



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	2
1	Introduction au Full Stack	2
II	Réalisation et conception	2
1	Nos choix	2
1.1	Le Back de notre site	2
1.2	Le Front	4
1.3	Choix des APIs	5
2	Le déroulement du projet	6
2.1	Exemple complet d'un Sprint	6
2.2	Nos différents Sprints effectués	7
2.3	Embellissement du front	7
III	Résultats et discussions	8
1	Prise en Main de l'Application Web	8
1.1	Comment lancer l'Application Web	8
1.2	Arrivée sur le site	9
1.3	Partie Entraînement	9
1.4	Inscription	12
1.5	Connexion	13
1.6	Menu de l'utilisateur	13

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 séparée en plusieurs parties distinctes : la partie côté serveur dite le "back", puis la partie côté utilisateur dite le "front".

La partie front va être chargée 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).

À l'inverse, la partie back va gérer tous les comportements communs que l'on doit garder côté serveur (interagir avec la base de données, 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 tous les aspects du projet".

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

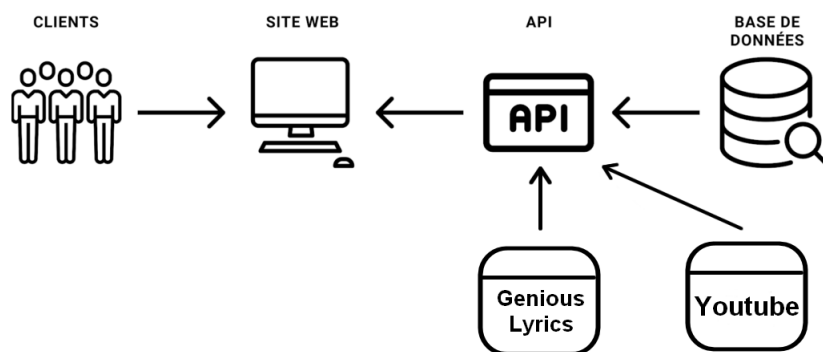


FIGURE 1 – Organigramme de notre Application Fullstack

II Réalisation et conception

1 Nos choix

1.1 Le Back de notre site

Pour la partie back de notre projet, nous avons choisi de prendre Node.js 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, Node.js 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 permis 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 mélangée dans les mêmes fichiers.

Ici, nous avons le back dans un dossier et le front dans un autre, et ces deux aspects sont parfaitement indépendants ! 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 back à l'aide de requêtes envoyées à celui-ci, auxquelles 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 appelle 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 appelée par la couche business. Elle correspond au fichier *gestion_database.js*.

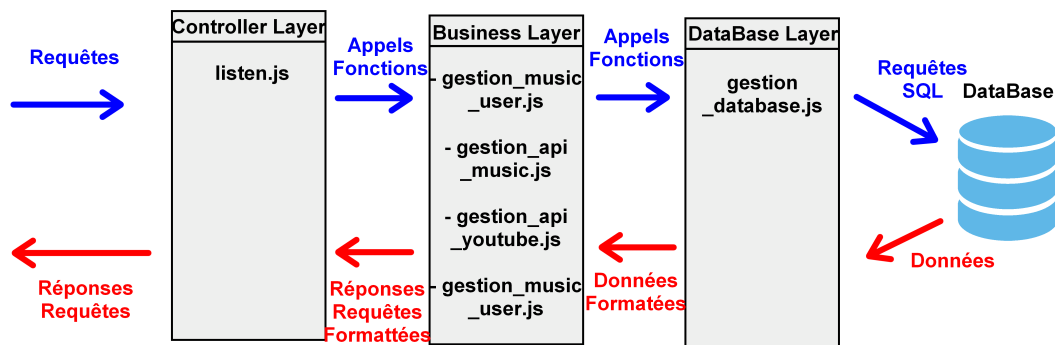


FIGURE 2 – L'organisation de l'API

Pour démarrer notre API, nous lançons le fichier **server.js** qui positionne notre API sur le port souhaité de notre machine. Ensuite, il se connecte à la base de données. Enfin, une fois que tout cela est effectué, nous activons les différents contrôleurs de l'API et nous sommes prêts à recevoir des requêtes !

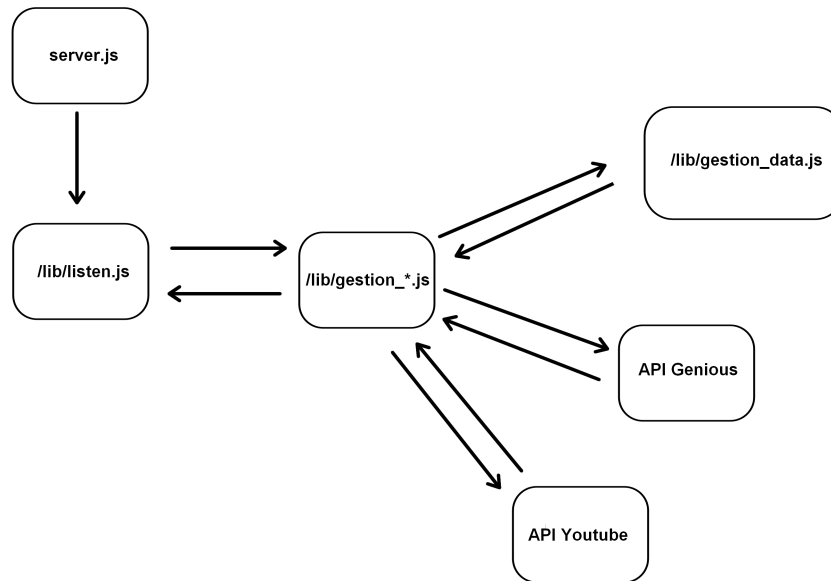


FIGURE 3 – Architecture de l'API

On peut voir ici que nous communiquons avec d'autres API publiques ! Nous vous expliquerons cela plus en détail par la suite.

1.2 Le Front

Pour la partie Front, nous avons choisi d'utiliser un Framework car ces technologies sont plus évoluées pour effectuer des comportements complexes.

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

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

Ainsi, nous avons hésité entre le Framework Angular et ReactJS qui sont les plus courants et les plus utilisés de nos jours. Notre choix s'est finalement porté sur Angular car nous étions familiers avec ce Framework grâce à des projets personnels. Angular est un Framework développé par Google qui permet de créer des applications web dynamiques et interactives. Il offre une grande flexibilité et une architecture bien structurée pour le développement de sites web complexes.

L'avantage de ce Framework est qu'il utilise nativement Typescript dans la version que nous utilisons, ce qui permet de créer des applications plus claires en typant les variables et les 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 séparer les trois 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 stylisée grâce à ce Framework.

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

1.3 Choix des APIs

Nous avons dû utiliser 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'elles, 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 clés "*nom musique*" + "*nom artiste*" + "lyrics".

L'API nous répond en envoyant une liste des 5 vidéos sur sa plateforme qui correspondent le mieux à la recherche et 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 d'entre elles.

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

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

Le problème de cette API est que, dû à des problèmes de droits 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 **Genius Lyrics**. Nous récupérons ce lien et, grâce au Web scraping, nous arrivons à récupérer les paroles.

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

- On contacte l'API Genius 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'elles soient plus adaptées 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 utilisateur, etc.). À chaque partie du projet, nous avons effectué un sprint durant lequel nous suivions toujours le même schéma de travail :

- Nous avons implémenté ou modifié la base de données.
- Nous avons relié la partie serveur (back) du projet à la base de données.
- Nous avons implémenté les fonctionnalités nécessaires côté serveur.
- Nous avons relié la partie utilisateur (front) à la partie serveur (back).
- Nous avons testé la fonctionnalité implémentée.

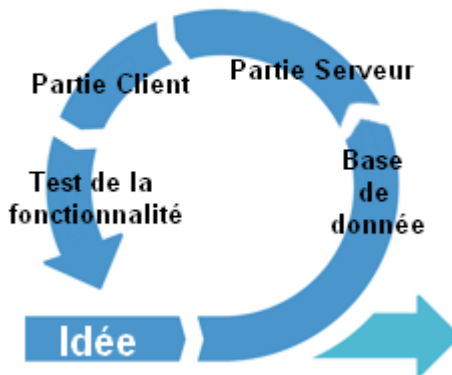


FIGURE 4 – Les différentes étapes d'ajout d'une fonctionnalité

Finalement, nous avons effectué à la 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.

Maintenant, voyons en détail ce que chaque partie d'un sprint impliquait.

2.1 Exemple complet d'un Sprint

Modification de la base de données

À chaque nouveau sprint, nous regardions tout d'abord si une modification de la base de données était nécessaire.

Si c'était le cas, nous reconstruisions notre base de données en ajoutant les nouvelles tables et colonnes nécessaires.

Cela impliquait de modifier le script `setupDatabase.js` pour que nous ayons chacun de notre côté, en local, exactement la même base de données.

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

Mise en lien de la base de données avec notre partie serveur

Ensuite, on reliait la base de données 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 futures implémentations.

On incluait ensuite ces requêtes dans le code JavaScript de notre back dans le fichier `gestion_database.js`.

Enfin, on créait 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 envoient des requêtes HTTP à des APIs si nécessaire. Comme avec la base de données, après avoir reçu les données, on les formatées afin de mieux les exploiter dans le front.

Mise en lien de la partie serveur avec la partie utilisateur

La page d'accueil de notre site web 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 entraînement avec les paroles
- Le développement de la partie inscription/connexion
- Le développement de la 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 celles-ci ne présentaient aucun bug.

Après nous être assurés que tout fonctionnait 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 utiliser, par exemple, des contrastes de couleur qui pourraient gêner certaines personnes comme les daltoniens.

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

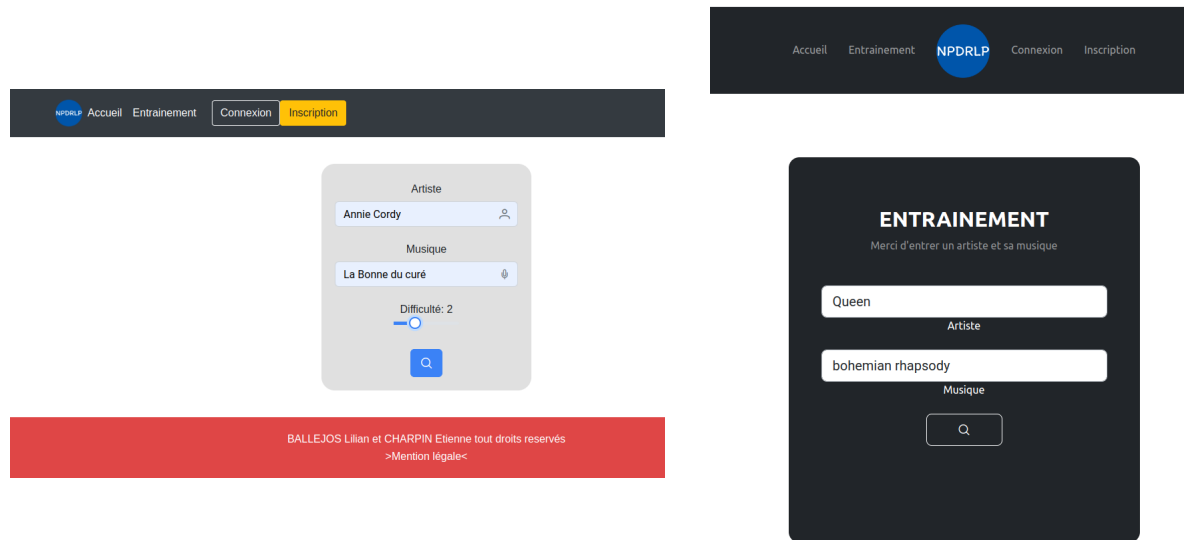


FIGURE 5 – Avant / Après de l'apparence de la partie client

III Résultats et discussions

1 Prise en Main de l'Application Web

1.1 Comment lancer l'Application Web

Nous n'avons pas exporté notre projet car il est encore en phase de développement. Pour lancer notre application sur une machine, il faut suivre quelques étapes.

Installation et configuration (à faire une fois) Tout d'abord, il vous faudra installer quelques logiciels :

- **mysql** : Pour la gestion de la base de données.
- **node** ainsi que son package manager **npm** : Pour lancer la partie serveur et la partie utilisateur.

Ensuite, vous pouvez trouver notre projet sur GitHub à ce lien :

<https://github.com/LilianHub/projetZZ2-NPDRLP>

Après avoir installé le projet sur votre machine, il va falloir aller dans les dossiers "*backend-api*", "*frontend-angular*" et "*setupDataBase*" et exécuter **dans ces 3 dossiers** la commande **npm i**, qui va permettre d'installer les dépendances nécessaires.

Une fois cela fait, lancez **mysql** et créez une nouvelle base de données nommée **npdrp** en entrant la commande **CREATE DATABASE npdrp**.

Ensuite, rendez-vous dans le dossier *setupDataBase* et exécutez **node setup.js** afin de lancer la configuration de la base de données.

Lancer l'application Pour lancer l'application et la tester, il vous suffit de suivre ces deux étapes :

- Dans *backend-api*, entrez la commande **ng serve**.

— Dans *frontend-angular*, entrez la commande **node server.js**.

Une fois cela fait, vous pouvez accéder à notre projet en local à cette adresse :

http://localhost:4200

1.2 Arrivée sur le site

En arrivant 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, cela sans difficulté !

En haut de la page, vous pouvez voir les différentes fonctionnalités de notre site web proposées dans le header. Vous avez la possibilité d'aller à l'accueil (là où nous sommes actuellement) mais aussi dans la partie entraînement. Vous pouvez également vous inscrire ou, si cela est déjà fait, vous connecter !

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

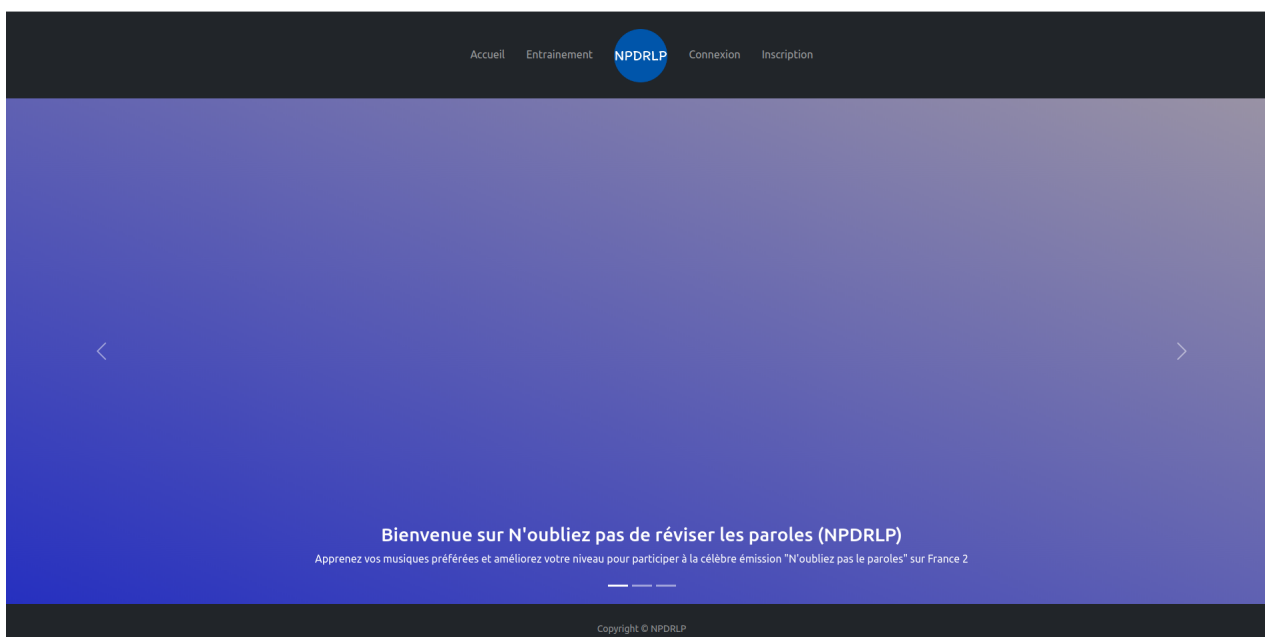


FIGURE 6 – La page d'accueil de notre site web.

1.3 Partie Entraînement

En arrivant dans la partie entraînement, un formulaire vous demande de renseigner à la fois le nom de l'artiste ainsi que la musique que vous souhaitez apprendre.

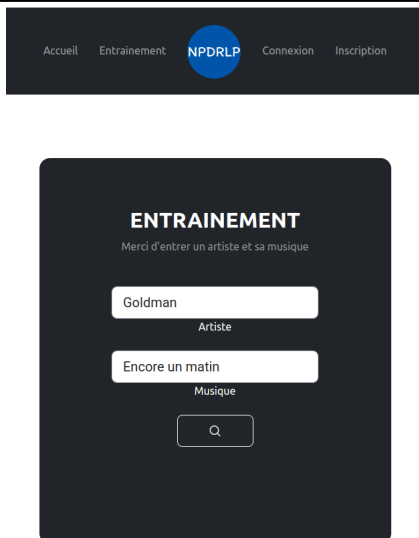


FIGURE 7 – Formulaire de recherche de paroles

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

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



FIGURE 8 – Affichage des paroles dans le mode entraînement

Enfin, nous trouvons au-dessus des paroles un bouton "Options" sur lequel nous pouvons cliquer. Lors du clic sur celui-ci, un menu apparaît sur la gauche avec deux fonctionnalités :

- Sauvegarder la musique dans un dossier
- Changer la difficulté

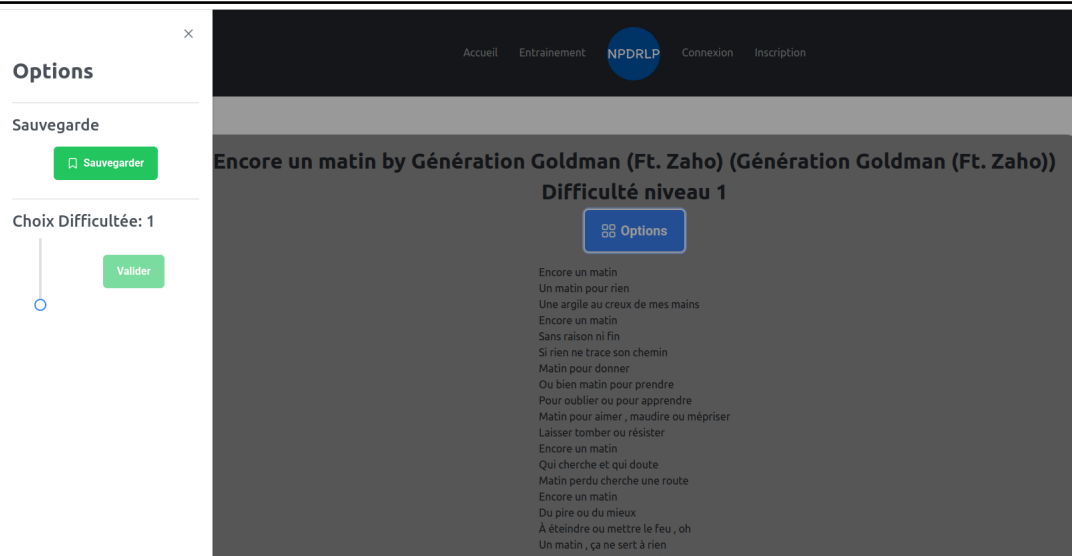


FIGURE 9 – Options du mode entraînement

Sauvegarder la musique dans un dossier

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

Après ce clic, une pop-up apparaît vous demandant de confirmer votre choix. Lorsque cela est fait, une notification apparaît 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 choisie, allant de 1 à 4, un certain nombre de trous vont apparaître dans les paroles de la musique et le but de l'utilisateur est de les compléter. Un bouton "Valider" apparaît aussi en bas des paroles afin d'effectuer une correction de vos réponses. Si la réponse est juste, la case se colore 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 réponses justes par rapport au nombre de réponses attendues.

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



FIGURE 10 – Paroles avec des trous

1.4 Inscription

Lors de votre arrivée sur la page d'inscription, un formulaire est à remplir avec un nom d'utilisateur et une double vérification de mot de passe.

- Lorsque l'utilisateur appuie sur le bouton "S'inscrire", plusieurs éléments sont vérifiés :
- Nous vérifions déjà côté client que les deux mots de passe entrés sont identiques, sinon nous mettons un message d'erreur
 - Ensuite, nous envoyons les informations au serveur. Deux retours sont alors possibles :
 - Le nom d'utilisateur est déjà utilisé : on met un message d'erreur
 - Nous avons bien enregistré le nouvel utilisateur

Si l'inscription a réussi, un message nous l'informe et nous sommes automatiquement redirigés vers la page de connexion après 5 secondes.

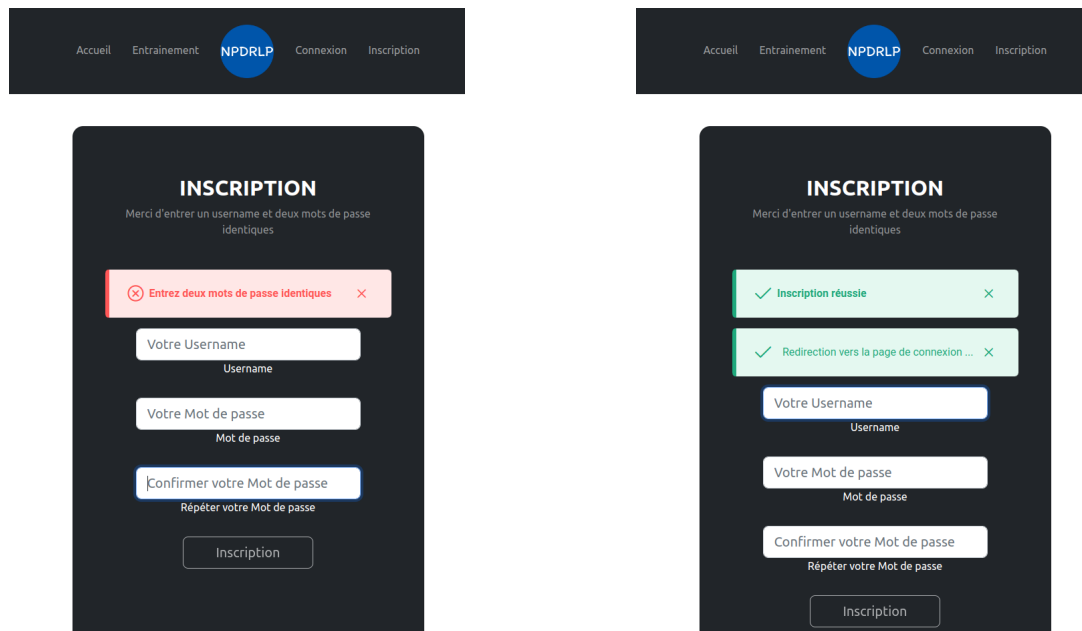


FIGURE 11 – Page d'inscription

1.5 Connexion

Nous avons ensuite la page de connexion. C'est ici que l'utilisateur, après s'être inscrit, va pouvoir débloquent l'accès à son menu personnel.

On demande de renseigner un nom d'utilisateur et le mot de passe qui va de pair avec.

Plusieurs cas d'erreur peuvent alors survenir :

- Le nom d'utilisateur n'existe pas
- Le nom d'utilisateur existe mais le mot de passe est erroné

Dans tous les cas, si une erreur a lieu, on empêche la connexion de l'utilisateur et on l'informe de l'erreur qui est survenue.

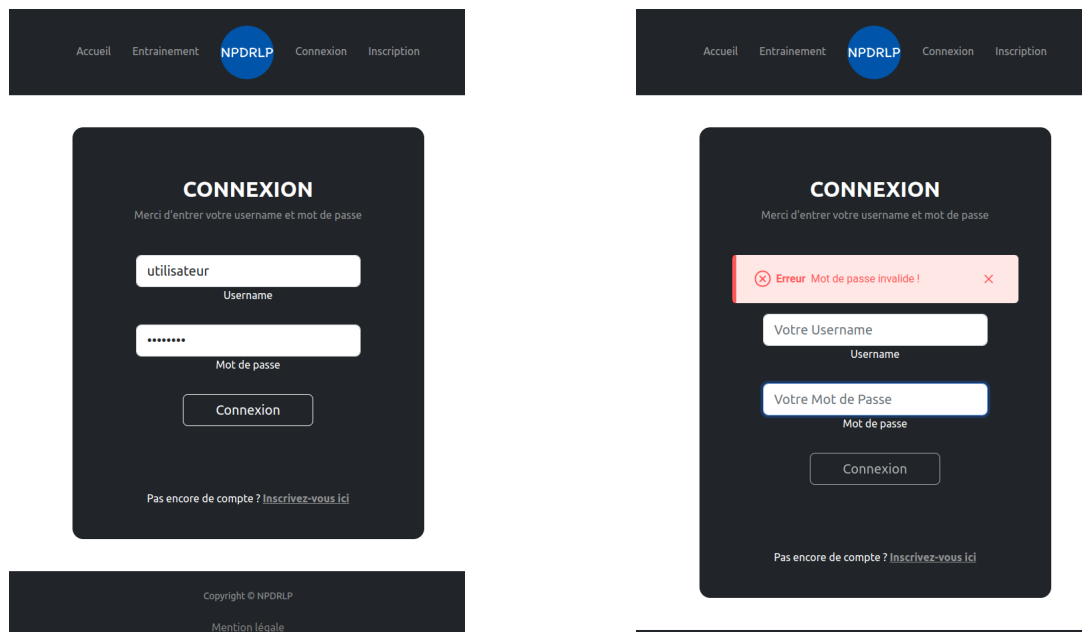


FIGURE 12 – Page de connexion

Au contraire, si la connexion est réussie, on redirige automatiquement l'utilisateur vers sa page de menu. Cela arrivera quoi qu'il arrive : si jamais un utilisateur connecté essaye de retourner sur la page de connexion, nous le redirigeons directement vers son menu car il est déjà connecté !

1.6 Menu de l'utilisateur

Enfin, nous avons le menu de l'utilisateur.

Nous avons placé dedans un système d'édition de dossier ainsi qu'une explication brève et simple de comment fonctionne le site.

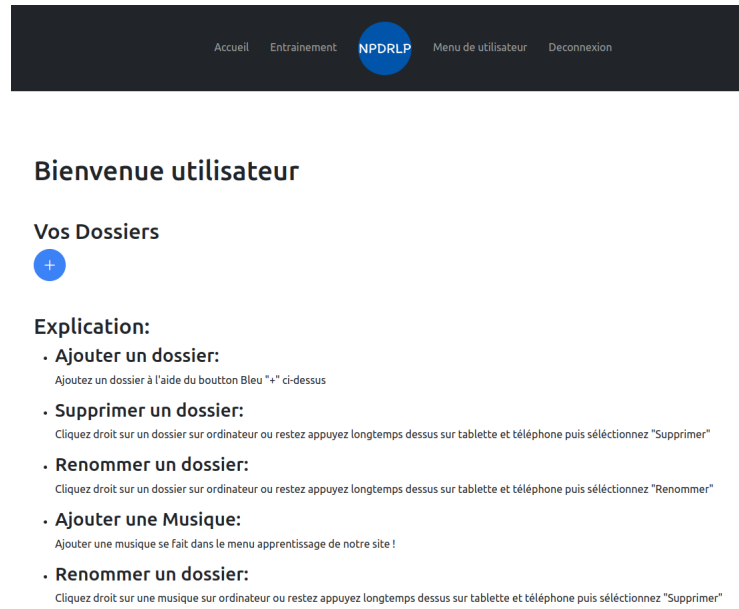


FIGURE 13 – Le menu utilisateur

Le système d'édition de dossier

À la première connexion, vous trouverez seulement un bouton avec le symbole "+". Ce bouton vous permet de créer un dossier avec le nom que vous voulez.



FIGURE 14 – Création d'un dossier

Vous avez ensuite la possibilité d'ajouter une musique dans un dossier comme expliqué plus tôt dans le mode entraînement.

Vous avez enfin la possibilité d'éditer vos dossiers. En faisant sur votre ordinateur un clic droit (ou un clic long sur votre téléphone et tablette) sur un dossier, vous pouvez soit le supprimer lui et toutes les musiques qui le composent, soit le renommer.

Si vous faites un clic droit sur une musique, vous pouvez la supprimer.

Avant chaque suppression d'un élément, nous demandons une confirmation à l'utilisateur car cette action est irrévocable. Une notification vous informe si votre action a bien été prise en compte ou non en haut à droite de votre écran.

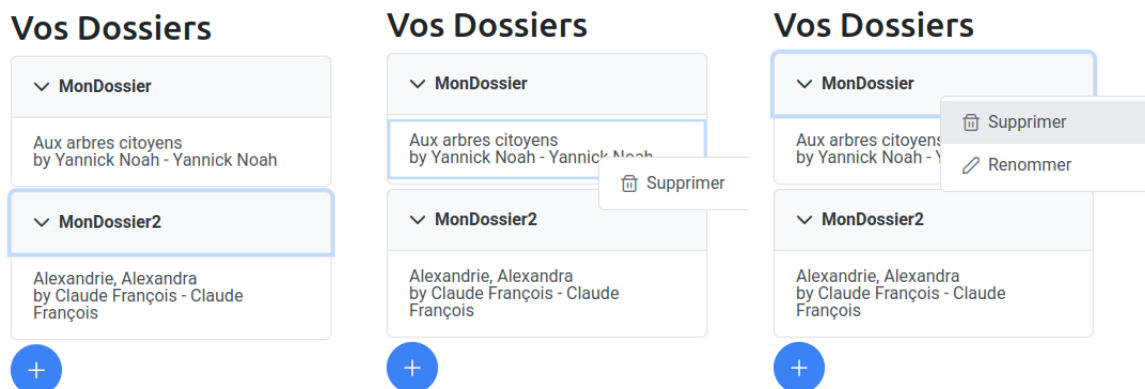


FIGURE 15 – Édition des dossiers et musiques