

Projet dans le cadre du cours NTIC

FlashCard vocal

—

Rapport



UNIVERSITÉ
DE GENÈVE

Auteurs:

Sara Cousin

Manish Kumar

Étudiants à l'université de Genève

En 3ème année du Bachelor en Système d'Information et Science des Services.

Encadrants Académiques :

Luka Nerima

Haozhou Wang

Client :

Jean-Philippe Goldman

| | |
|--|-----------|
| Introduction | 2 |
| Analyse | 2 |
| Besoins | 2 |
| Contraintes techniques | 3 |
| Use case | 4 |
| Déroulement du projet | 6 |
| Comparatif des APIs | 7 |
| Prototype | 7 |
| Plan des interactions de notre prototype | 9 |
| Détails des écrans | 10 |
| Architecture | 11 |
| Schéma d'architecture | 11 |
| Schéma base de données | 13 |
| Moodle (structure du plugin) | 13 |
| APIs | 16 |
| Traitement Audio et Scoring | 16 |
| Résultat | 17 |
| Prochaines étapes & Recommandations | 22 |
| Conclusion | 23 |
| Difficultés rencontrées | 23 |
| Notre évolution | 24 |
| Le mot de la fin | 25 |

1. Introduction

En 1970, le concept de FlashCard a vu le jour, ce dispositif d'apprentissage est fondé sur la technique de répétition espacée. L'apprenant.e dispose de cartes qui comportent sur une face la question et sur l'autre la réponse. Il est alors possible de s'auto-évaluer et d'apprendre par soi même grâce à ce système.

Ce dispositif a cependant quelques limites pour certains types d'apprentissage. Dans le cadre de ce projet, nous nous concentrerons sur l'apprentissage des langues, dont la limite évidente avec des cartes physiques est qu'aucun retour n'est fait sur la prononciation et la justesse des accents tonique au sein des mots.

Le but de ce projet est de créer un dispositif de FlashCard vocal qui serait capable de reconnaître si l'apprenant.e a correctement découpé les syllabes, mis l'accent au bon endroit du mot et bien évidemment s'il a dit le bon mot. Le procédé suivra donc l'alignement forcé qui consiste à écouter la prononciation de l'apprenant.e et de la comparer à la prononciation correcte de l'enregistrement du mot. Notre projet ne traitera que de mots isolés, aucune phrase ne sera traitée.

2. Analyse

Dans cette section nous parlerons des besoins fonctionnels et non-fonctionnels déjà discutés, ainsi que les contraintes techniques exigées. Une phase d'analyse des besoins et exigences du client sera nécessaire pour compléter cette partie.

2.1. Besoins

Fonctionnel

- Le système doit afficher des images.
- L'utilisateur peut enregistrer sa voix.
- Le système donne un retour à l'utilisateur basé sur son enregistrement et lui fournit un score.
- L'utilisatrice peut demander au système de lui faire écouter réponse
- L'utilisateur doit pouvoir passer d'une FlashCard à l'autre
- Il est possible pour le.s propriétaire.s d'ajouter de nouvelles images et de nouveaux sons au système.

Non- Fonctionnel

- Le système reconnaît la place de l'accent dans le mot prononcer par l'utilisatrice et est capable de le comparer.
- Le système doit être capable de reconnaître le mot prononcé par l'apprenant.e
- Disponibilité en au moins une langue parmi (Français, Italien, Allemand, Anglais et Espagnol)

- Les données vocales fournies par les utilisateurs ne pourront pas être utilisées à autre chose qu'à améliorer le système de reconnaissance vocale et ce seulement avec leur accord.
- L'interface doit permettre une prise en main rapide pour tout type d'utilisateur.

Hors du scope défini du projet

- Apprentissage adaptatifs : le mot prononcé faux revient
- Création de plus d'exercices
- Lors d'une session d'apprentissage le type d'exercice varie et est généré automatiquement.
- Utilisation des données des apprenant.e.s pour enrichir la base de données vocale du système.
- Connexion à l'API Miaparle.
- Faire un retour à l'utilisatrice sur la position temporelle des syllabes qu'elle a prononcées.

2.2. Contraintes techniques

- Le système est capable d'enregistrer le son et de le mettre dans le format désiré pour pouvoir être traité par l'API choisie.
- La solution doit être intégrable à Moodle sous la forme d'un plugin
- Le langage pour créer un plugin Moodle est le PHP
- Une ou des API.s devront être intégrées pour permettre la reconnaissance vocale. Le choix de de ce.s API.s devra prendre en compte le coût, les langues disponibles, le format de retour et de traitement, la capacité de l'API à reconnaître les accents.

2.3. Use case

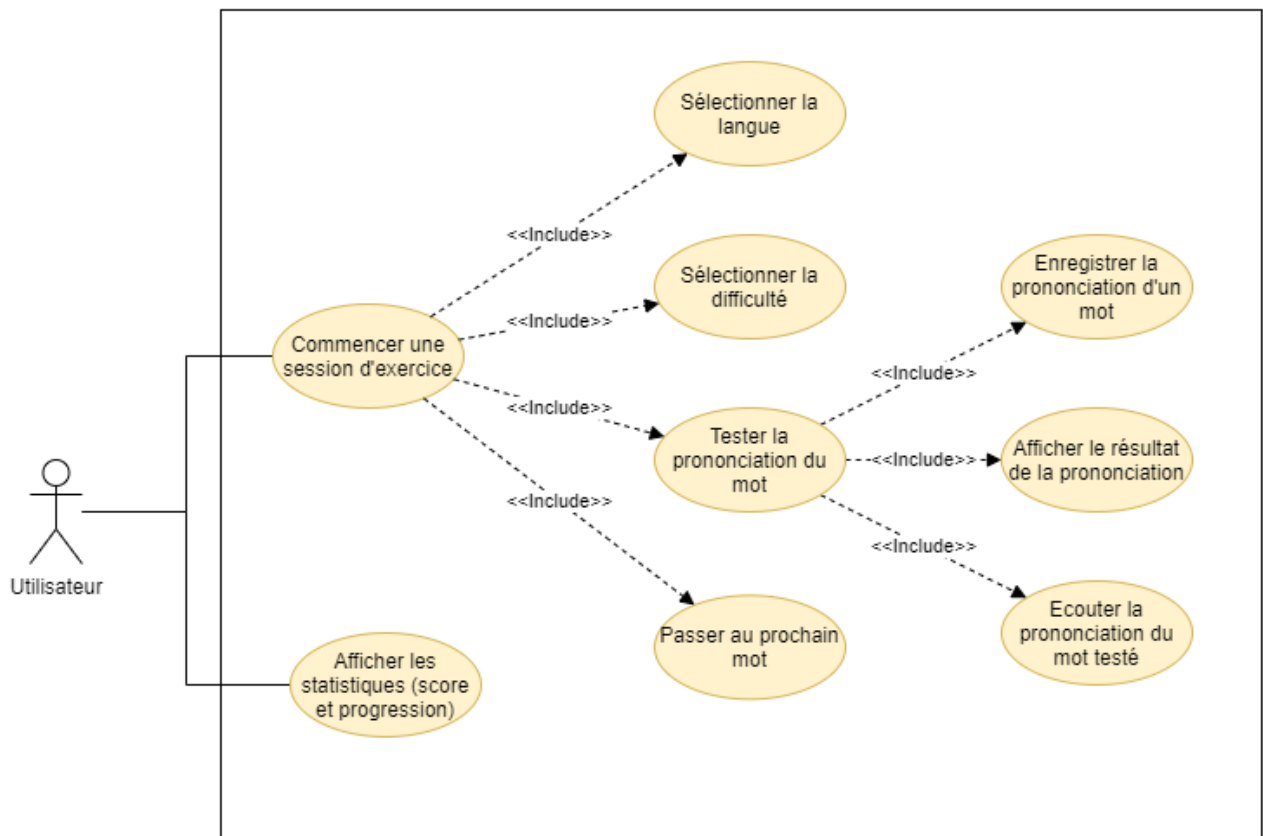


Figure 1 : Use case Utilisateur

Dans le diagramme des Use Case, nous retrouvons les cas d'usage principaux de l'application. Un utilisateur peut soit commencer une session d'exercices, soit afficher ses statistiques. Dans notre implémentation finale, nous n'avons finalement pas implémenté l'affichage de statistique. L'utilisateur va donc pouvoir suivre un seul scénario principal qui est : faire une session d'exercice. Plusieurs options vont s'offrir à lui, le choix de la langue, la difficulté. Ensuite, l'utilisateur peut tester sa prononciation du mot et analyser son résultat pour se corriger. Il peut aussi réécouter le son original ou sa propre voix pour comparer et comprendre son erreur. Il peut ensuite naviguer d'un mot à l'autre jusqu'à la fin de l'exercice.

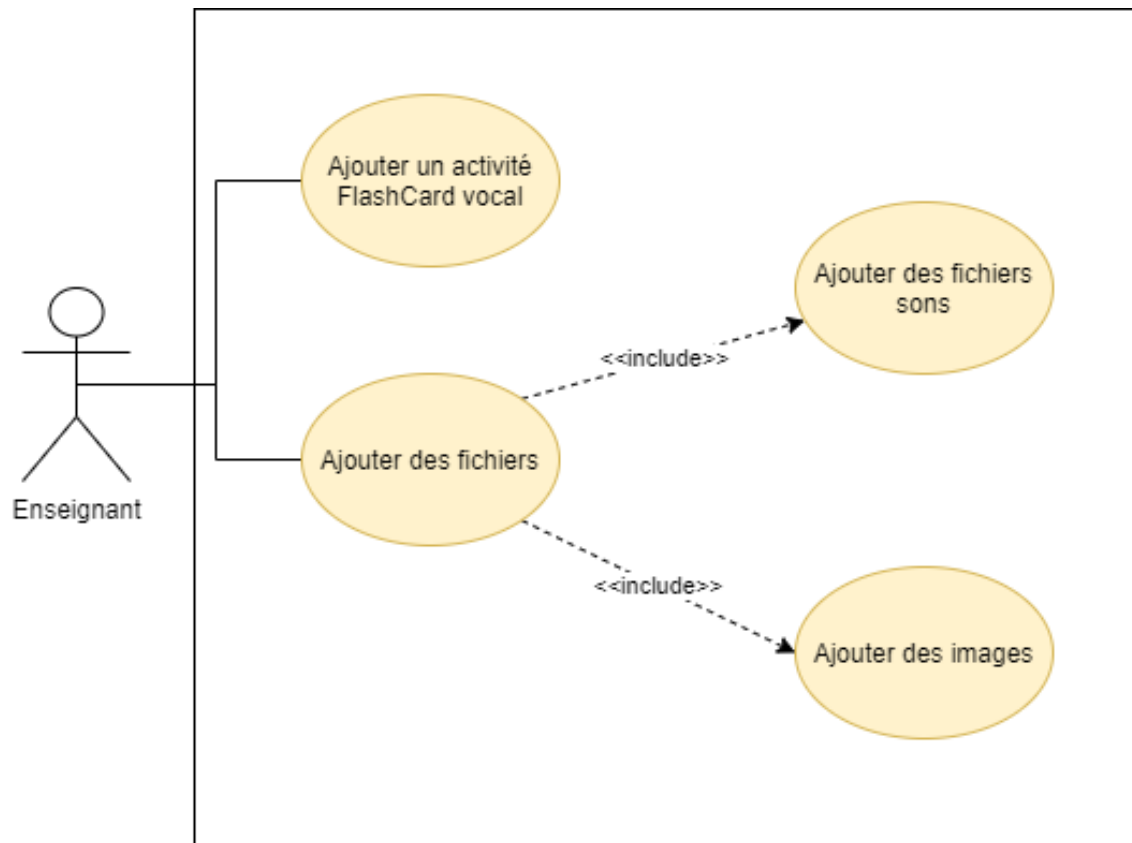


Figure 2 : Use case Enseignant

En plus de l'utilisateur, nous avons un autre acteur important qui est l'enseignant. Ce dernier est le responsable du cours et peut inclure dans son cours le module FlashCard vocal. Il peut également ajouter tous les médias qu'il désire, son et image. Il faut seulement qu'il nomme de la même manière l'image et le son.

2.4. Déroulement du projet

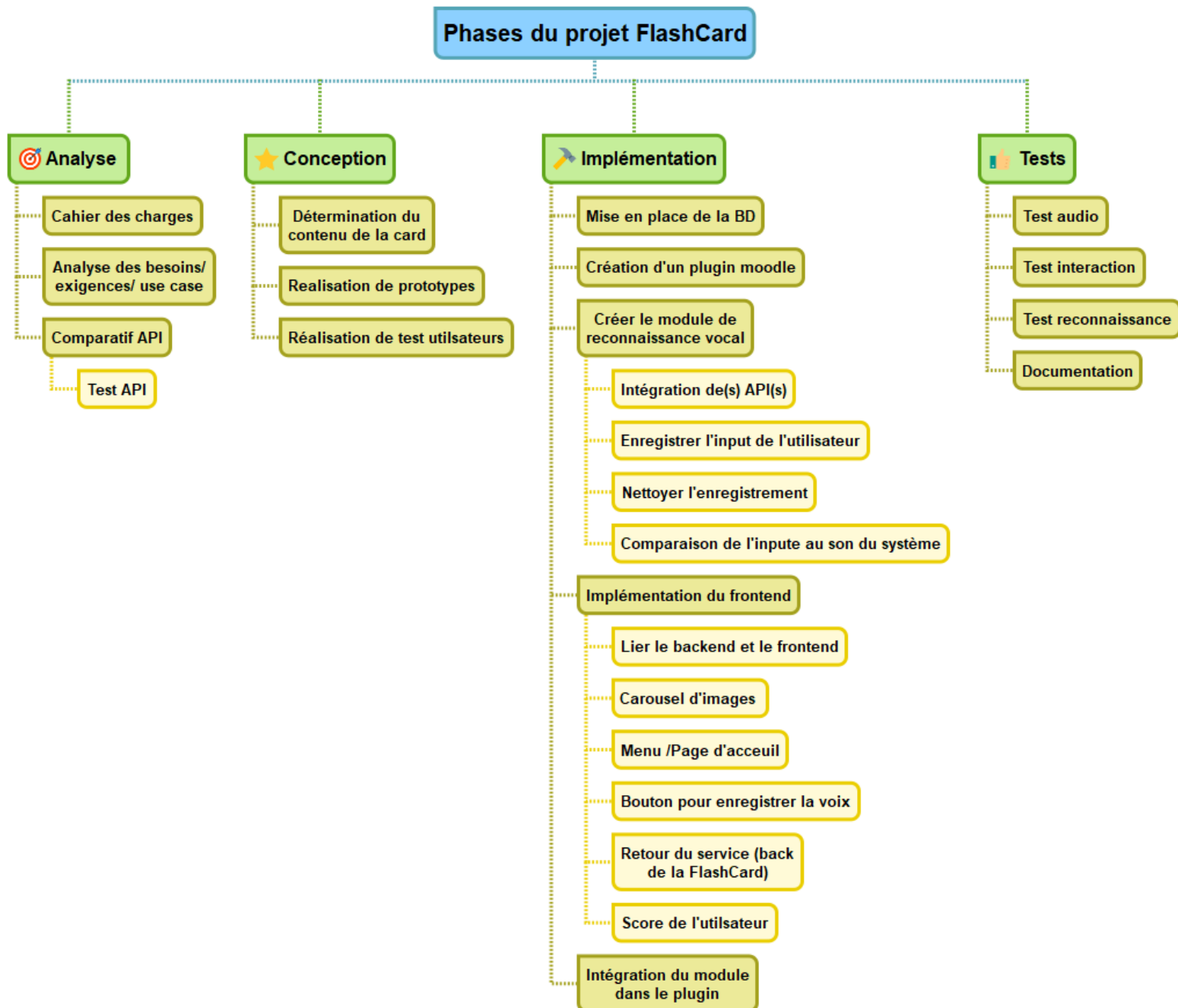


Figure 3 : WBS

Pour nous aider dans notre projet nous avons fait un plan sous forme WBS (Work Breakdown Structure). Grâce à cette structure de référence nous avons pu travailler en étant très au clair sur les tâches à accomplir. De cette décomposition, nous avons pu faire l'intégralité des points sauf la réalisation de tests utilisateurs, pour des questions de manque de temps, ainsi que le nettoyage de l'enregistrement, car finalement ce n'était pas nécessaire, car l'API que nous avons choisie détecte les silences automatiquement. Une

amélioration pourrait être faite sur d'autres nuisances potentielles. Mais entre la détection vocale de Google et le traitement de BAS Web Services, le rendu est bon, en tout cas dans un cadre de travail convenable.

2.5. Comparatif des APIs

En pièce jointe vous trouverez le détail de nos recherches sous format d'un tableau comparatifs.

Dans le cadre de notre projet nous avons finalement fini par utiliser l'API de l'université de Munich BAS Web Services (WebMAUS) pour le découpage en syllabes qui nous permettra de calculer le score. Nous avons aussi utilisé l'API de Google pour la détection vocale, car l'API BAS Web Service n'a pas rendu cette partie disponible. La combinaison des deux nous donne un rendu cohérent. Il faut juste faire attention car l'API de Google est payante et offre seulement 60 minutes gratuites par mois.

3. Prototype

La phase de prototypage est une phase clé de chaque projet. Durant cette phase, nous déterminons le résultat attendu tant par son design que par ses interactions. Pour réaliser cela nous avons utilisé l'outil Figma et réalisé deux prototypes.

Vous pouvez tester ces prototypes en cliquant sur les liens suivants :

- [Prototype V1](#)
- [Prototype V2](#)

Le premier prototype avait pour but d'être aussi proche que possible des couleurs de l'université pour s'intégrer dans moodle.

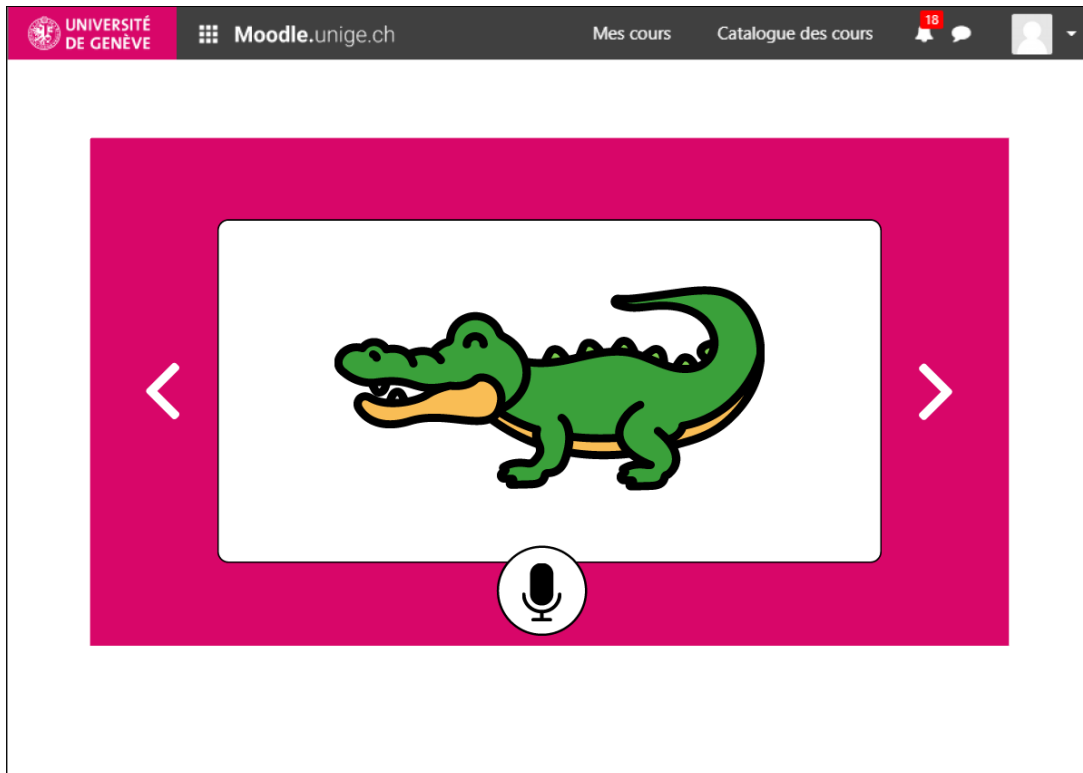


Figure 4 : Prototype 1

Alors que le deuxième, avait dans l'idée de se différencier grâce à un style un peu plus vintage.



Figure 5 : Prototype 2

Nous avons finalement choisi de faire notre plugin basé sur le second design. Effectivement, ce design avait été bien reçu lors de la présentation de nos designs et c'était notre favori.

3.1. Plan des interactions de notre prototype

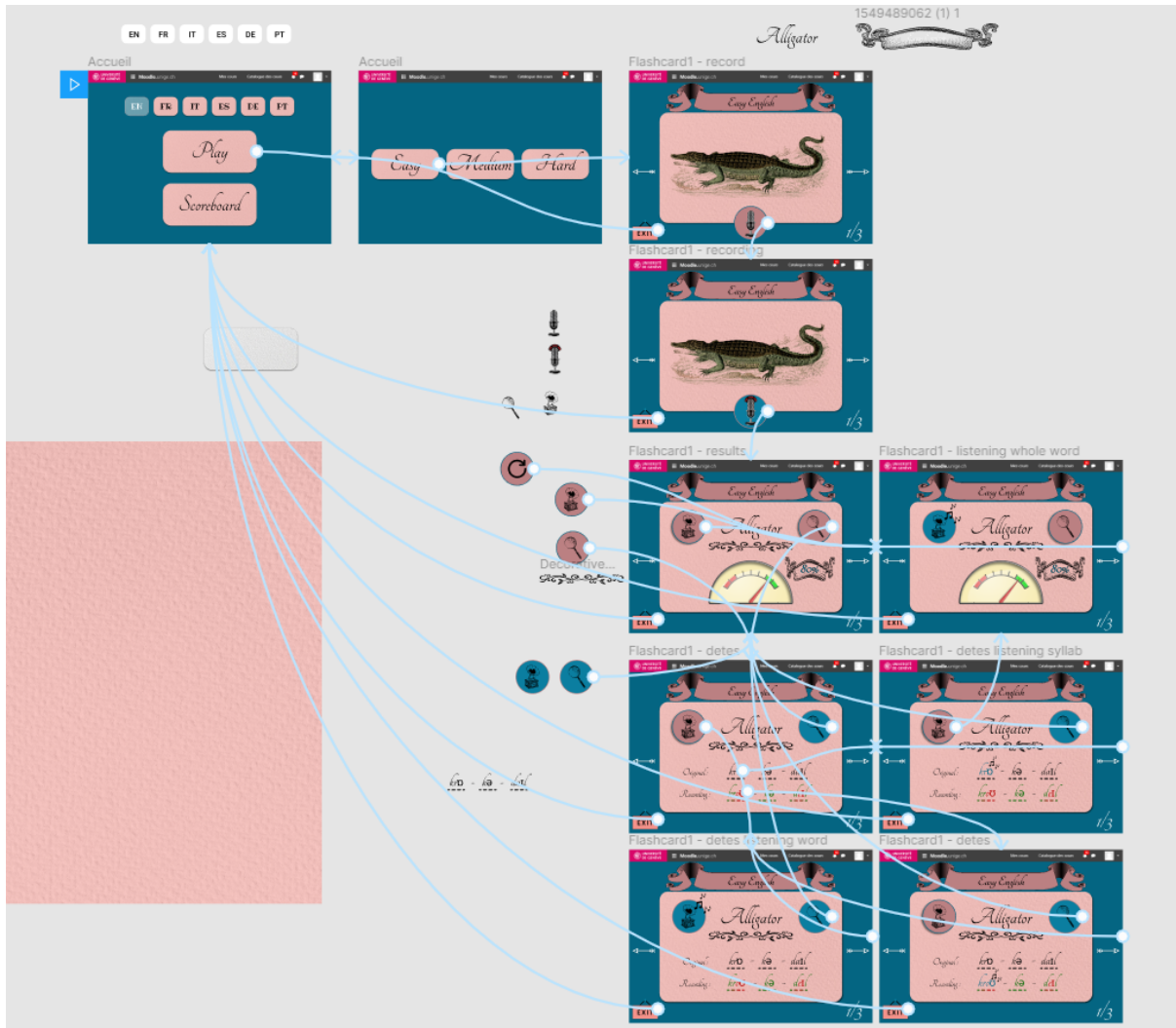


Figure 6 : Interactions du prototype

Grâce à la réflexion avancée au niveau des interactions que nous avons fait lors de la phase de prototypage, nous avons gagné du temps lors de l'implémentation. Nous n'avons plus à penser à ce qui devait se passer quand et où renvoyait telle ou telle action.

3.2. Détails des écrans

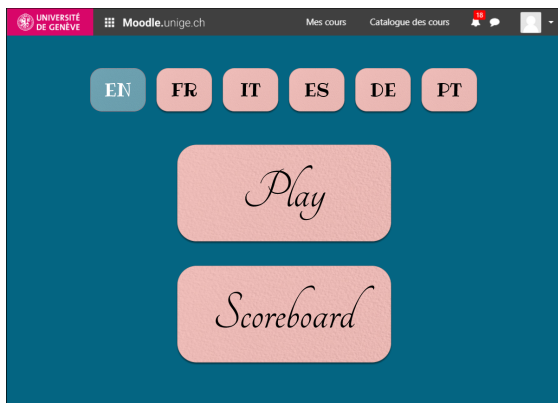


Figure 7: Écran d'accueil

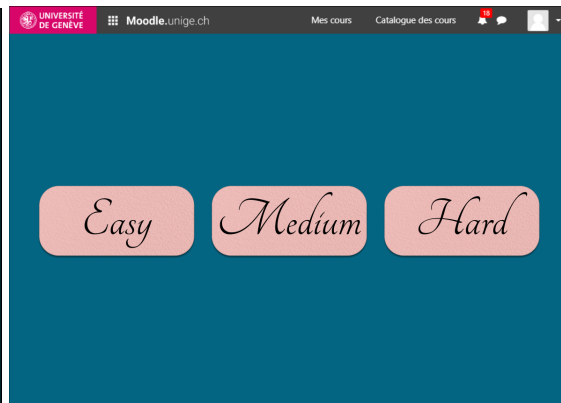


Figure 8: Choix de la langue

Les premiers écrans figures 7 et 8 permettent le choix des paramètres que l'utilisateur désire pour son exercice. Donc le choix de la langue et de la difficulté. Appuyer sur Scoreboard afficherait un dashboard de score, mais cette partie n'a pas été implémentée.



Figure 9: Exercices

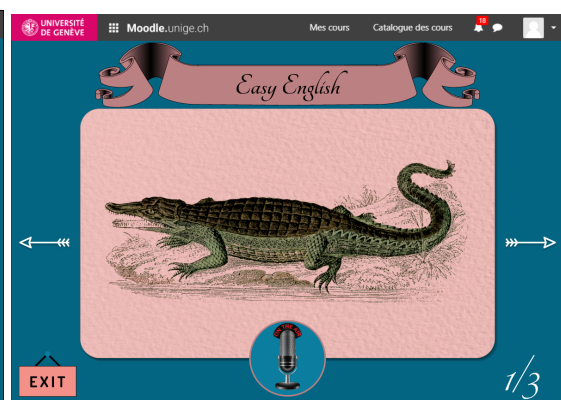


Figure 10: Enregistrement

Les écrans des figures 9 et 10 affichent le premier exercice et permettent l'enregistrement. Cliquer sur les flèches permet de passer à l'exercice suivant. En bas de l'écran, on peut cliquer sur exit pour revenir à l'écran d'accueil. On peut aussi voir notre niveau de progression dans la session d'exercice en bas à droite.



Figure 9: Résultats

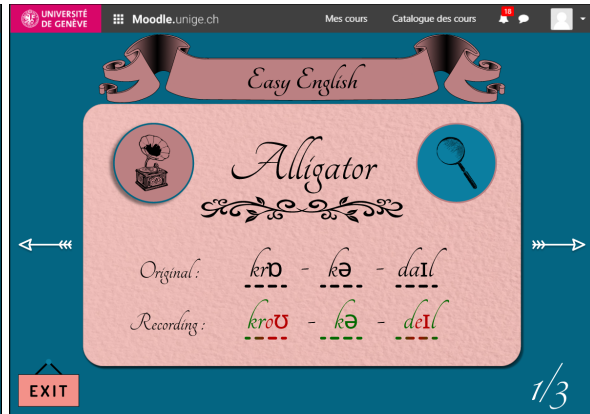


Figure 10: Détails des résultats

Les figures 9 et 10 représentent les résultats de l'exercice après que l'utilisateur ait enregistré sa voix. En cliquant sur le gramophone, l'utilisateur peut écouter le son original du mot pour savoir quelle était la bonne réponse et comment la prononcer. Sur l'écran Résultat (figure 9) on affiche le score globale. En cliquant sur la loupe, on peut voir plus de détails (figure 10), on y trouve la décomposition du mot en syllabes, en premier la version juste et en dessous tel que l'utilisateur l'a prononcé. Le vert indique que le son est juste et le rouge les erreurs de l'utilisateur.

4. Architecture

Dans cette section, nous allons nous intéresser plus concrètement à l'implémentation de notre plugin.

4.1. Schéma d'architecture

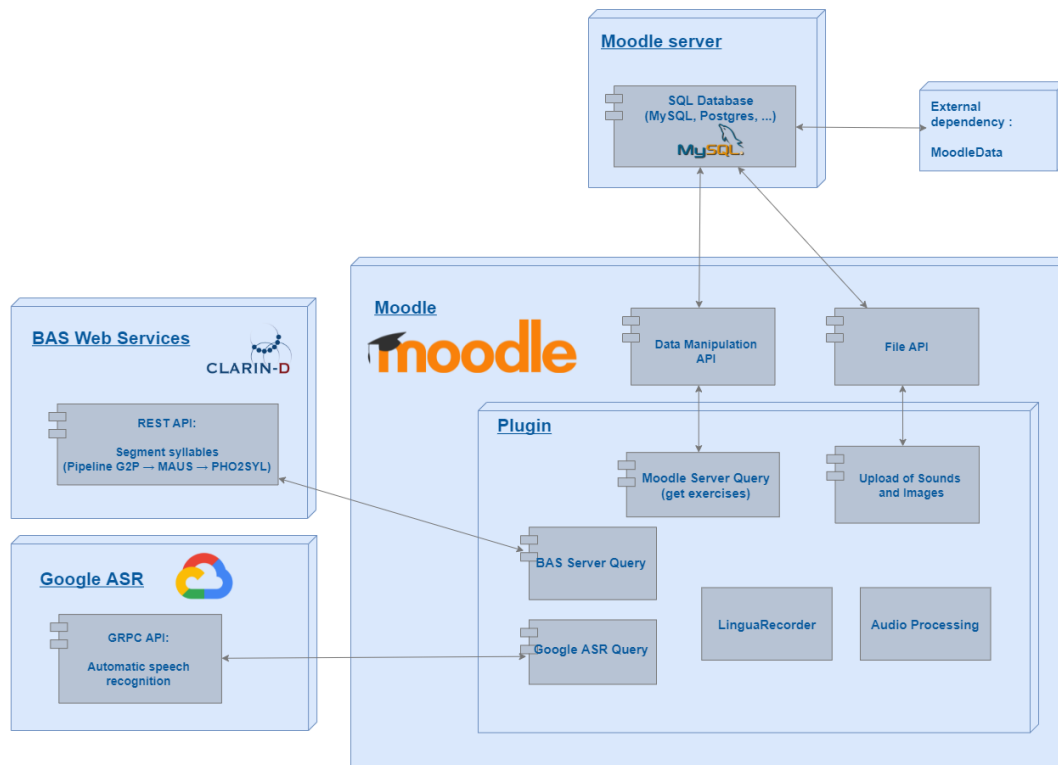


Figure 11: Schéma d'architecture

Le schéma décrivant toute l'architecture de notre projet est visible sur la figure 11 ci-dessus.

La première chose essentielle à relever est que le projet a été développé avec le framework de moodle pour pouvoir directement s'intégrer comme étant une activité d'un cours. C'est donc pour cela que le plugin se trouve à l'intérieur du composant **moodle**. Cela nous fournit également des API pour communiquer avec la base de données liée à moodle qu'on utilise pour uploader des fichiers et aussi pour chercher des images.

Ensuite, le plugin fait appel à deux API différentes pour traiter des segments audios : **BAS Web Services** (WebMAUS) et **Google ASR** (Google Cloud). Le premier service nous indique où se trouve les syllabes dans le fichier audio envoyé et le deuxième fait de la reconnaissance vocale automatique aussi basée sur fichier audio.

Finalement, nous avons également utilisé la librairie Javascript nommée **LinguaRecorder** qui nous permet d'enregistrer ce que l'utilisateur prononce et de manipuler cet enregistrement avec son composant **AudioRecord**.

4.2. Schéma base de données

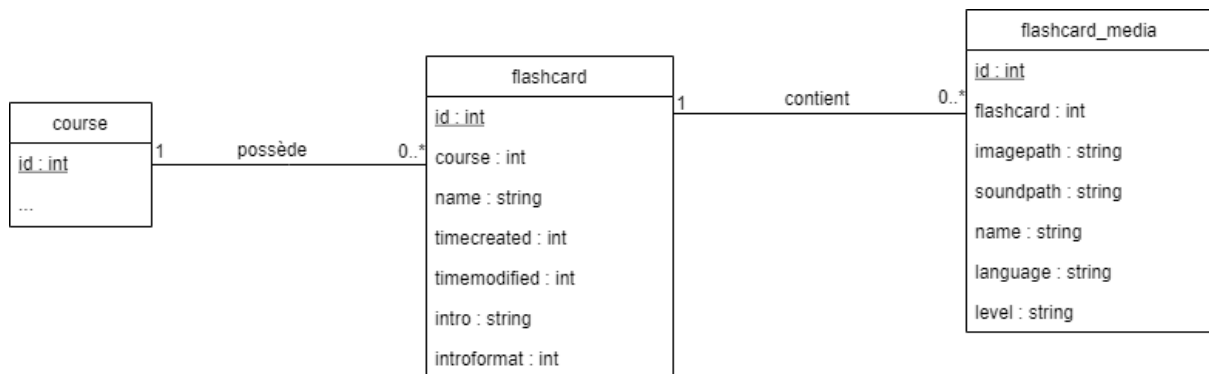


Figure 12 : schéma de la base de données

Pour ce projet, il était important de pouvoir distinguer chaque instance de l'activité et de sauvegarder les sons et les images de chacune de ces instances. C'est-à-dire que l'ajout de médias pour une instance ne modifierait pas les autres instances. C'est pourquoi nous avons créé et ajouté deux classes à la base de données de moodle qui sont *flashcard* et *flashcard_media*.

Notre première classe *flashcard* contient simplement les métadonnées d'une instance de l'activité. On retrouve également une référence au cours auquel cette activité appartient sous la forme d'une clé étrangère, ce qui permet de les différencier pour chaque cours. La classe *course* n'est pas détaillée sur ce schéma, car c'est une classe de base de moodle.

Ensuite, *flashcard_media* contient les informations d'un exercice appartenant à une instance spécifique de *flashcard*. On retrouve donc les chemins vers les fichiers image et son, le nom, le langage et le niveau de l'exercice.

4.3. Moodle (structure du plugin)

Comme nous avons développé un plugin pour moodle, nous avons dû nous adapter au framework (principalement php) de moodle. Cela nous a donc forcé à adopter une structure de base spécifique pour le plugin. Comme nous développons un plugin de type activité, c'est-à-dire qui est ajoutable dans les sections d'un cours, nous avons dû créer un nouveau dossier comportant le nom de notre plugin dans le dossier *mod* de moodle.

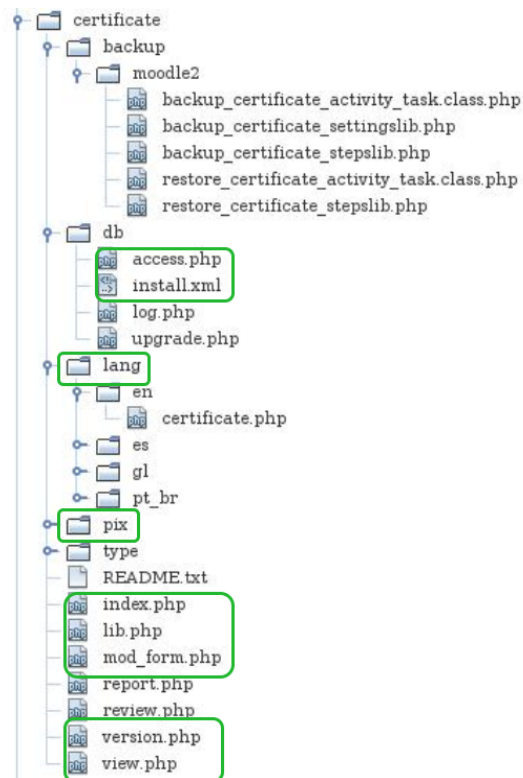


Figure 13: exemple de structure d'un plugin

Sur la figure 13 se trouve un exemple d'une structure d'un plugin pour une activité venant de la documentation de moodle. Dans cette structure, certains fichiers et dossiers doivent obligatoirement être présents, sinon le plugin ne fonctionnera pas du tout. Nous allons rapidement décrire les éléments obligatoires qui ont été essentiels au développement de notre projet qui sont mis en évidence sur la figure 13.

Tout d'abord, *access.php* décrit les permissions pour les utilisateurs concernant ce plugin. On peut donc définir qui a le droit d'instancier cette activité ou la consulter par exemple. Dans le même dossier, *install.xml* est un fichier important qui contient les définitions des tables propre à ce plugin à créer dans l'environnement de moodle. Les classes *flashcard* et *flashcard_media* sont donc définies dans ce fichier.

Le dossier *lang* contient les strings à récupérer et permet de charger une traduction pour une langue dynamiquement, mais certains mots de base doivent obligatoirement être définis pour le bon fonctionnement du plugin. Le dossier *pix* contient l'icône du plugin et peut également contenir d'autres images utilisées par le plugin.

Le fichier *index.php* cherche les instances de l'activité liées au cours actuel et affiche les liens vers chacune d'entre elles. *lib.php* est très important, car ce fichier définit comment créer et mettre à jour une instance du module, mais également comment servir des fichiers (nous reviendrons sur ce point lorsqu'on discutera des APIs). Il est appelé après *mod_form.php* qui contient la définition du formulaire contenant les paramètres d'une instance du plugin.

Finalement, *version.php* contient les métadonnées (version, nom) du plugin et *view.php* définit la vue du plugin.

Il y a d'autres fichiers/dossiers qui sont nécessaires pour que le plugin fonctionne, mais ceux décrit précédemment sont ceux que nous avons le plus explorés durant le développement.

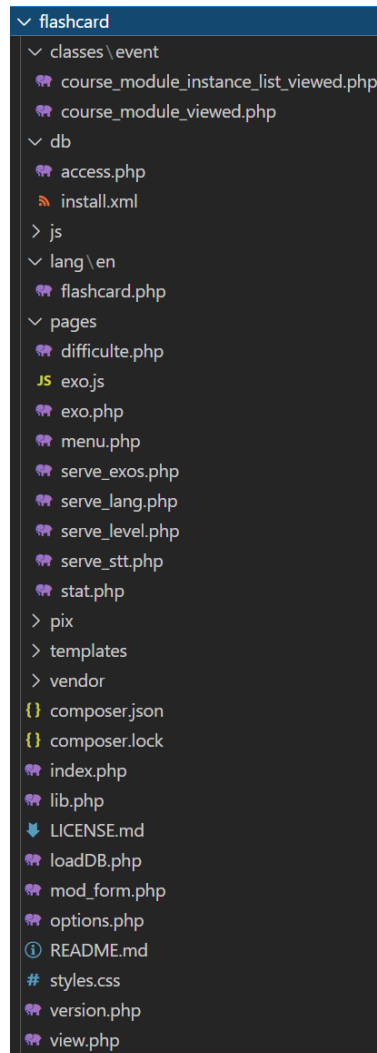


Figure 14 : aperçu de la structure du plugin flashcard

Sur la figure 14 se trouve la structure finale de notre plugin. On a donc ajouté la gestion des pages du plugin dans le dossier *pages* qui sont appelés dans la vue. Nous avons également ajouté des méthodes pour communiquer avec la base de données dans *loadDB.php*. Notre dossier *pix* contient aussi des fichiers sample pour charger des exercices par défaut disponibles pour toutes les instances de flashcard. Finalement, tout le style CSS du plugin se trouve dans *styles.css* et nous avons installé Google speech-to-text à l'aide de Composer, d'où la présence des fichiers *composer.** et du dossier *vendor*.

Pour faciliter la création du squelette du plugin, nous avons utilisé le plugin [Moodle plugin skeleton generator](#). Cela nous a beaucoup aidé, car la documentation n'est pas toujours claire, et nous avons dû étudier les autres implémentations de plugin dans le dossier *mod* lorsqu'il nous manquait quelque chose.

4.4. APIs

Comme représenté sur notre schéma d'architecture, notre projet utilise plusieurs APIs pour traiter les enregistrements, mais également pour communiquer avec l'environnement de moodle. Ces deux aspects sont explorés dans cette section.

4.4.1. Moodle

Pour communiquer avec la base de données, nous avons utilisé l'API *Data Manipulation* fournie par moodle pour facilement récupérer et insérer des données. Celle-ci a été utilisée dans des fichiers comme le *loadDB.php* et les *serve_*.php* (excepté celui utilisé pour la reconnaissance vocale).

Ensuite, pour gérer l'upload de fichiers par l'utilisateur, nous avons utilisé l'API *File* qui gère le stockage des fichiers sous forme hachée. Elle a donc été utilisée lors de la construction du formulaire dans *mod_form.php* et dans le *lib.php* pour sauvegarder les changements.

4.4.2. APIs externes

Nous avons dû faire appel à 2 APIs différentes pour traiter les enregistrements audios. La première **Google ASR** (Automatic Speech Recognition) qui nous permet de transcrire l'enregistrement de l'utilisateur. Cette API utilise le protocole gRPC pour lequel nous avons dû installer des packages dans le plug et elle nous retourne un ensemble de transcription possible accompagné d'un taux de confiance pour chaque possibilité.

La deuxième est le **BAS Web Services** (WebMAUS) qui fournit un éventail de services de transcription et de traitement audio. On souhaitait initialement utiliser leur service transcription ASR, mais, malheureusement, il est toujours sous forme expérimentale et n'est donc pas exposé à leur API REST. Nous avons utilisé leur service de pipeline combinant diverses fonctionnalités, afin de retourner un fichier contenant les timestamps des différentes syllabes d'un enregistrement. Cela nous permet donc de segmenter nous même les enregistrements pour chaque syllabe.

4.5. Traitement Audio et Scoring

Le traitement audio dans le fichier *exo.js* combine donc les deux APIs présentées précédemment pour segmenter les enregistrements, ainsi que pour calculer un score. Nous allons donc rapidement décrire le déroulement de cette procédure dans cette section :

- Nous utilisons *LinguaRecorder* pour enregistrer la prononciation de l'utilisateur.
- Cet enregistrement est envoyé à l'API Google ASR (au travers du fichier *serve_stt.php*) et nous retourne les possibilités de transcription.
- On compare les possibilités et on met à jour le score
- L'audio et la transcription sont envoyés à l'API BAS qui nous retourne les timestamps des syllabes.
- L'audio est segmenté à l'aide d'*AudioRecord* (composant de *LinguaRecorder*).

Certaines de ces actions sont effectuées de manières asynchrones et peuvent donc être effectuées en parallèle. La segmentation de l'audio de la réponse est également effectuée en parallèle, on ne perd donc pas de temps supplémentaire.

Concernant le scoring, nous avons 2 méthodes dans notre plugin : une se basant sur le taux de confiance retourné par l'API Google ASR, et une autre comparant les syllabes.

La première reprend le taux et l'ajuste pour laisser une marge de tolérance pour que l'évaluation ne soit pas trop stricte.

La deuxième compare les syllabes de la réponse avec les syllabes de la transcription de l'utilisateur. A cette fin, nous avons fait usage de la librairie externe [string-similarity](#) qui calcule la similarité basée sur l'indice de Sørensen-Dice.

Nous avons ensuite identifié 3 cas différents dans lesquels on applique soit l'une des deux méthodes, soit les deux méthodes :

1. La transcription avec le plus grand taux de confiance est correcte. Dans ce cas, on ajuste simplement ce taux de confiance.
2. La transcription correcte est dans les alternatives. Ici, on va effectuer les deux méthodes et prendre la moyenne.
3. Aucune des transcriptions n'est correcte. Cette fois-ci, on va prendre la transcription la plus fiable et comparer les syllabes.

5. Résultat

Dans cette partie nous exposerons le résultat de notre travail. La première partie consistait à créer une activité moodle pouvant s'intégrer dans un cours.

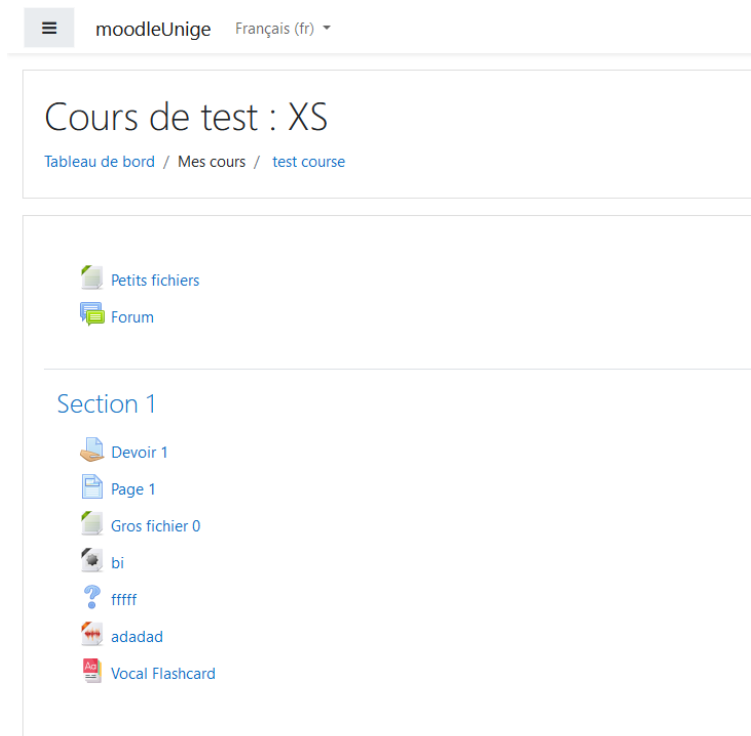


Figure 15: Intégration moodle

L'activité est devenue un des modules disponibles pour les enseignants, ils leur suffit de la sélectionner pour l'intégrer à leur cours. Sur la figure 15 on peut voir notre activité dans la section 1.

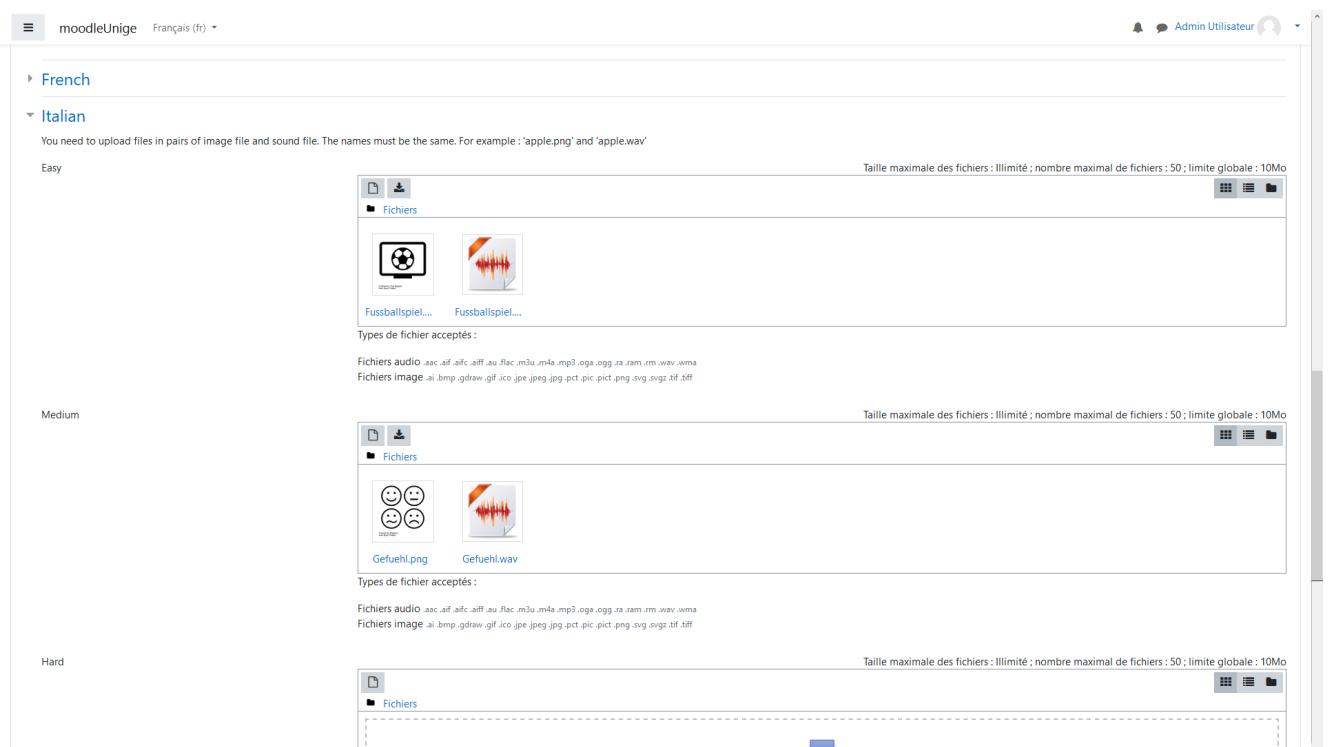


Figure 16: Ajoute de média

L'enseignant peut ensuite ajouter tous les médias qu'il désire dans la langue choisie et dans le niveau désiré. L'upload de fichier se fait de la même manière que partout ailleurs sur moodle. Nous avons prédéfini des sections, voir Figure 16, pour les différentes langues et les différents niveaux. Ainsi les fichiers peuvent être facilement classés par l'enseignant. Il a pour seule restriction de donner le même nom à l'image et au son correspondant, cela est indiqué en dessous de chaque langue.

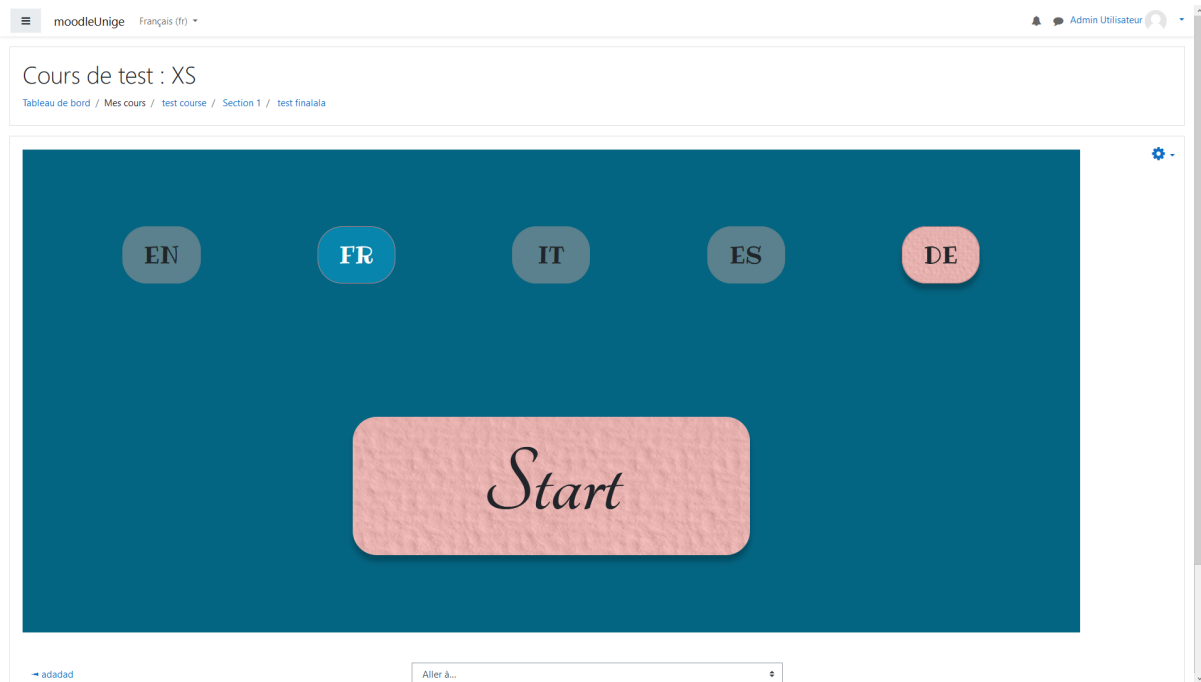


Figure 17: Écran d'accueil

Une fois qu'on entre dans l'activité, on arrive sur l'écran d'accueil, figure 17, sur cet écran on peut sélectionner la langue. Les langues en couleur sont celles qui contiennent du contenu ajouté par les enseignants. Les langues grisées n'ont pas de contenu disponible. Une fois que l'on a sélectionné une langue elle devient bleu claire, comme le FR dans l'image de l'exemple. Une fois la langue choisie, l'utilisateur peut cliquer sur Start. Cela le redirigera sur l'écran suivant, Figure 18.



Figure 18: Choix du niveau

Cet écran sert à choisir le niveau de difficulté, encore une fois les niveaux qui n'ont pas de contenus sont grisés. Cliquez sur un niveau pour commencer l'activité.

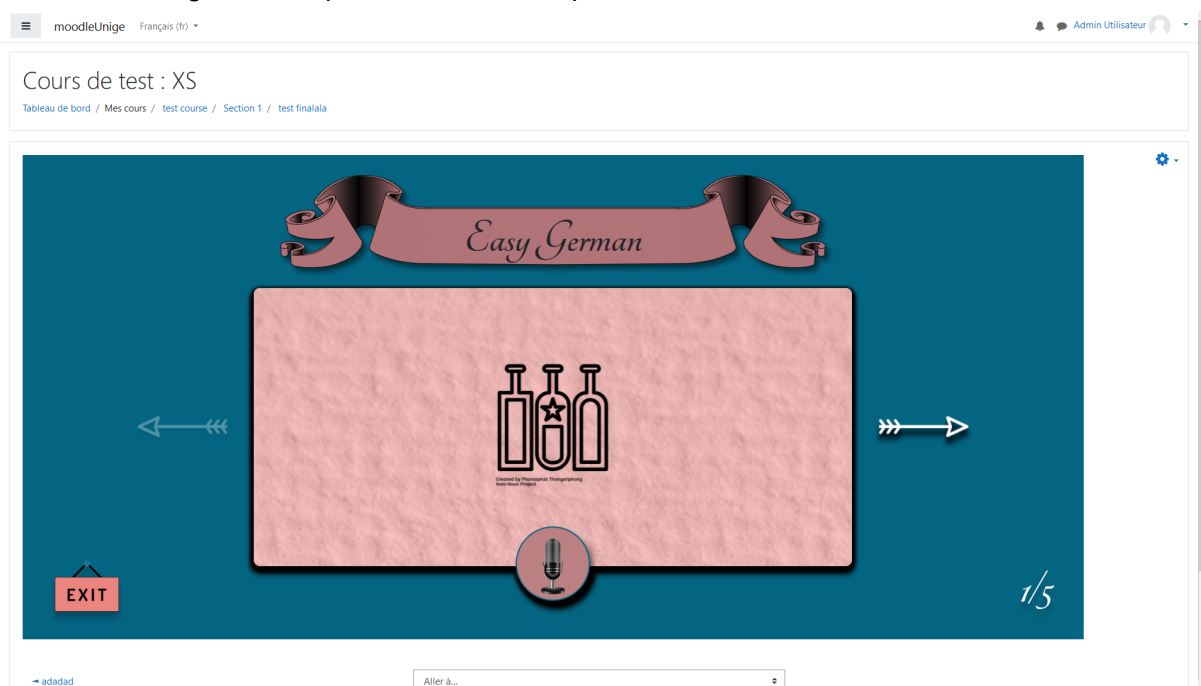


Figure 19: Exercice

Le premier exercice s'affiche, comme sur la figure 19, l'utilisateur voit une image et en cliquant sur le micro, il peut enregistrer sa solution. En bas de l'écran à droite il voit son avancée dans l'activité et à gauche il peut revenir à l'écran d'accueil quand il le désire.

Une fois que l'utilisateur a enregistré sa réponse il clique à nouveau sur le micro, une animation de chargement apparaît et cela lance le traitement de l'audio. Une fois le traitement terminé, la carte se retourne pour afficher le score, Figure 20.

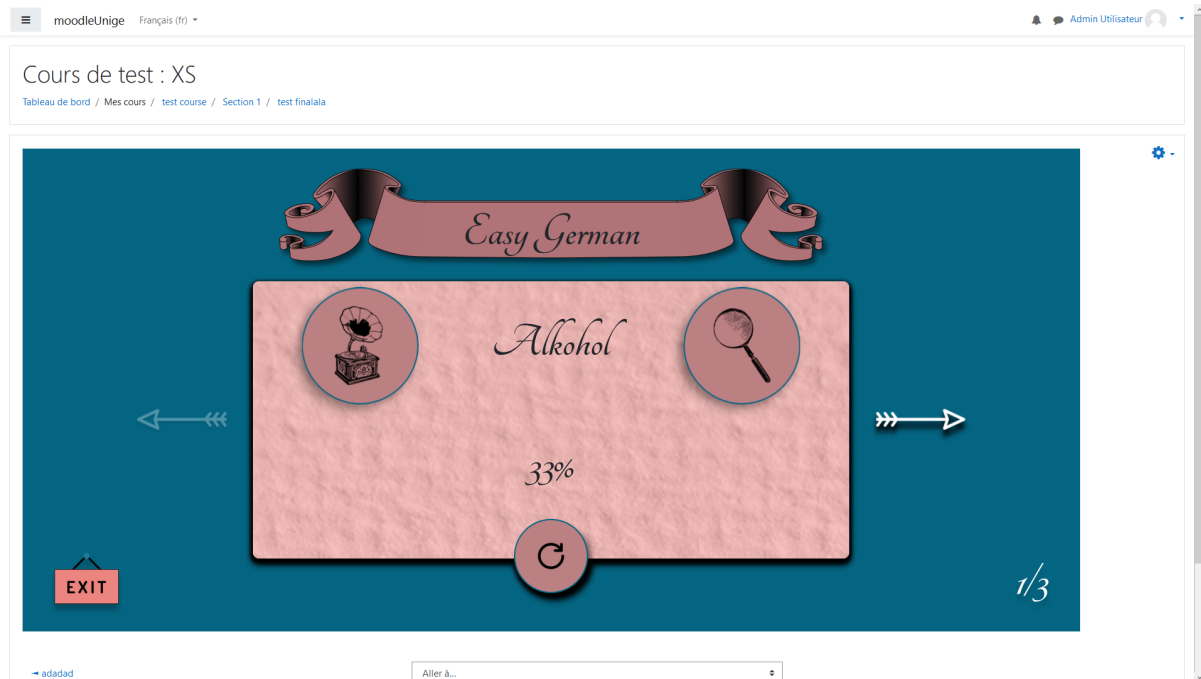


Figure 20: Score

Sur l'écran du Score, au dos de la carte testée, l'utilisateur peut voir son score. Sur cette page l'utilisateur peut écouter la réponse correcte en appuyant sur le gramophone, s'enregistrer à nouveau en cliquant sur la flèche "rafraîchir" en bas de la carte, ou il peut cliquer sur la loupe pour analyser ses erreurs plus en détails, Figure 21.

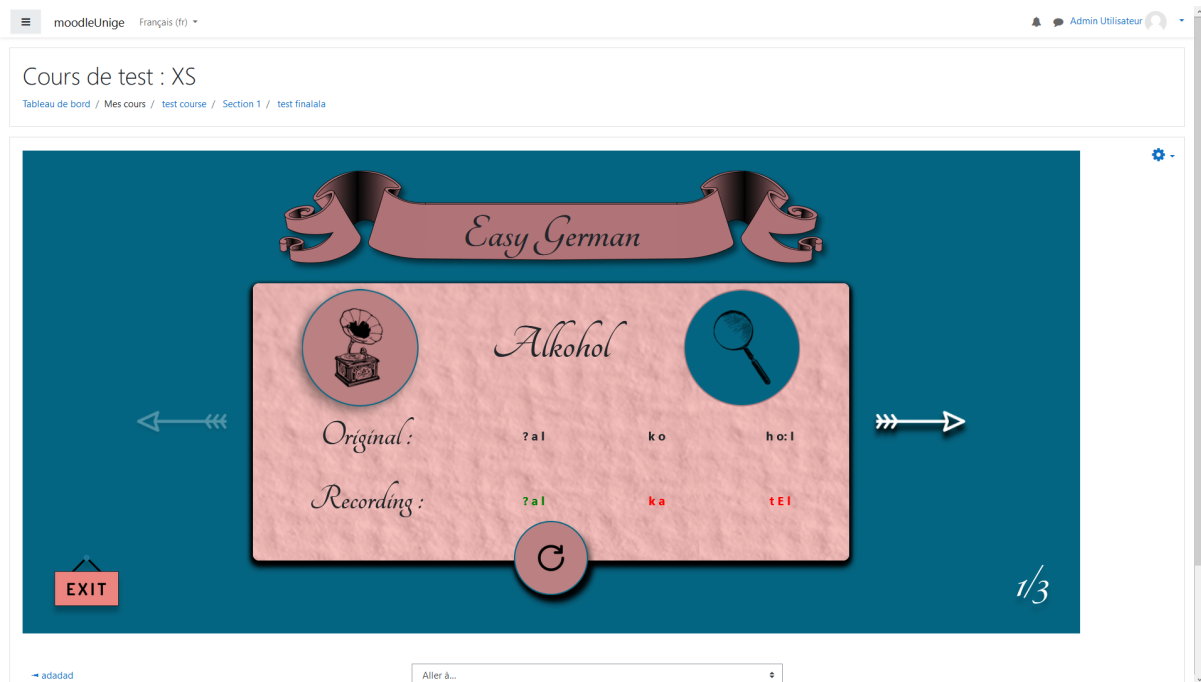


Figure 21: Détails

L'écran affichant les détails de l'analyse comparant la réponse enregistrée par l'utilisateur et le son original sont affichés, figure 21. Découpé en syllabe par l'API de BAS Web Services (WebMAUS), la réponse de l'utilisateur et la réponse originale sont comparées. En vert, l'utilisateur voit ce qu'il a fait juste et en rouge ses erreurs. En cliquant sur une syllabe, l'utilisateur peut écouter sa prononciation et la comparer avec la réponse. Il peut également écouter sa réponse entière en cliquant sur Recording. Il peut évidemment toujours écouter le son original grâce au gramophone ou s'enregistrer à nouveaux en cliquant sur la flèche "rafraîchir" en bas de la carte.

Une fois qu'il a fini l'exercice, l'utilisateur peut naviguer sur la carte suivante grâce aux flèches.

6. Prochaines étapes & Recommandations

Ce chapitre contient nos recommandations et nos suggestions pour le futur de ce projet.

Notre module est utilisable et prêt à être intégré dans le Moodle de l'université si vous le désirez. Cependant quelques améliorations pourraient être faites et quelques explications seront importantes à transmettre aux enseignants qui l'utiliseront.

Les améliorations que nous vous recommandons, sont les suivantes:

- Améliorer le score : le calcul de score que nous avons fait est assez rudimentaire. Nos méthodes de calcul ont l'air de fonctionner, mais elles ont été implémentées rapidement et manquent probablement de peaufinage. Nous pensons donc qu'elles pourraient être améliorées.
- Enregistrer les scores: Les scores de l'utilisateur ne sont pour le moment pas encore enregistrés. Nous vous recommandons de les enregistrer et de les afficher afin de

permettre à l'utilisateur de voir sa progression. Vous pouvez aussi utiliser l'enregistrement de ses scores pour créer un dashboard, montrant les statistiques de l'utilisateur. Nous avions initialement prévu d'implémenter cela, mais n'avons pas pu par manque de temps.

- Intégrer la reconnaissance d'accent: Vous pourriez aussi ajouter une interaction avec l'API MiaParle pour la reconnaissance des accents sur les mots.
- Ajouter des langues : Vous pourriez très facilement ajouter autant de langues que vous désirez tant que ces langues sont dans la liste des langues traitée par l'API de BAS Web Services et de Google ASR
- Ajouter des langues pour l'interface : Pour le moment, toute l'interface est uniquement en anglais, mais on pourrait détecter et utiliser la langue de l'environnement moodle pour améliorer l'accessibilité.

Les recommandations que nous vous suggérons d'appliquer pour le bon fonctionnement de l'application:

- Suivre les améliorations et mise à jour des APIs pour voir si de nouvelles propositions sont disponibles. Par exemple, nous avons remarqué que de temps en temps l'API de BAS Web Services est mise à jour. Donc, si elle rend disponible la reconnaissance vocale vous pourriez éliminer l'utilisation de Google ASR. Ils offrent également un service nommé WebMINNI qui segmente un enregistrement audio sans avoir besoin de texte en entrée, mais il n'est pas assez précis pour le moment.
- Faire un suivi des coûts avisé pour l'API google, vous disposerez de 60 minutes gratuite par mois, ensuite cela coûte de \$0.004/15 secondes jusqu'à \$0.009/15 secondes, selon le nombre d'utilisateur et la fréquence d'utilisation cela peut aller assez vite nous vous recommandons donc d'être très attentif à ce point, pour éviter de mauvaises surprises.
- Implémenter des restrictions sur le naming des fichiers ou bien informer les enseignants sur les bonnes pratiques des noms des fichiers. Il faut absolument que le nom de l'image et du son correspondant soit identique sinon il y aura des erreurs. Nous n'avons pas eu le temps de mettre en place une gestion d'erreur très poussée pour ce point, nous vous recommandons donc aussi de faire attention à cela.

7. Conclusion

7.1. Difficultés rencontrées

Nous avons passé beaucoup de temps à apprendre à utiliser le framework moodle. En effet, ce framework peut être intéressant à utiliser, mais la documentation est, pour la plupart du temps, très superficielle. Nous avons donc souvent dû parcourir les forums de moodle et regarder des vidéos sur le sujet, car la documentation officielle était souvent incomplète et/ou ne présentait pas tous les cas d'utilisation. Un exemple qui illustre bien à quel point cela est problématique est que nous avons dû utiliser un plugin **externe** pour générer le

squelette de notre plugin. Nous espérons donc qu'ils ajouteront beaucoup plus d'exemples dans leur documentation.

Ensuite, la recherche d'APIs nous a aussi pris beaucoup de temps, car c'était difficile de trouver des services qui s'occupaient exactement de ce dont nous avons besoin.

Finalement, nous sommes content d'avoir découvert BAS Web Services grâce au client, mais nous avons quand même dû utiliser Google ASR pour la reconnaissance vocale.

Le découpage d'un enregistrement à partir d'un fichier audio nous a également posé quelques problèmes. Lorsque l'audio provenait de l'enregistrement de la voix de l'utilisateur, on avait directement accès à l'API d'AudioRecord pour segmenter l'audio, car l'enregistrement se faisait au travers de LinguaRecorder et nous fournissait directement le bon format. Cependant, les fichiers audio stockés et ouverts séparément (les enregistrements des réponses) ne nous fournissent pas directement le bon format. Cela nous a donc forcé à convertir le format de l'audio afin de pouvoir utiliser l'interface d'AudioRecord. On a donc passé pas mal de temps à comprendre comment tout cela fonctionnait. Tout ce processus est visible dans la méthode *blobToAudioRecord* du fichier *exo.js*.

7.2. Notre évolution

Ce projet était un défi intéressant pour nous, autant techniquement que pour notre évolution personnelle. Nous avons dû faire preuve d'adaptabilité, nous avons découvert la plateforme Moodle qui est peu documentée. Ce nouveau framework a été un grand défi qui nous a appris beaucoup de choses. C'était une grande satisfaction pour nous de pouvoir appliquer nos apprentissages théoriques et de voir un résultat aussi fonctionnel! Nous n'avons pas codé en PHP en dehors du premier cours NTIC le semestre passé et nous sommes vraiment heureux de voir que cela a été suffisant pour mettre en place un tel projet. Par le biais de ce projet nous avons relevé un défi de taille en intégrant plusieurs API dans un projet.

Nous avons beaucoup apprécié de pouvoir avoir un vrai "client" dans le cadre de ce projet, car nous avons été très peu confrontés à cela dans le cadre du Bachelor. Cela nous a permis de faire une analyse des besoins et d'avoir un vrai échange. Le rythme régulier des séances était agréable et nous a mis dans un cadre de gestion de projet agile. Les sprints de deux semaines, nous permettaient d'avoir toujours des nouvelles choses à présenter. Nous avons beaucoup apprécié pouvoir avoir des retours fréquents, afin d'améliorer et recentrer les besoins.

S'il y a quelque chose qui manquait à notre projet, c'était le temps d'effectuer des tests utilisateurs, autant pour l'interface que pour la version finale. Nous aurions trouvé intéressant de passer un peu de temps avec des utilisateurs finaux, afin de voir si notre solution leur serait vraiment utile et afin de comprendre leurs désirs et besoins. Si le projet était à refaire, nous proposerions de faire passer des questions aux débuts du projet pour recueillir des idées et besoins de potentiels utilisateurs. Puis, nous aurions fait passer des tests utilisateurs dès le prototype afin de voir si l'interface UX correspond et recueillir des retours.

7.3. Le mot de la fin

Un grand merci pour l'organisation et l'accompagnement dont vous avez fait preuve tout au long de ce projet. Nous avons été heureux de réaliser ce projet et d'en sortir avec un résultat dont nous pouvons être fiers. Ce travail nous a beaucoup apporté et nous a permis d'appliquer de nombreuses connaissances théoriques du Bachelor : gestion de projet, interface utilisateur, analyse des objectifs, cours de programmation et modélisation. Tout en nous faisant découvrir de nouvelles choses, un nouveau framework, de l'intégration d'API multiple et la collaboration directe avec un client.