Cahier des charges

Contexte

Dans le courant de l'année 2020, le professeur Yves Chevallier a réalisé une plateforme web permettant à ses élèves de réaliser des quiz interactifs. Cette plateforme a été réalisée à l'aide du *framework* PHP, **Laravel** pour le backend et du *framework* javascript, **VueJs** pour le frontend.

Par la suite, un travail de Bachelor a été proposé afin que cette plateforme permette aux élèves d'y effectuer des travaux écrits. Cela apporte des avantages conséquents autant pour les professeurs que pour les élèves. En effet, la plateforme permet une correction partielle du travail écrit (notamment pour les questions QCM) mais elle permet aux élèves de pouvoir écrire du code et le compiler. Deux choses complétement impossibles avec des tests papiers.

Situation

L'ajout de travaux écrits a été en partie réalisée, mais elle est incomplète. Actuellement, le projet n'est pas complétement fonctionnel et la documentation contient pas mal de lacunes rendant la reprise de ce projet plus difficile que nécessaire. De plus, cette plateforme utilise d'anciennes versions de **Laravel** et de **VueJs**. Il conviendrait donc de mettre ces technologies à jour.

Objectif

Les objectifs de ce travail de Bachelor s'organisent autour de deux axes principaux :

1. Le remaniement du code : comme mentionné précédemment, il est important de tenir le projet et la documentation à jour afin qu'il soit le plus simple possible à reprendre et à modifier. C'est pourquoi je vais me baser sur le code existant afin de réécrire cette plateforme avec les technologies détaillées au chapitre suivant.
2. L'ajout de l'algorithme Super Memo de Anki permettant ainsi aux élèves de se driller avec des quiz de révision. Pour rappel, cet algorithme permet de noter avec quelle facilité l'étudiant répond à une question. En fonction cette notation, la question reviendra plus ou moins souvent. Cela permet de tomber plus régulièrement sur des questions nous posant un problème. Dans notre cas, nous n'allons pas demander à l'élève de noter la difficulté de la question, mais nous allons nous baser sur le temps que ce dernier a mis à y répondre.

Choix technologiques

Les choix technologiques faits par les personnes ayant précédemment travaillé sur ce projet sont cohérents. C'est pourquoi dans la plupart des cas, je vais juste utiliser les dernières versions de ces technologies.

- Backend : je vais utiliser la version 10 de **Laravel** (actuellement version 8 de Laravel) afin de créer une API REST pour l'application.
- Frontend : je vais utiliser la version 3 de VueJs (actuellement v2 de VueJs) afin de communiquer avec notre API et d'afficher l'interface aux utilisateurs. Je vais également remplacer **Vuex** par le nouveau state manager de VueJs, **Pinia**. De plus, je vais utiliser **TailwindCSS** comme *framework* pour l'interface graphique (actuellement **Bootstrap**). **TailwindCSS** est un *framework* très populaire et permet une plus grande flexibilité que Bootstrap. De plus, je suis à l'aise avec cette technologie et n'aurait donc pas besoin d'investir de temps afin de m'y familiariser.

- Data base : je continuerai d'utiliser la version 8 de **MYSQL.**
- Outil SSO : afin que les élèves et les professeurs puissent se loguer facilement sur cette plateforme, l'idée est d'utiliser **Keycloack** afin qu'ils puissent se connecter avec leur compte **SwitchAAI** (Actuellement Shibboleth). **Keycloack** est système de gestion d'identité en pleine croissance.

Objectifs fonctionnels

Ces objectifs sont en partie repris du TB de M. Stéphane Bouyiatiotis.

- Le projet doit avoir une documentation précise expliquant comment l'installer et le lancer
- Un utilisateur doit pouvoir s'identifier à la plateforme à l'aide de son compte switchAAI
- Un professeur doit pouvoir créer et ajouter des étudiants une classe
- Un professeur doit pouvoir créer, via une interface, plusieurs types de questions :
    - QCM
    - Texte à trou
    - Question à développement
    - Question de code
- Les questions utilisent un format markdown modifié et il doit donc y avoir une page de documentation expliquant comment créer chaque type de question.
- Un professeur doit pouvoir créer et gérer un quiz contenant des questions
- Un professeur doit pouvoir créer un travail écrit
  Un professeur doit pouvoir planifier un travail écrit
- Un travail écrit doit s'arrêter après la fin du temps imparti
- Un professeur doit pouvoir lancer la correction automatiques des questions simples (QCM, texte à trou)
- Une fois, le travail écrit corrigé, un professeur doit avoir accès aux statistiques de bonne réponse des questions.
- Un élève doit pouvoir faire des quiz.
- Un étudiant doit pouvoir utiliser le mode drill du quizz
- Un élève doit pouvoir répondre aux questions d'un travail écrit
- Un élève doit pouvoir compiler son code
- Un élève doit pouvoir rendre son examen avant la fin du temps imparti

Objectifs non-fonctionnels

- L'interface doit être fluide et intuitive pour les utilisateurs
- L'application doit être fonctionnelle et fiable
- Un CI/CD doit être mis en place afin de faciliter la reprise du projet et son déploiement
- L'application est open source et le code est hébergé sur **GitHub**
- Les réponses d'un élève ne doivent pas pouvoir être modifiée après la fin de l'examen
- Les questions de l'examen ne doivent pas être modifiables après l'examen en question
- Les messages d'erreur doivent être clairs et compréhensibles pour l'utilisateur