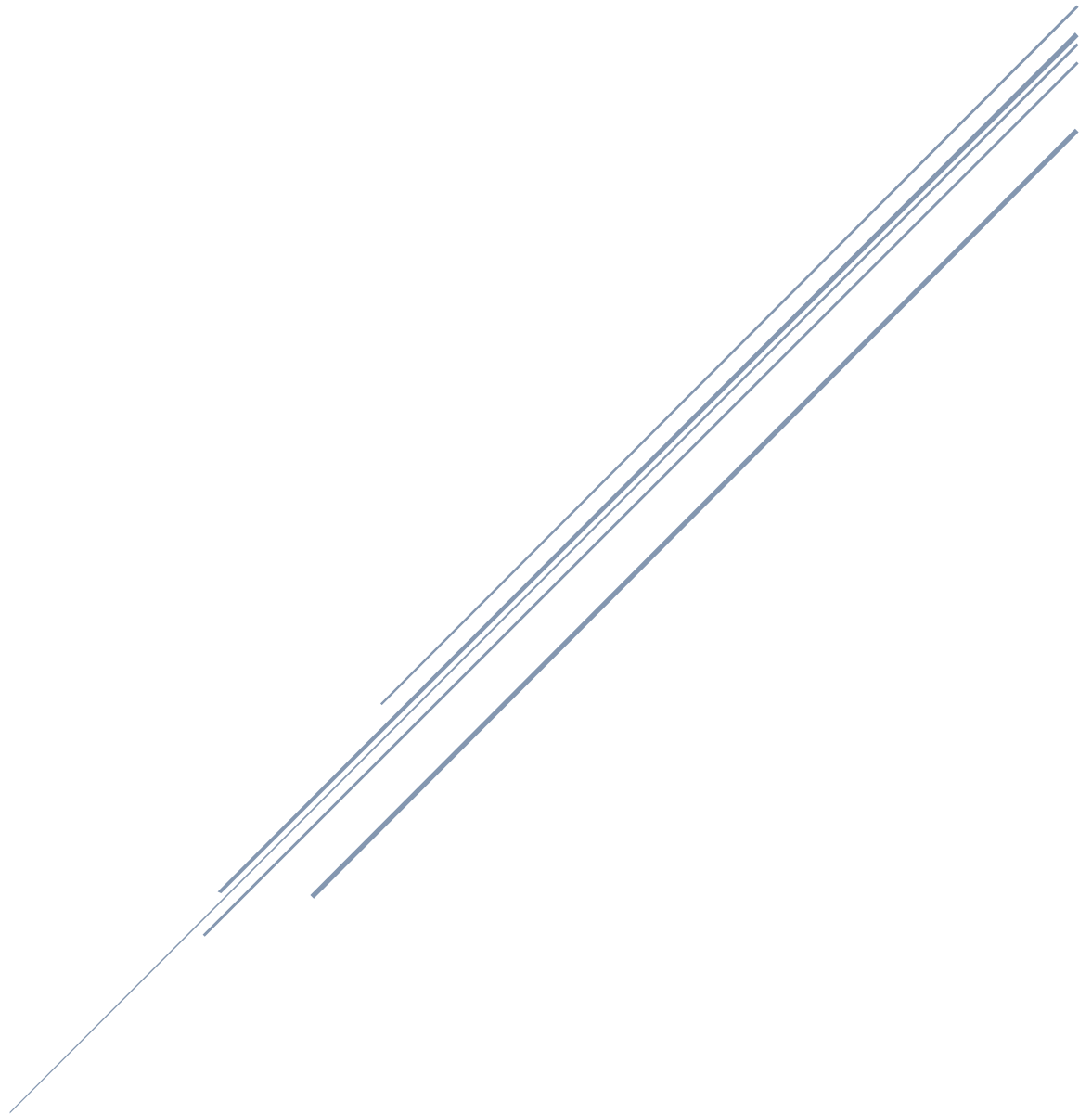


RAPPORT DE PROJET TUTEURÉ

Année 2015 - 2016



IUT Informatique – Lyon 1

Louis LAGIER – Sébastien RIGAUD – Vahic LAYDIER – Laurent MOREL

REMERCIEMENTS

En préambule, les membres du groupe se joignent pour remercier monsieur Christophe JALOUX pour avoir proposé le projet et assumé le rôle de tuteur. Il nous a permis de mieux planifier le projet en gardant un contact régulier avec nous et en nous proposant des repères temporels.

En tant d'instigateur du projet, il a également tenu le rôle du "client" en spécifiant ses besoins pour que nous sachions dans quelle direction avancer.

Ses précieux conseils, sa bonne humeur et son sens du dialogue ont largement contribué à faire de ce projet une expérience plaisante pour nous tous.

TABLE DES MATIERES

Remerciements.....	2
Table des matières.....	3
Partie 1 : Introduction	4
Présentation du projet	4
Environnement technique.....	5
<i>Composer</i>	5
<i>Git</i>	5
<i>GitHub</i>	6
Partie 2 : Réalisation du projet.....	7
Répartition des tâches	7
Architecture de l'application.....	7
<i>Back-end</i>	7
<i>Front-end</i>	13
Fonctionnalités.....	17
<i>Page d'accueil</i>	17
<i>Liste d'exercices</i>	18
<i>Résolution d'un exercice</i>	18
<i>Administration</i>	19
Partie 3 : Bilan individuel	21
Louis	21
Sébastien	21
Vahic.....	22
Laurent	22
Conclusion.....	23

PARTIE 1 : INTRODUCTION

PRESENTATION DU PROJET

Le projet consiste à réaliser un site internet de pratique d'exercices mathématiques.

Dans la même volonté éducative que le "projet Voltaire", le site a pour but d'entraîner les élèves qui peuvent suivre leurs résultats et leur progression par rapport au cours ainsi qu'aux connaissances qu'il contient.

Les professeurs peuvent mettre en ligne des exercices portant sur différents points du programme. Les élèves peuvent ainsi choisir sur quelle leçon travailler grâce à une catégorisation des exercices, par chapitre et par modules.

Enfin, le professeur peut consulter des statistiques individuelles afin de savoir ce qui est réussi ou non pour éventuellement orienter son cours.

Ce projet s'adresse aux étudiants de l'IUT mais pourrait être proposé à quiconque souhaitant avoir une application gérant des exercices pour ses élèves.

ENVIRONNEMENT TECHNIQUE

Dans cette section, nous allons décrire les principaux outils que nous avons utilisés pour réaliser ce projet.

COMPOSER



Composer est un gestionnaire de dépendances pour PHP. Il facilite l'installation, la mise à jour des dépendances du projet et permet de s'assurer que tous les développeurs utilisent les mêmes versions.

Pour l'utiliser, il faut lancer la commande "composer install" suivie du nom de la dépendance et de la version souhaitée. Composer crée alors un fichier nommé "composer.json" à la racine du projet. Ce fichier contient le nom complet des dépendances ainsi que la version ciblée.

composer.json

```
{
  "require": {
    "slim/slim": "~2.6",
    "tuupola/slim-jwt-auth": "~1.0",
    "firebase/php-jwt": "~3.0"
  }
}
```

GIT



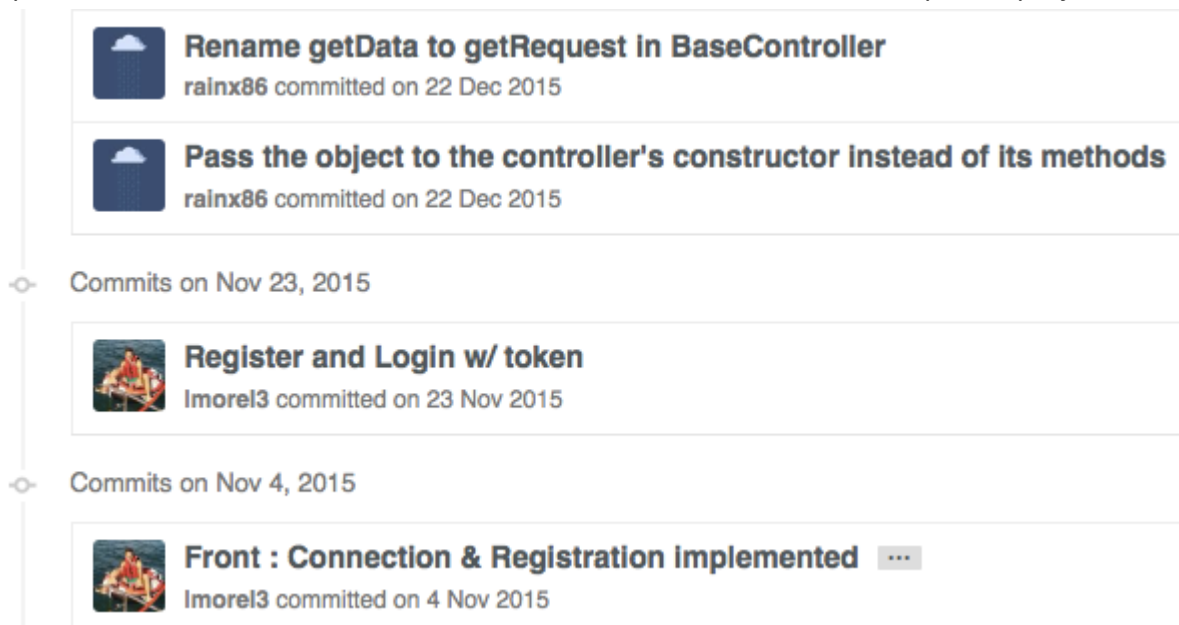
Git est un système de gestion de version. Logiciel libre créé par Linus Torvald, fondateur de Linux, c'est l'un des plus populaires et des plus utilisés.

GITHUB



GitHub est un site qui propose l'hébergement gratuit de dépôts Git. Il existe également un client GitHub, qui est un client Git très simple d'utilisation permettant d'avoir accès aux fichiers gérés par la gestion de version avec une interface graphique. Le dépôt peut aussi être géré en ligne de commande. Notre dépôt est privé et accessible uniquement sur invitation.

GitHub permet de visualiser l'ensemble des modifications effectuées sur le dépôt du projet.



The screenshot shows a list of commits on a GitHub repository. The commits are grouped by date. The first group is for November 22, 2015, with two commits by user 'rainx86'. The second group is for November 23, 2015, with one commit by user 'Imorel3'. The third group is for November 4, 2015, with one commit by user 'Imorel3'. Each commit entry includes a small icon, the commit message, the author's name, and the date.

- Rename getData to getRequest in BaseController**
rainx86 committed on 22 Dec 2015
- Pass the object to the controller's constructor instead of its methods**
rainx86 committed on 22 Dec 2015
- Commits on Nov 23, 2015**
- Register and Login w/ token**
Imorel3 committed on 23 Nov 2015
- Commits on Nov 4, 2015**
- Front : Connection & Registration implemented**
Imorel3 committed on 4 Nov 2015

GitHub dispose aussi d'un système de rapport de bugs dont nous nous sommes servis pour garder une trace des bugs à corriger et des améliorations possibles.



The screenshot shows the GitHub Issues page for a repository. At the top, there is a summary bar indicating 8 Open issues and 4 Closed issues. Below this, there is a list of issues. Each issue entry includes a checkbox, a status icon (green for open, red for closed), the issue title, the issue number, the date it was opened, the author's name, and a label (bug or enhancement).

- Gérer le cas où l'utilisateur ne s'est pas encore entraîné pour la génération des stats** **bug**
#12 opened on 6 Feb by rainx86
- Nettoyer le js et le markup (Indentations, noms des variables, etc)** **enhancement**
#10 opened on 6 Feb by rainx86
- Limiter l'inscription aux étudiants de l'IUT** **enhancement**
#9 opened on 6 Feb by rainx86
- Ne pas envoyer `is_correct` si l'utilisateur est un étudiant** **bug**
#8 opened on 30 Jan by rainx86

PARTIE 2 : REALISATION DU PROJET

Lors du second semestre, nous avons réalisé l'étude du projet composée de l'analyse du projet, de l'établissement d'un cahier des charges, d'un choix des technologies adoptées, de la réalisation d'un modèle conceptuel de données et d'une réflexion quant à la gestion du projet. C'est donc durant les semestres trois et quatre que nous nous sommes attelés à la partie développement. Cependant, certaines hypothèses étaient plus ou moins fausses et nécessitaient d'être corrigées. Aussi, de nombreuses fonctionnalités ont été rajoutées au cours des diverses réunions de projets.

REPARTITION DES TACHES

Les tâches ont été réparties en fonction des compétences des membres, pour plus d'efficacité. Louis a alors pris en charge la partie API, aidé par Vahic. Sébastien s'est quant à lui occupé de la base de données et Laurent a développé le client.

Cependant, tous les membres ont pu contribuer, de près ou de loin, à toutes les composantes du projet grâce à un bon esprit d'entraide.

ARCHITECTURE DE L'APPLICATION

BACK-END

API

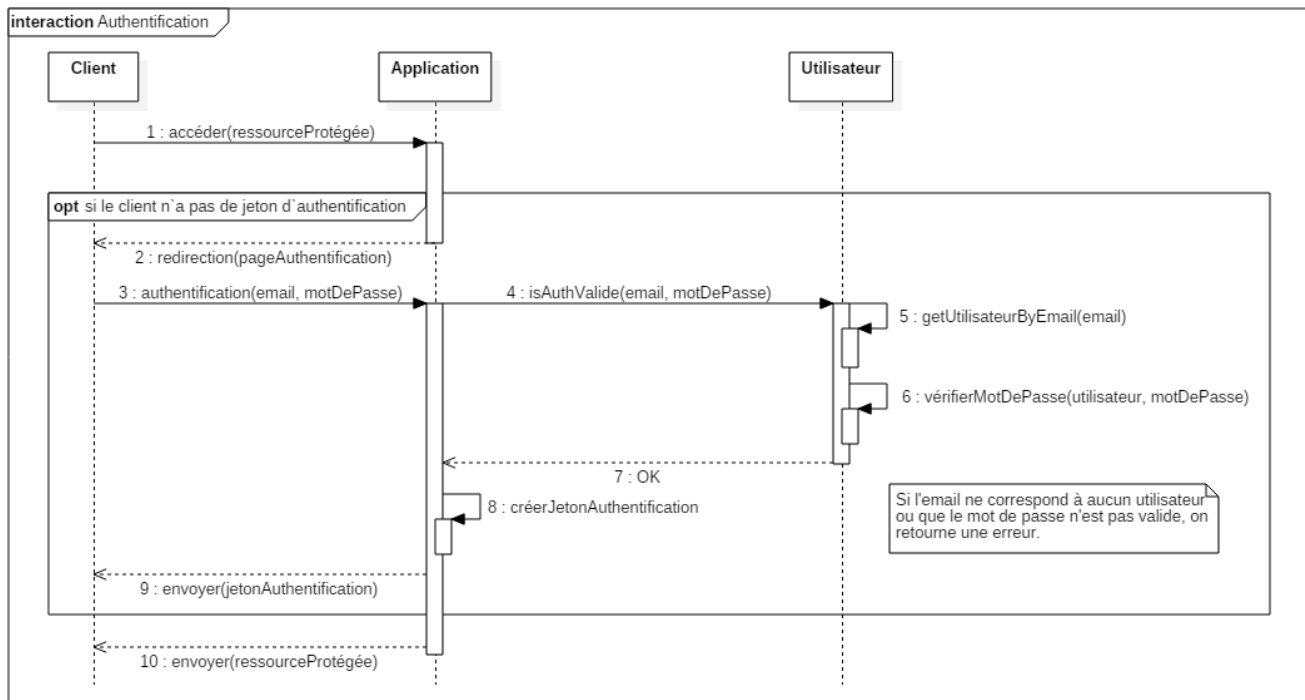
L'application respecte l'architecture MVC : Modèle - Vue - Contrôleur. Dans cette architecture, le contrôleur fait appel au modèle pour mettre à jour la vue, la partie visible. Cette architecture permet une bonne adaptabilité et un code très clair et bien organisé. Ainsi, toutes les fonctions nécessaires au bon fonctionnement de l'application sont clairement séparées de ce qui gère la partie visible de l'application.

Afin que le développement soit le plus simplifié possible et que le code soit maintenable, nous avons décidé de baser le code de l'Application Programming Interface (API) de notre application sur le framework PHP Slim. Nous l'avons choisi car il est destiné à développer des APIs et il est simple d'utilisation.

AUTHENTIFICATION (*ETUDE D'UN POINT TECHNIQUE*)

L'authentification utilisateur a été implémentée avec des JSON Web Tokens (JWT). Lors de l'inscription ou de la connexion d'un utilisateur au site, un jeton est généré par le serveur puis renvoyé au client. Ce jeton est ensuite stocké dans le "local storage" du client et est envoyé avec chaque requête vers une ressource protégée.

Le diagramme de séquence suivant décrit le processus d'authentification :

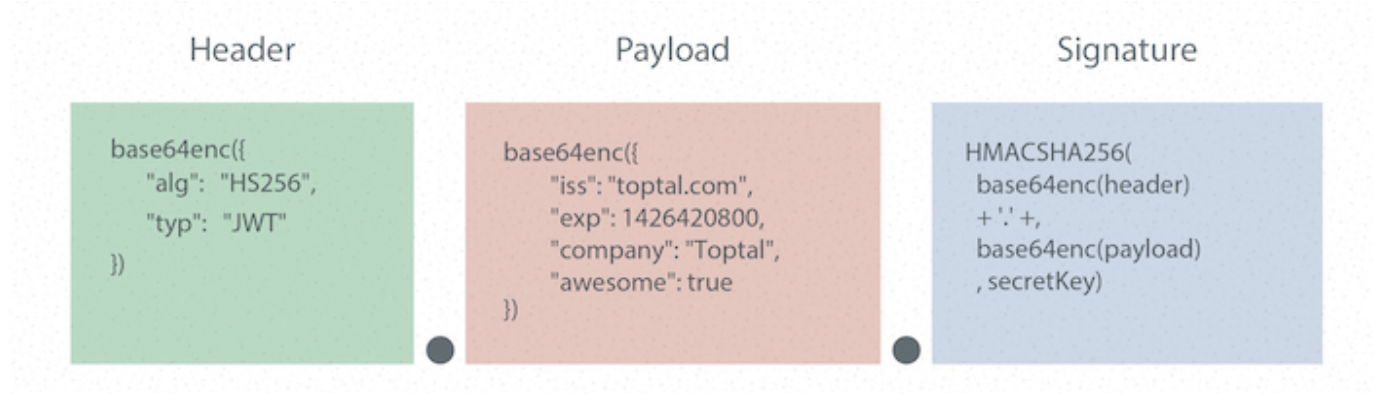


Un JWT est composé de trois parties :

- Une **en-tête** contenant l'identifiant de l'algorithme de signature numérique utilisé et le type de jeton.
- Un **corps** contenant la date à laquelle le jeton a été remis, sa date d'expiration, l'identifiant de l'utilisateur, son nom et son rôle (étudiant ou professeur).
- Une **signature numérique** générée avec l'algorithme HMAC-SHA1. Elle permet au serveur de vérifier que le jeton n'a pas été modifié par un client malicieux. Ainsi, un client ne peut pas se faire passer pour un autre utilisateur en modifiant l'identifiant stocké dans le jeton, ou pour un professeur en modifiant le rôle.

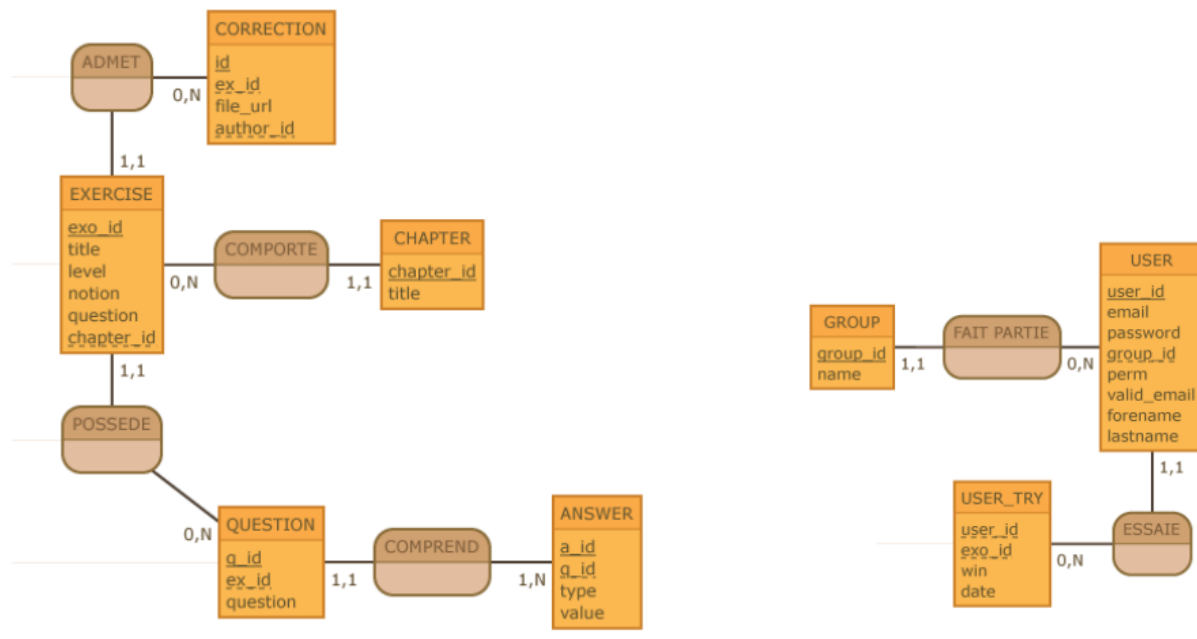
Notez que l'en-tête et le corps du jeton ne sont pas chiffrés. Ainsi, la partie cliente peut utiliser les informations stockées dans le jeton (identifiant et nom de l'utilisateur). Cela signifie aussi qu'un jeton volé peut être utilisé pour usurper l'identité d'un autre utilisateur. Il est donc primordial que les échanges entre un client et le serveur soient chiffrés (à l'aide du protocole HTTPS par exemple).

Enfin, les différentes parties sont encodées en *base64* puis concaténées avec des points. Le schéma suivant résume ce qui a été dit précédemment :

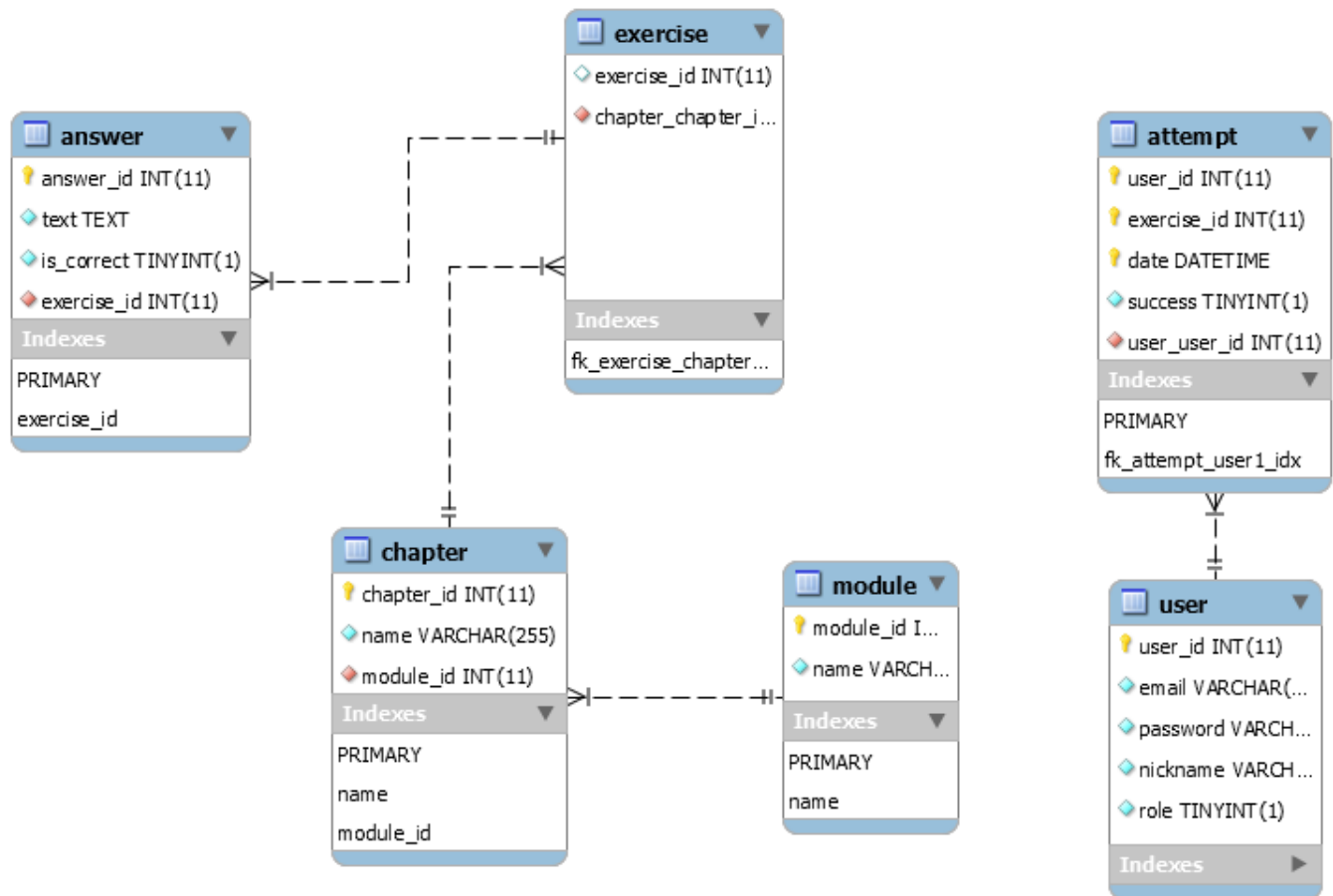


(Source: <https://www.toptal.com/web/cookie-free-authentication-with-json-web-tokens-an-example-in-laravel-and-angularjs>)

BASE DE DONNEES



Voici l'ancien modèle que nous pensions utiliser pour la base de données. Finalement nous avons retiré la table *question* pour insérer la *question* dans la table *exercice*. Nous avons également stocké la correction de l'exercice dans la réponse à ce dernier.



La base de données est désormais composée de 6 tables liées les unes aux autres.

La table **answer** sert à stocker les réponses aux exercices. Elle contient les champs suivant :

- `answer_id` : Identifiant de la réponse
- `text` : Texte de la réponse
- `is_correct` : Un booléen pour indiquer si cette réponse est correcte
- `exercise_id` : référence sur l'identifiant de l'exercice correspondant

La table **exercise** sert à stocker les exercices. Elle contient les champs suivant :

- `exercise_id` : Identifiant de l'exercice
- `chapter_id` : Référence sur l'identifiant du chapitre concerné.

La table **attempt** est utile pour les statistiques. Elle enregistre les essais des étudiants aux exercices et leur réussite ou non dans la base de données. Elle contient les champs suivants :

- user_id : Identifiant de l'utilisateur qui fait l'essai
- exercice_id : Identifiant de l'exercice concerné par l'essai
- date : Date de l'essai
- succes : booléen indiquant ou non le succès à l'exercice
- user_user_id : Référence sur l'utilisateur.

La table **chapter** stocke les noms de chapitres. Elle contient les champs suivants :

- chapter_id: Identifiant du chapitre
- name : Nom du chapitre
- module_id : Référence sur l'identifiant du module contenant le chapitre

La table **module** comprend :

- module_id : Identifiant du module
- name : nom du module

La table **user** stocke les utilisateurs enregistrés de l'application. Elle contient les champs suivants :

- user_id : Identifiant de l'utilisateur
- email : Email de l'utilisateur
- password : Mot de passe de l'utilisateur

DEPENDANCES

L'API dépend des librairies suivantes :

- **Slim** : un micro-framework PHP facilitant le développement d'API, l'accès aux paramètres des requêtes HTTP et la génération de réponses HTTP.
- **slim-jwt-auth** : une extension de Slim facilitant la mise en place d'un système d'authentification utilisateur à l'aide de JSON Web Tokens.

DOCUMENTATION

Nous avons écrit une documentation de l'API, en **Markdown**, décrivant le format des réponses, le processus d'authentification, ainsi que l'accès aux différentes ressources (route, verbe HTTP à utiliser, paramètres à envoyer). Cette documentation permet à n'importe qui de développer son propre client pour notre service. Voici une section de la documentation :

Ressources

Utilisateurs

Connexion

Connecter l'utilisateur au site et obtenir un jeton d'authentification.

Méthode	Route
POST	/api/user/login

Paramètres JSON :

Nom	Type	Description
email	string	L'adresse e-mail de l'utilisateur
password	string	Le mot de passe de l'utilisateur

Exemple :

```
{
  "email": "test@example.com",
  "password": "test"
}
```

Retour : en cas de succès, `data` contient le jeton d'authentification.

TESTS

Nous avons réalisé des tests unitaires en Python à l'aide des librairies ***requests*** (client HTTP) et ***unittest*** (mise en place de tests unitaires). Ces tests nous ont permis de vérifier le bon fonctionnement de l'API de manière automatique après chaque modification, dans un souci de maintenabilité.

FRONT-END

La partie visible par l'utilisateur est gérée grâce **AngularJS** qui intègre un système de "**templating**" qui permet de découper le site en plusieurs parties (cf. Partie 2 : Réalisation du projet, Interface utilisateur). AngularJS nécessite la mise en place d'une API pour accéder aux données. Cette API est donc développée à l'aide de Slim Framework.

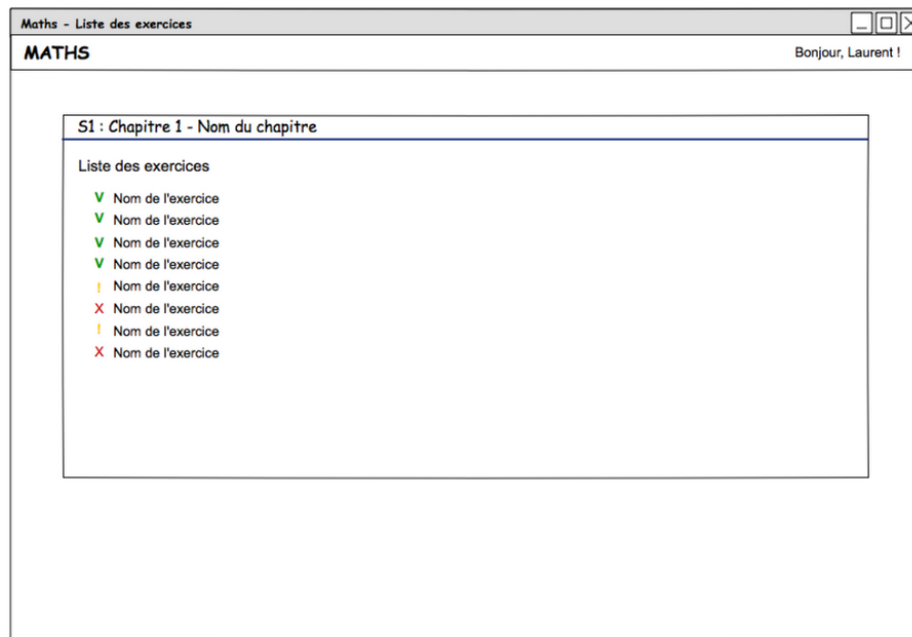
Les fichiers correspondant sont présents dans le dossier **client** à la racine du projet.

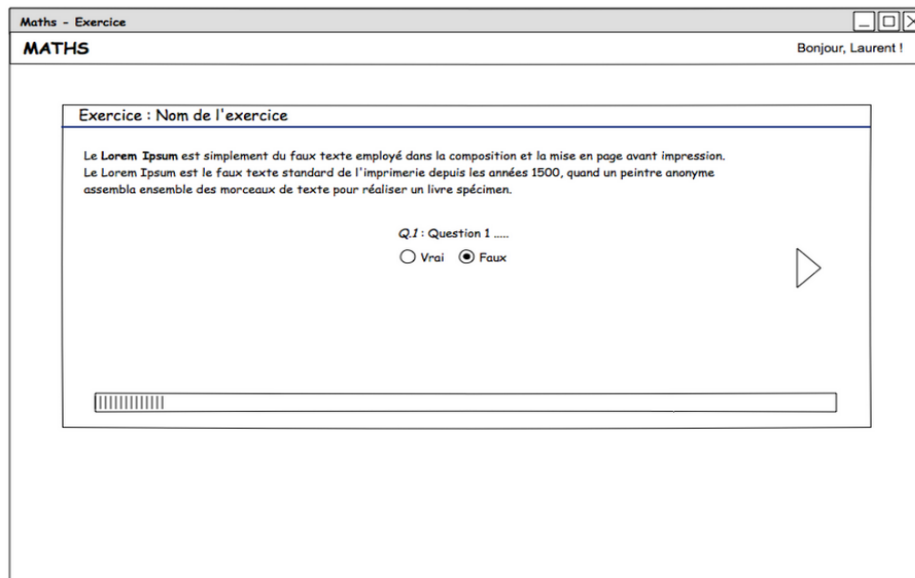
On y retrouve les dossiers suivants :

- **app** : les divers scripts Javascript
- **assets** : les styles CSS, les fonts, les images et les librairies Javascript externes
- **partials** : les templates au format HTML inclus dans l'application

PROTOTYPAGE DE L'INTERFACE

Nous avons tout d'abord identifié des fonctionnalités clés pour l'application. De ce fait, nous avons travaillé sur les différents écrans de l'interface utilisateur à l'aide du logiciel Open Source et gratuit Pencil.





A partir de ces modèles, nous avons pu commencer à designer l'interface utilisateur en nous basant sur le framework **Bootstrap** qui fournit des bases HTML et CSS pour réaliser une application web.

Nous avons également profité de l'avantage non négligeable d'AngularJS, nous permettant de réaliser une application qui conserve une structure principale et inclut dynamiquement le corps de la page (des templates), comme le montre le schéma ci-dessous.



La partie restant la même (tout en possédant du contenu dynamique !) est contenue dans le fichier à la racine du dossier `partials` (correspondant aux couleurs rouge et verte). Le reste de l'application est chargée dynamiquement dans la partie bleue (ng-view est la directive correspondante, cf. Découpage de la partie client, `directives.js`)

DECOUPAGE DE LA PARTIE CLIENT

Le dossier **app** contient donc les fichiers gérant l'application côté client. On y retrouve les fichiers suivants.

APP.JS

Gère la **configuration de l'application** et des diverses **dépendances** utilisées. Nous avons également choisi d'y ajouter certaines **fonctions personnelles**, réutilisées dans d'autres fichiers.

Parmi les configurations, il y a la gestion des routes (`$routeProvider`). Il est alors simple de spécifier quel contrôleur et quel template charger lorsque l'on se trouve à la racine du site.

Par exemple, si l'on veut charger le template "home_guest" et le contrôleur "MainCtrl" lorsque l'utilisateur est à la racine :

```
when('/', {  
  templateUrl: 'partials/home_guest.html',  
  controller: 'MainCtrl'  
})
```

Ces routes sont chaînées entre elles, grâce à l'opérateur `":"` : `$routeProvider.route_A.route_B`;

Par ailleurs c'est là que sont gérées les erreurs des requêtes effectuées à l'API.

CONTROLLERS.JS

Contient les divers **contrôleurs** utilisés pour ordonnancer l'application. Un contrôleur gère une fonctionnalité de l'application, comme par exemple la page d'accueil (HomeController).

Il est défini comme suit :

```
.controller('FeatureCtrl', function ($rootScope, $scope, $location, Flash) { /* corps du contrôleur */ })
```

Nous avons choisi de suffixer les contrôleurs par "Ctrl" pour plus de clarté. Les paramètres de la fonction anonyme du contrôleur correspondent aux dépendances qu'il utilise.

DIRECTIVES.JS

Les diverses **directives** utilisées. Une directive permet de modifier le DOM de la page.

Par exemple nous disposons de la directive “*hideWhenConnected*” qui, ajoutée à une balise HTML, cache cette dernière si l'utilisateur est connecté. Ci-après un exemple d'utilisation :

```
<div hide-when-connected>Contenu masquée lorsque l'utilisateur est connecté</div>
```

SERVICES.JS

Contient les **factories**, comparable au “Model” de la couche MVC, qui gèrent les appels à l'API. Elles sont alors utilisées comme dépendances des contrôleurs.

Les requêtes effectuées vers l'API sont gérées de manière asynchrone et utilise le système de promesses. De ce fait, le contrôleur exécute des instructions dans la fonction *then*, exécutée une fois la requête terminée. Ci-dessous un exemple pour récupérer la liste des modules :

```
Module.query().then(function (modules) { /* modules contient un tableau avec les divers modules */ });
```

L'API retourne uniquement des données au format JSON, automatiquement converties en objets par AngularJS.

DEPENDANCES

Tout comme l'API, la partie client dépend de librairies externes. Nous utilisons les dépendances suivantes :

- **angular.min.js** : AngularJS
- **angular-flash.min.js** : dépendance d'AngularJS permettant d'afficher des messages éphémères
- **angular-route.min.js** : dépendance d'AngularJS permettant la gestion des routes
- **angular-chart.min.js** : dépendance d'AngularJS permettant la gestion des graphiques de Chart.js
- **Chart.min.js** : librairie Javascript permettant la création de graphiques

FONCTIONNALITES

PAGE D'ACCUEIL

VISITEUR

La page d'accueil affiche un message de bienvenue et propose à l'étudiant, soit de se connecter, soit de s'inscrire. L'inscription sera filtrée pour n'accepter que les adresses du type "prenom.nom@etu.univ-lyon1.fr" (ou seulement @univ-lyon1.fr pour les professeurs). Ainsi, le prénom et le nom de l'étudiant seront facilement récupérés et pourront être utilisés par l'application.

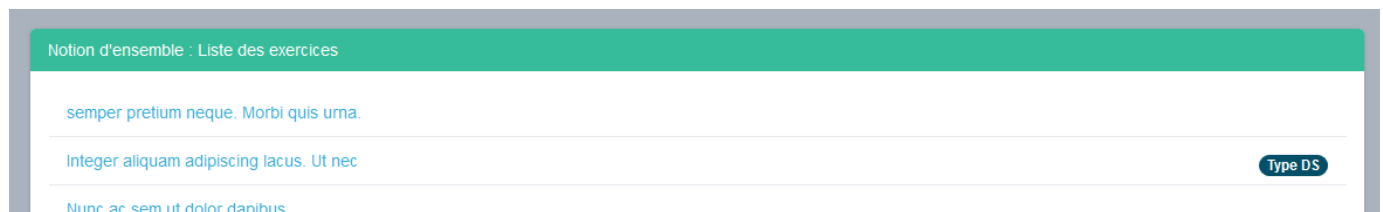
C'est la seule page visible par une personne non connectée.

UTILISATEUR

L'utilisateur connecté voit la page d'accueil différemment. En effet il a directement accès à la liste des modules et des chapitres, moyennant un clic sur le module. Il peut également visualiser son avancement à l'aide de barre de progression module par module. L'administrateur peut accéder directement à la gestion des modules.

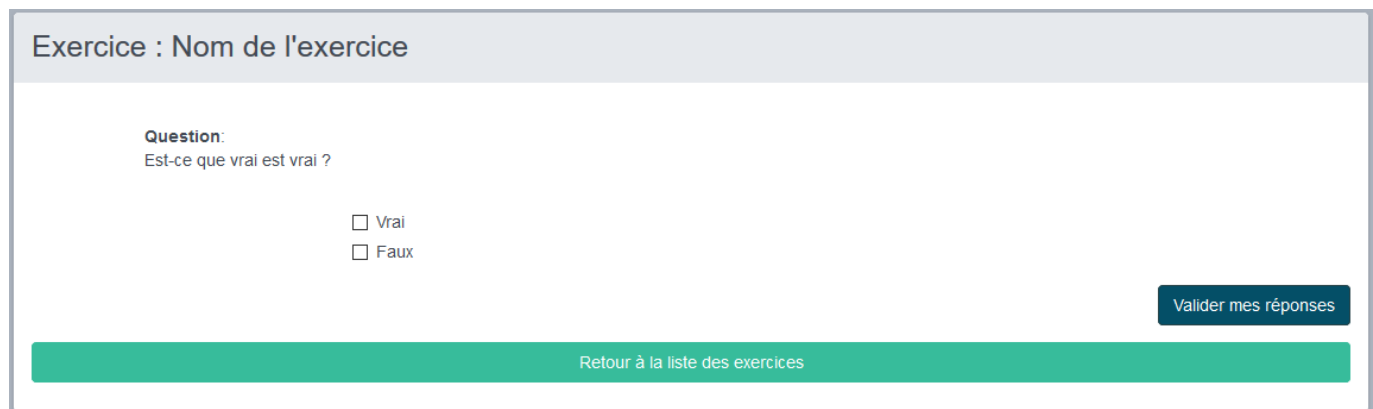
L'autre point clé réside dans la possibilité de générer une série d'exercices, en fonction de chapitres choisis dans un module. L'utilisateur sélectionne donc les chapitres et entre le nombre d'exercices - maximal - voulus. Une série d'exercice est alors lancée, cela revient à faire les exercices un par un.

LISTE D'EXERCICES



Lors d'un clic sur un chapitre, la liste des exercices correspondant est proposée à l'utilisateur. Il peut alors choisir de lancer un exercice. Certains sont marqués "type DS" pour indiquer à l'utilisateur que cet exercice peut être utile pour préparer l'examen qui contiendra des exercices façonnés comme eux.

RESOLUTION D'UN EXERCICE



Les exercices ont donc deux formes possibles : le QCM ou la réponse sous forme de valeur avec un intervalle. Lorsque l'utilisateur soumet sa réponse, son essai est sauvegardé en base de données pour les statistiques. Un exercice comprend : un titre et un énoncé au format LaTeX.

ADMINISTRATION

GESTION DES MODULES

Liste des modules

Modules

- Mathématiques discrètes
- Algèbre linéaire
- Graphes et langages
- Analyse et méthodes numériques
- Probabilités et statistiques
- Modélisations mathématiques

Chapitres

Notion d'ensemble	6 exo(s)
Numération	5 exo(s)
Algèbre de Boole	0 exo(s)
Fonctions booléennes	1 exo(s)
Calcul propositionnel	2 exo(s)
Prédicats	5 exo(s)
Algèbre de Boole des ensembles	1 exo(s)
Sommes, suites usuelles et récurrence	2 exo(s)
Démonstration	1 exo(s)

Exercices

semper pretium neque. Morbi quis urna.	 
DS Integer aliquam adipiscing lacus. Ut nec	 
Nunc ac sem ut dolor dapibus	 
turpis. Aliquam adipiscing lobortis risus. In	 
magna tellus faucibus leo, in lobortis	 
DS quam a felis ullamcorper viverra. Maecenas	 

Ajouter un exercice

Cette section propose à l'administrateur la liste des modules et chapitres correspondant, ainsi que les différents exercices dans le but de les modifier, d'en ajouter ou de les supprimer.

GESTION DES UTILISATEURS

Liste des utilisateurs

apitest@maths.dev

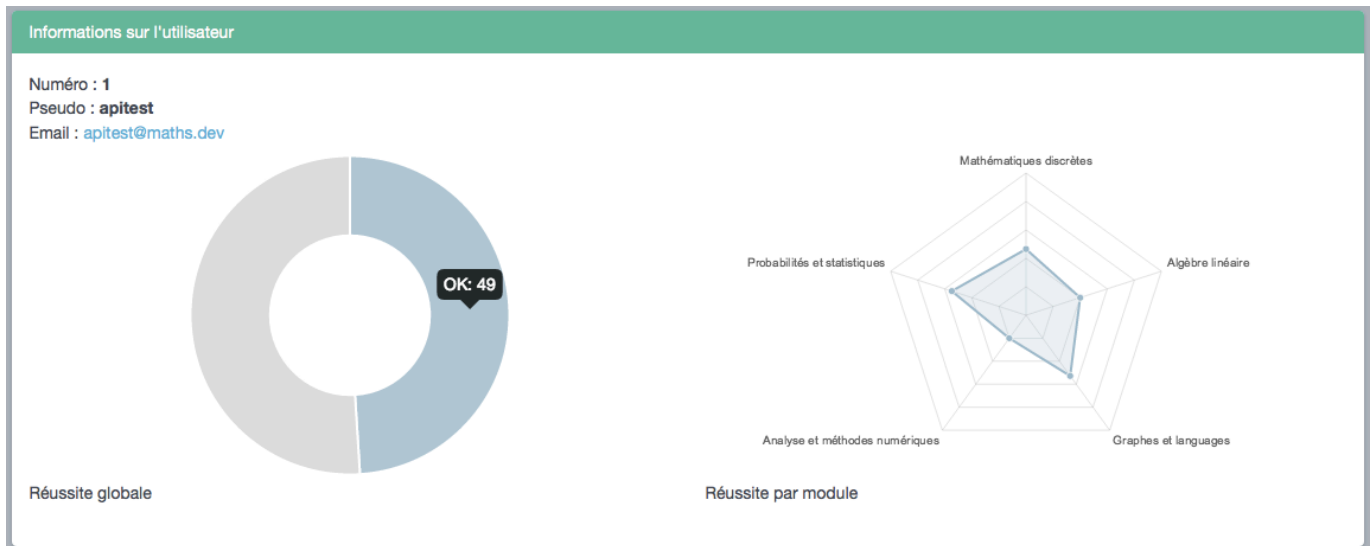
test@test.com

test2@test.com

laurent.morel30@gmail.com

Dans cette partie, l'administrateur peut rechercher un utilisateur et voir ses informations ainsi que ses diverses statistiques. C'est fonctionnalité la plus importante pour les professeurs afin d'avoir un suivi personnalisé, pour chaque étudiant.

STATISTIQUES UTILISATEUR



Cette section permet à un professeur de visualiser les statistiques d'un utilisateur (pourcentage de réussite globale et par module).

PARTIE 3 : BILAN INDIVIDUEL

LOUIS

Ce module a été pour moi l'occasion de travailler en équipe sur un projet intéressant.

J'ai réalisé la partie serveur (API, base de données) avec l'aide de Sébastien, et j'ai épaulé Laurent pour la réalisation de la partie cliente. Ce projet m'a permis d'approfondir mes connaissances en JavaScript et de découvrir le framework Angular.js.

Portant un intérêt certain pour la sécurité informatique et la rétro-ingénierie, je me suis assuré du respect des bonnes pratiques en matière de sécurité, notamment en ce qui concerne l'authentification utilisateur et le stockage des mots de passe en base de donnée.

M. Jaloux a très bien joué le rôle d'un client fictif, en prenant régulièrement des rendez-vous avec nous, en nous donnant ses avis et des pistes d'amélioration.

SEBASTIEN

Travailler sur ce projet m'a permis d'utiliser un peu plus les compétences acquises au cours du DUT. Travaillant plus sur la structure de la base de donnée et le back-end avec Louis, j'ai pu voir comment certaines parties du projet devaient être approfondies et certaines fois repensées. J'ai cependant été légèrement trop en retrait sur ce projet dû à des compétences en développement inférieure à celle du groupe. Cependant Louis m'a permis de toujours me maintenir à jour et de comprendre très rapidement les avancées du projet. Il était donc vraiment le bienvenu dans ce dernier.

M. Jaloux fut d'une aide précieuse, avec ses nombreuses propositions et son point de vue. Il m'a ainsi rapidement amené à repenser quelque peu la structure de base de données déjà envisagée.

VAHIC

Ce projet m’a apporté des compléments que je pense importants aux connaissances enseignées par l’IUT. Pour commencer, il m’a bien évidemment permis de travailler en groupe, dans la continuité des projets des semestres précédents.

Le fait de travailler sur plusieurs semestres à un même projet permet de dépasser un peu la dimension scolaire que peuvent avoir d’autres projets moins conséquents menés dans d’autres matières : ce projet est bien plus proche de ce que nous pourrions connaître en entreprise.

De plus, dans les 2 derniers semestres, M. Jaloux a vraiment endossé une casquette de client fictif, avec des réunions régulières pour faire le point sur le projet et confirmer que nous répondions aux attentes, quitte à parfois remanier le projet pour prioriser un objectif ou abandonner une fonctionnalité inutile.

En dehors de cette expérience intéressante, ce projet m’a permis de travailler avec 2 codeurs beaucoup plus expérimentés (Laurent et Louis) et d’observer leur code et leur manière de gérer un projet (notamment l’utilisation d’un gestionnaire de version, habitude que je n’avais pas auparavant).

En conclusion, ce projet aura été une très bonne expérience pour moi, positive sur le plan humain comme sur celui des compétences acquises.

LAURENT

Ce module fût pour moi l’occasion de travailler sur un projet étalé sur plusieurs mois, impliquant une équipe ainsi qu’un client “factice” : un premier pas vers le monde professionnel. Bien qu’il y ait eu quelques problèmes de communication interne, le travail en équipe m’a grandement plu. Notre tuteur, M. Jaloux, nous a également grandement guidé et nous a donné de nombreuses idées, aide non négligeable. Le sujet était par ailleurs intéressant et motivant car la mise en service d’une application de ce type est envisageable et elle pourrait être utilisée par un grand nombre d’étudiants.

Le projet m’aura permis d’améliorer mes bases en conception d’applications ainsi qu’en Javascript. De plus, j’ai pu acquérir de nombreuses compétences sur le framework AngularJS, actuellement en vogue en entreprise. De ce fait, nous avons pu développer une structure souple pour envisager de possibles améliorations ainsi qu’un code clair et maintenable.

J’ai également apprécié la venue de Louis durant le semestre 3 qui m’a aidé à concevoir au mieux l’architecture de l’application et qui m’a donné de nombreuses informations en matière de sécurité.

CONCLUSION

Ce projet a été une très bonne expérience pour tous les membres du groupe. Chacun a pu améliorer ses compétences à son rythme et les plus expérimentés ont fait profiter de leur savoir aux autres.

Il nous a permis d'enrichir nos connaissances et d'appliquer les compétences acquises durant ces deux années d'IUT en ayant un véritable objectif.

De plus, il nous a montré, avec les nombreux autres projets réalisés lors du semestre quatre, combien il devient difficile de s'organiser convenablement. Les délais ne nous ont pas posé réellement problème, cependant arriver à réunir les membres du groupe sur un même créneau horaire pour les parties devant être discutées en commun s'est révélé parfois compliqué.

Ce projet était réaliste et pourra tout à fait s'implanter dans les CV des membres souhaitant s'orienter dans le domaine du développement Web. Nous sommes dans l'ensemble satisfaits du résultat final et fier d'avoir pu mener à bien ce projet.