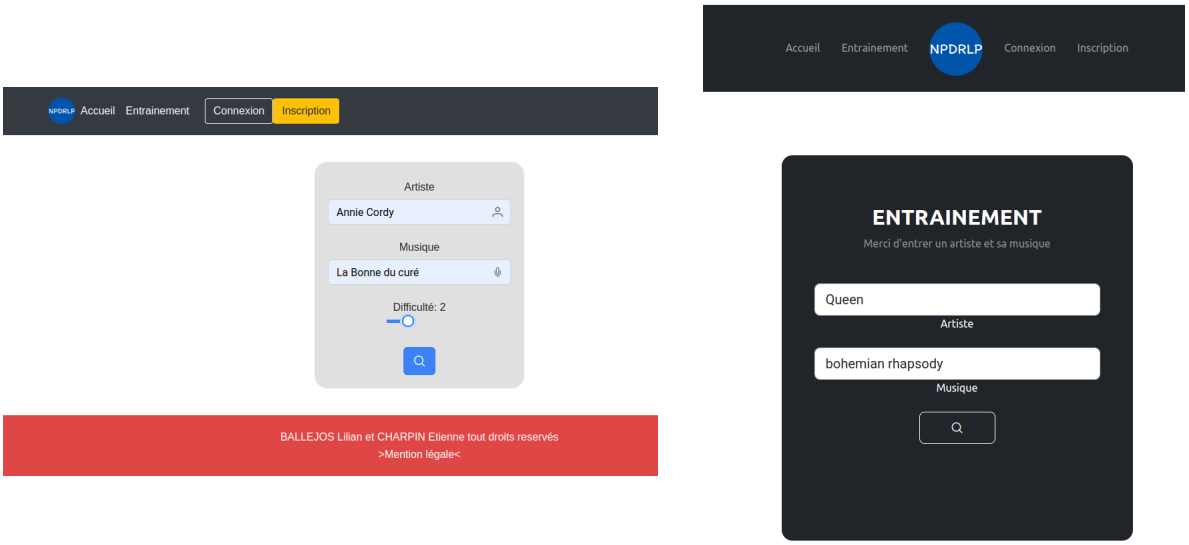


FIGURE 5 – Avant / Après de l'apparence de la partie client

# III Résultats et discussions

## 1 Prise en Main de l'Application Web

### 1.1 Arrivé sur le site

Lors de votre arrivée sur le site, vous arrivez sur une page d'accueil défilante qui vous informe du concept de notre application , de qui nous sommes et de nombreuses autres informations.

Nous pouvons si nous le souhaitons changer le code source du site web afin de rajouter des éléments à l'accueil et cela sans difficulté !

En haut, vous pouvez voir les différentes fonctionnalités de notre site web qui sont proposés dans le header.

On a donc la possibilité d'aller à l'accueil (là où nous sommes actuellement) mais aussi dans la partie entrainement. Nous pouvons aussi au choix nous inscrire où si cela est déjà fait de nous connecter !

Le header va changer en fonction de si vous êtes connecté ou non. Si vous êtes connecté, au lieu de la possibilité de s'inscrire ou se connecter, vous aurez la possibilité d'aller à votre menu utilisateur ou de vous déconnecter.

**IMAGE ACCUEIL**

## 1.2 Partie Entrainement

En arrivant dans la partie entrainement, un formulaire vous demande de renseigner à la fois une nom d'artiste ainsi que la musique que vous souhaitez apprendre.

Si vous renseignez quelques choses de faux ou erroné, l'application affichera un message d'erreur vous informant que les paroles n'ont pas pu être trouvé.

Si les informations renseignées sont justes, alors l'entièreté des paroles de la musique est affichée sur votre écran. En dessous de celle-ci vous trouverez un lecteur Youtube avec une vidéo de la musique recherchée.

Enfin nous trouvons au dessus des paroles un bouton "Options" sur lequel nous pouvons cliquer.

Lors du clique sur celui-ci, un menu apparait sur la gauche avec deux fonctionnalités :

— Sauvegarder la musique dans un dossier
— Changer la difficulté

### Sauvegarder la musique dans un dossier

Lorsque vous cliquez sur le bouton sauvegarder la musique, deux possibilités s'offre à vous. Si vous n'êtes pas connecté, nous allons vous informer qu'il faut que vous vous connectiez si vous souhaitez sauvegarder une musique. Sinon, une liste déroulante de tous vos dossiers apparait, il suffit de cliquer sur le dossier dans lequel vous voulez sauvegarder la musique pour lancer la phase de sauvegarde.

Après ce clique, une pop-up apparait vous demander de confirmer votre choix. Lorsque cela est fait une notification apparait en haut à droite de votre écran vous informant que la sauvegarde a été effectuée !

### Changer la difficulté

Un curseur vertical vous permet de changer la difficulté. Selon la difficulté choisit allant de 1 à 4, un certain nombre de trou vont apparaitre dans les paroles de la musique et le but de l'utilisateur est de les compléter. Un bouton "Valider" apparait aussi en bas des paroles afin d'effectuer une correction de vos réponses. Si la réponse est juste, la case se colorie en vert et vous ne pouvez plus modifier votre réponse, si elle est rouge c'est que c'est faux.

Vous pouvez voir au dessus des paroles un avancement du nombre de réponse juste par rapport au nombre de réponse attendu.

A noter que les trous sont disposés aléatoirement, si jamais vous modifiez la difficulté un nouveau pattern de trou sera appliqué aléatoirement.

## 1.3 Inscription