QuizMaker

Kevin Low Pein

Ghassen Belkhir

Junior Mabika

Tony Vo

Projet Long

Octobre 2018 à Mars 2019

Professeur référent : Mme. Tan Sovanna

Établissement universitaire : Université Paris-Est Créteil

Table des matières

Table des figures

Chapitre 1

Présentation du projet « Rédigé par Tony VO »

1.1 Introduction

Dans le monde entier, le numérique est un facteur de développement colossal et tous les secteurs en sont touchés notamment le secteur bancaire, sanitaire et particulièrement celui de l'enseignement.

Aujourd'hui, nous constatons que les étudiants accèdent à leur emploi du temps en ligne ainsi qu'à leur cours qui peut être mis à jour facilement et en temps réel. L'enseignant dispense les cours aux étudiants en effectuant de simple clic grâce à certaines plateformes qui lui permettent d'interagir avec eux, à travers des annonces ou bien à travers des forums. Non seulement cette évolution a changé la manière d'enseigner mais elle a aussi dilaté la communication hors des murs de salle du cours.

Malgré tous ce que nous apporte la technologie dans le secteur de l'enseignement, nous pensons que ce domaine a encore du progrès à suivre : les progrès sont encore loin d'exploiter les nouvelles puissances de calcul et la fiabilité de nos ordinateurs ainsi que nos smartphones. C'est dans ce sens que nous souhaitons que le numérique donne aux enseignants la possibilité d'oublier les papiers classiques et ainsi de passer outre la correction des copies, via des outils numériques dédiés.

1.2 Contexte du projet

Afin de faciliter le travail d'un enseignant en exploitant les nouvelles technologies et les ressources matérielles et logicielles chez les étudiants, ce projet vise à mettre en disposition aux enseignants un outil qui permet de faire des évaluations en temps réel.

Il s'agit d'un système de QCM en ligne intelligent permettant d'un côté, aux enseignants de faire des sondages ou bien des examens (les deux se composant de QCM), et de l'autre côté, les étudiants qui accèdent à ce QCM, répondent à travers leur ordinateur ou leur smartphone.

Dans ce système on trouve deux acteurs principaux : l'enseignant et l'étudiant dont chacun a ses propres fonctionnalités qui sont comme suit :

Les fonctionnalités proposées pour l'enseignant :

- Créer et modifier un QCM.
- Créer une classe.
- Créer, modifier et supprimer un examen ou un sondage.
- Corriger le QCM automatiquement.
- Collecter les résultats.
- Générer l'examen en PDF.
- Faire des statistiques sur les sondages.

Les fonctionnalités proposées pour l'étudiant :

- Consulter ses dernières soumissions.
- Rejoindre une classe.
- Accéder aux examens ou aux sondages.
- Répondre aux QCM.
- Recevoir sa note de l'examen en temps réel.

1.3 Architecture

Pour ce système nous proposons l'architecture 3-tiers :

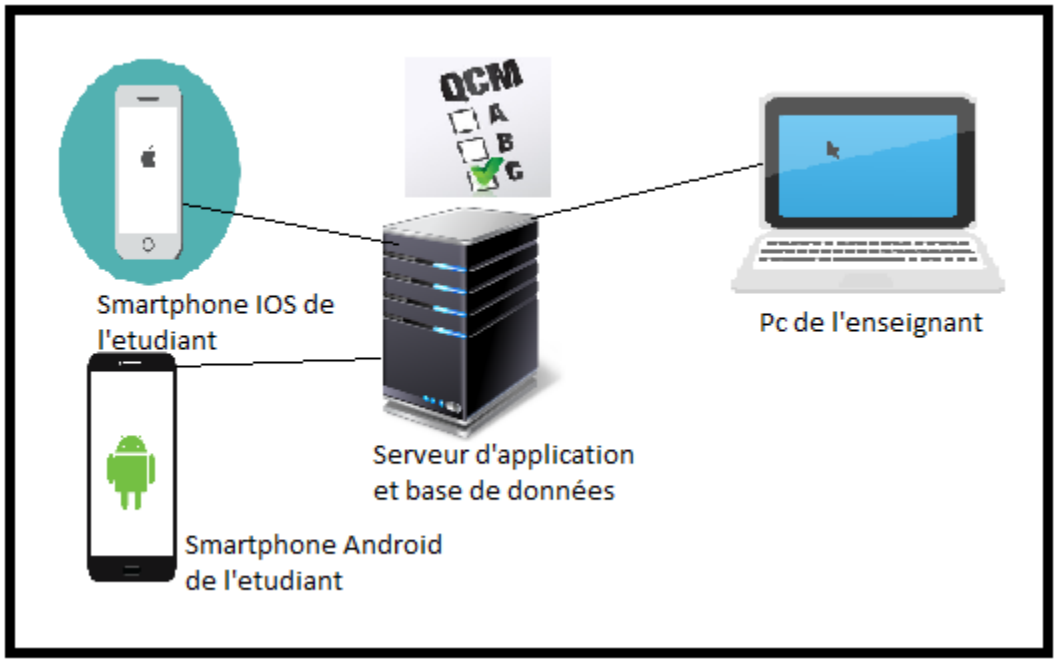

FIGURE 1.1 – Architecture 3-tiers

Comme le montre cette figure, nous trouvons : l'application Web qui se trouve sur l'ordinateur de l'enseignant, avec un serveur Web qui interroge la base de données. Il offre les services Web à l'application Web ou bien à l'application mobile qui sera utilisée par les étudiants. La communication avec la base se passe par l'intermédiaire de ces services Web.

1.4 Les technologies utilisées

Ce projet est réalisé en utilisant le Mean Stack ("MongoDB, Express, Angular, Node.js ") qui est un ensemble de technologies qui forment un environnement complet pour une application :

- **MongoDB** : C'est un système de gestion de base de données orientée documents, répartissable sur un nombre quelconque d'ordinateurs et ne nécessitant pas de schéma prédéfini des données.
- **Express** : C'est une infrastructure d'application Web back-end s'exécutant sur Node.js.
- **Angular** : C'est un Framework de développement qui permet de créer des applications web dynamiques et développé par Google et la communauté AngularJS. Il est

une réécriture d'Angulars.JS.

— **Node.js** : C'est un environnement bas niveau qui exécute du JavaScript côté serveur. Il s'occupe de recevoir les requêtes HTTP (Hypertext Transfer Protocol), de les traiter et de renvoyer un résultat.

Et pour développer l'application mobile nous avons utilisé **Apache Cordova** qui est un Framework de développement mobile open-source. Il permet d'utiliser les technologies Web standard - HTML5 (Hypertext Markup Language) , CSS3 (Cascading Style Sheets) et JavaScript pour le développement multi-plateforme.

Chapitre 2

Présentation du travail réalisé

2.1 Conception

La première tâche de notre équipe était de déterminer les fonctionnalités à réaliser, ensuite tous les membres de l'équipe ont participé à la conception du notre système.
Notre diagramme était en évolution pendant toute la phase de développement jusqu'à l'arrivée à ce modèle final que nous présentons dans la figure suivante :

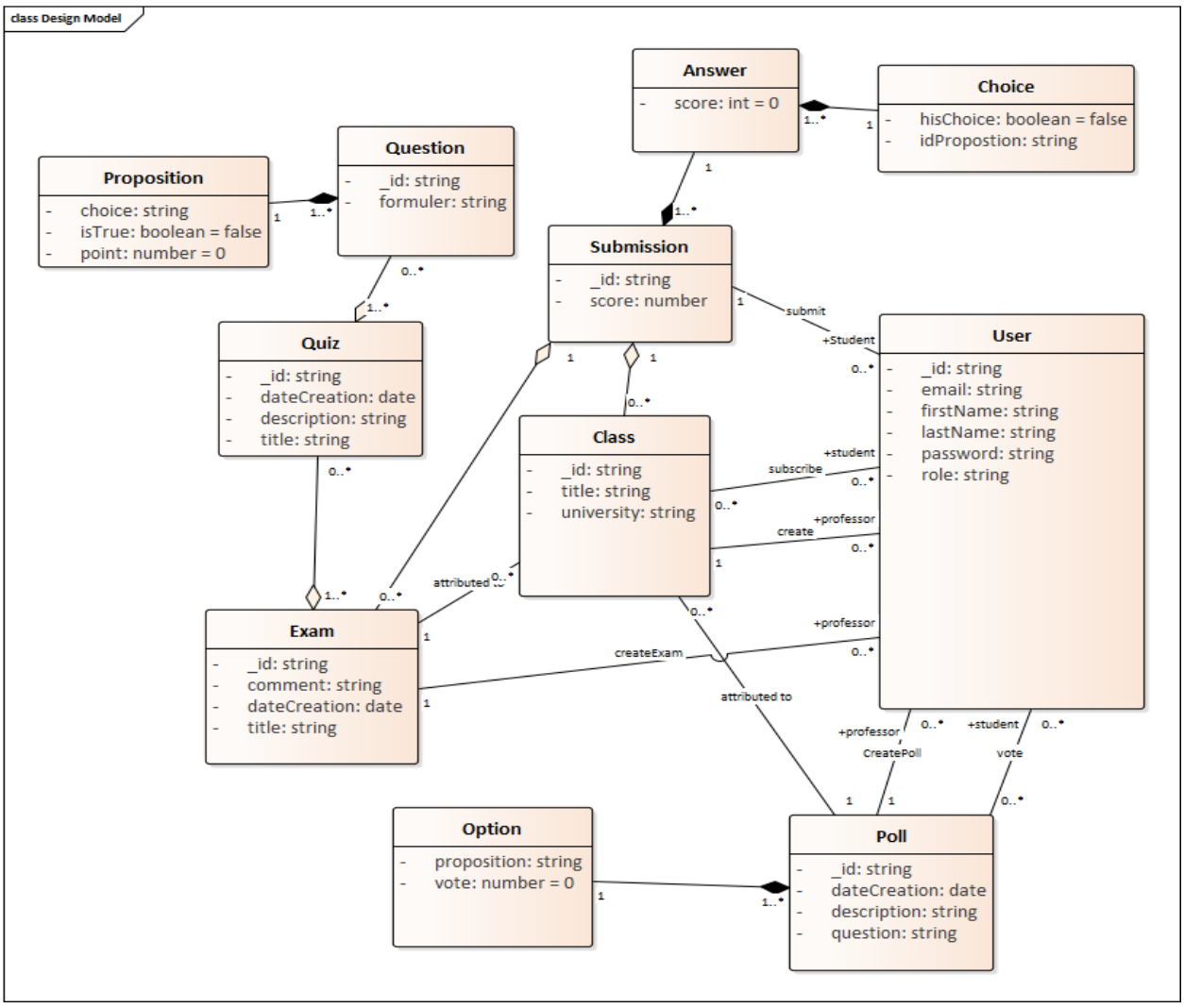

FIGURE 2.1 – Diagramme de classe de notre projet

Description textuelle :

Nous avons réalisé le diagramme de classe avec les règles suivantes :

Un enseignant et un étudiant sont des utilisateurs représentés par un nom, un prénom, un email, un mot de passe et un rôle (student / professor).

Un enseignant peut créer plusieurs classes auxquelles chaque étudiant peut s'inscrire.

Un questionnaire se compose de questions qui elles-même sont composées de propositions.

Dans un examen, il peut y avoir plusieurs sujets (questionnaires).

L'enseignant peut créer des examens ainsi que les questionnaires dont il fait partie.

Un sondage est composé d'une question et d'une liste des propositions.
Les étudiants passent un examen en créant une soumission qui est liée à un examen et qui contient une liste de réponse.
Tous les examens, sondages, et les soumissions sont attribués à une seule classe.

2.2 Partie Backend

« Rédigé et réalisé par Ghassen Belkhir »

2.2.1 Serveur

Après avoir préparé notre modèle et l'architecture du système, nous avons commencé à construire le serveur de l'application qui représente la partie Backend et qui interroge la base de données pour traiter les requêtes reçues du client pour ensuite lui envoyer une réponse.
Afin de mettre en place ce serveur, nous avons dû définir la façon permettant de le communiquer, c'est pour quoi nous avons développé des Api rest pour le système.

Ces Api sont construits en utilisant les Frameworks Nodejs et Express qui fournissent des bibliothèques permettant de construire des routes et gérer des requêtes à travers les méthodes GET POST PUT et DELETE. Cela permet de gérer séparément les demandes à différents chemins d'URL pour ensuite créer dynamiquement la réponse.
Chaque URL est attribuée à une méthode qui traite les différents types de requêtes :

FIGURE 2.2 – Portion du code (fichier server/routes/index.router.js)

Cette route (/newClass) reçoit la requête d'un utilisateur qui est du type POST pour demander l'insertion d'une nouvelle classe, donc la méthode CreateClass du contrôleur va traiter cette requête.
On illustre le fonctionnement de cette méthode avec le code suivant :

```
module.exports.CreateClass = (req, res, next) => {
    var myClass = new Class();
    myClass.title = req.body.title;
    myClass.university = req.body.university;
    myClass.professorCreated = req._id;
    myClass.arrayOfExams = [];
    myClass.arrayOfStudents = [];
    myClass.arrayOfPolls = [];
    myClass.save((err, doc) => {
        if (!err)
            res.status(200).send(doc);
        else {
            return res.status(433).send(err);
        }

    });
}
```

FIGURE 2.3 – Portion du code (fichier server/controllers/class.controller.js)

À travers le paramètre req qui contient l'objet envoyé, cette méthode va interroger la base pour insérer la classe, et elle renvoie à la fin une réponse (res) au client pour l'informer si sa demande est effectuée avec succès ou non.
Pour plus de détails veuillez consulter la documentation des Apis dans notre rendu.

2.2.2 Base de données

Notre base de données est du type Mongodb qui appartient au système NoSql orienté document et qui stockent des données semi-structurées à travers les concepts de collection et document.

MongoDb facilite la manipulation des bases de données parce qu'il n'impose pas un schéma pour une collection (table en Sql). Et chaque document d'une collection (entrée dans une table) peut avoir des attributs différents, ce qui n'est pas possible dans le cas des bases de données relationnelles.

Au niveau des relations entre les collections, Mongodb nous offre la possibilité de créer non seulement des relations référencées mais il permet de créer des relations via l'approche incorporée, ce qui facilite la création des agrégations fortes en évitant de créer d'autres collections ensuite les lier à travers des clés étrangères.

Afin de communiquer le serveur avec Mongodb nous avons utilisé le Framework Mongoose qui permet la conversion entre les objets du code (Java script) et la représentation de ces objets dans MongoDb.
Mongoose utilise des schémas pour modéliser les données comme suit :

```
const mongoose = require('mongoose'), Schema = mongoose.Schema;
var QuestionSchema = new mongoose.Schema({

    formuler: { type: String, required: true },
    professorCreated: {
        type: Schema.Types.ObjectId,
        ref: 'User',
        required: 'professor can\'t be empty'
    },
    listProposition: [{
        choice: { type: String, required: true },
        isTrue: { type: Boolean, required: true },
        point: { type: Number, required: true }
    }]

});
mongoose.model('Question', QuestionSchema);
```

FIGURE 2.4 – Portion du code (fichier server/models/class.model.js)

Cette figure montre la présentation du modèle de question sur Mongodb, on trouve ici le champ professorCreated qui représente une référence vers la classe user, ainsi le champ listProposition qui contient un tableau d'objet sous forme d'une relation incorporée.

2.2.3 Authentification avec JWT

Afin de sécuriser l'accès et de créer des sessions, nous avons implémenté JSON Web Token (JWT) qui permet aux clients autorisés d'accéder au système avec un jeton délivré après la phase d'authentification.
Pour accéder à une route, on doit inclure un token dans l'en-tête HTTP sur le champ Authorization avec la syntaxe suivante :
Authorization : Bearer <token>

2.2.4 Correction d'un examen

Pour corriger les examens, nous avons implémenté la méthode suivante :

```
var calculScore = (question, answers) => {
    let score = 0;
    if (verifyAllAnswerAreCorrect(question, answers) == true) {
        for (let i = 0; i < question.listProposition.length; i++) {
            if (question.listProposition[i].isTrue == true) {
                score += question.listProposition[i].point;
            }
        }
        answers.correction = "correct";
    }
    else if (verifyAllAnswerAreNotCorrect(question, answers) == false) {
        for (let i = 0; i < question.listProposition.length; i++) {
            if (question.listProposition[i].isTrue == true) {
                if (answers.listProposition[i].hisChoice == true) {
                    score += question.listProposition[i].point;
                }
            }
        }
        answers.correction = "incomplete";
    } else {
        answers.correction = "incorrect";
    }
    return score;
}
```

FIGURE 2.5 – Portion du code (fichier server/controllers/submission.controller.js)

Cette méthode va calculer la note obtenue sur chaque question en vérifiant dans un premier temps si toutes les réponses sont correctes et alors on cherche toutes les réponses qui sont vraies et on y ajoute ses points.
Sinon on teste s'il y a au moins une réponse correcte alors on calcule les points attribués et à la fin si les deux premières conditions ne sont pas validées, alors on constate que toutes les réponses pour cette question sont incorrectes et donc il n'y a pas de points à ajouter.
En appliquant cette méthode à toutes les questions on obtient à la fin la note finale sur l'examen.

2.2.5 Implémentation des services coté application Angular

Dans le but de consommer les Api rest développés sur notre serveur, nous avons intégré sur l'application web des services qui contiennent les opérations nécessaires afin d'envoyer et de récupérer des données à partir d'un serveur. Prenons par exemple le service suivant :

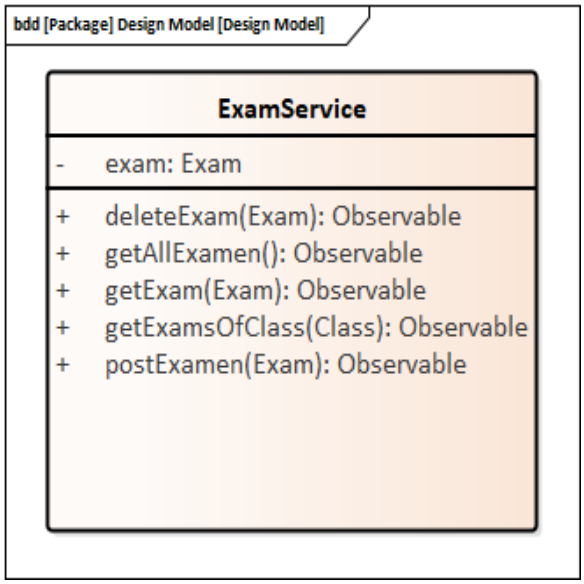

FIGURE 2.6 – Class ExamService

Chaque méthode renvoie un observable qui contient le corps de réponse et qui sera invoqué par la méthode subscribe pour obtenir soit un résultat, soit une erreur :

```
fetchAllExams(){
  this.examenService.getAllExamen().subscribe(
    res => {
      this.myExams = res;
      console.log(this.myExams);
    },
    err => {
      console.log(err);
    }
  );
}
```

FIGURE 2.7 – Class ExamListComponent

2.3 Partie gestion des questionnaires, examens et sondages « Rédigé et réalisé par Kevin Low Pein »

2.3.1 Gestion de questionnaires

Cette partie consiste à réaliser un questionnaire qui est composé d'une multitude de questions, où chaque question elle-même comprend des choix de réponses (style QCM).

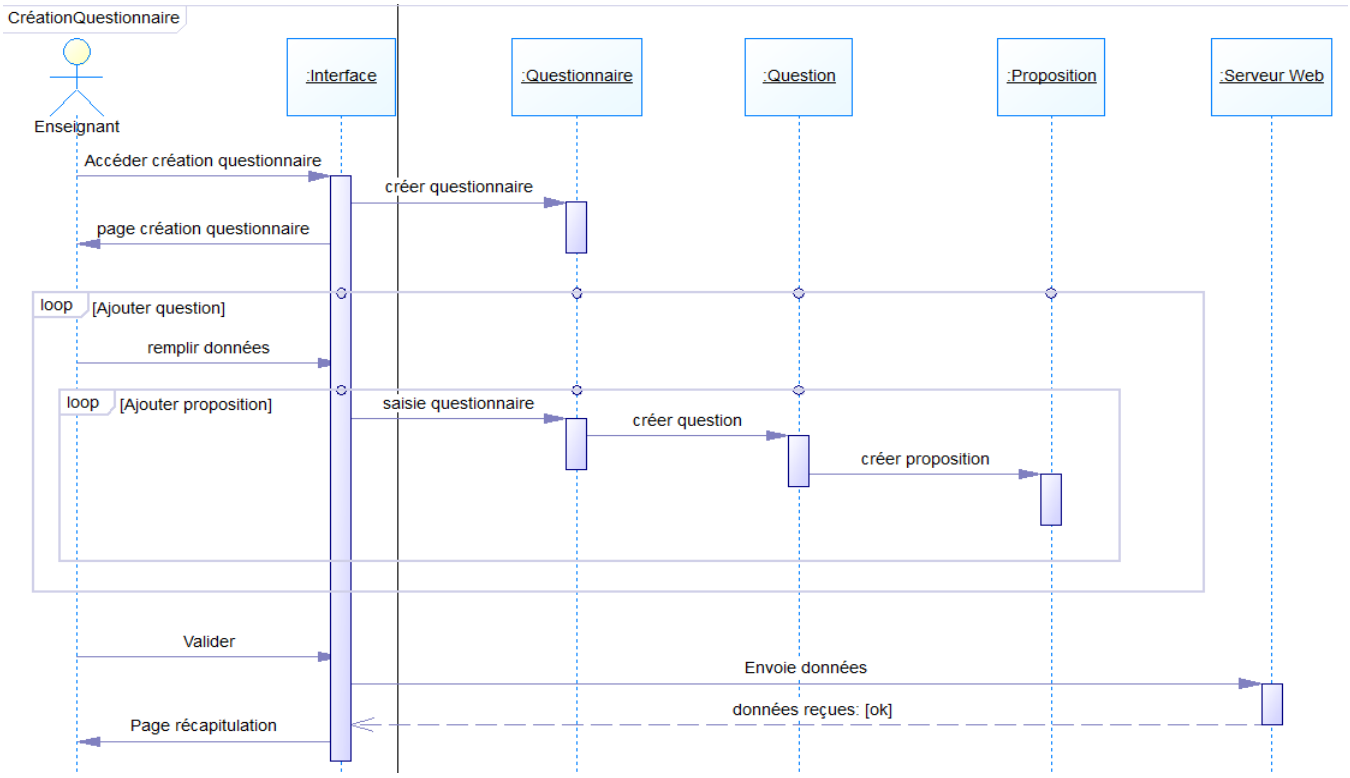

FIGURE 2.8 – Diagramme de séquence – création questionnaire

Le diagramme de séquence ci-dessus montre une interaction homme machine (IHM). Le professeur accède à la création du questionnaire, l'interface renvoie la page correspondante. Ensuite, il remplit les données et il ajoute des questions et propositions. A la fin, il valide et les données sont récoltées puis il sera redirigé vers la page récapitulative.

a. Création du quiz

Cette section est accessible depuis la page « **home** » de l'enseignant. Elle s'affiche en pop-up et contient que deux champs : le titre et la description du quiz elle-même. Le titre

est requis pour que cette création soit acceptée, et une fois créée, elle sera redirigée vers la page de la création des questions. Elle permet d'évaluer les connaissances, similaire aux tests et si elle est belle bien notée. Cette création consiste aux professeurs de suivre l'état de connaissance des étudiants en la matière, pour faire des études sur un sujet donné.

b. Création des questions

Cette section est la création des questions pour le questionnaire. Après voir valider la première étape précédemment, ses données sont envoyées depuis l'URL pour que chaque question soit reliée au quiz. Elle permet bien évidemment de laisser à l'enseignant de créer autant de questions qu'il voudra pour son questionnaire et rajouter autant de propositions pour chaque question. Le professeur doit obligatoirement renseigner tous les champs nécessaires. Une proposition comprends son libellé, une case à cocher pour dire si la réponse est correcte ou non et son point à attribuer. Bien qu'on puisse ajouter des propositions, on peut aussi supprimer un bloc de proposition déjà rempli, de même pour un bloc de question qui contient des propositions. Sachant que si on ne renseigne pas le champ « **point** », il sera par défaut à 0. Pour sauvegarder partiellement sa création, le bouton « **sauvegarder** » permettra cette action.

c. Création de la récapitulation

Cette section renferme un récapitulatif des deux étapes précédentes, c'est-à-dire un résumé du questionnaire. Elle permet à l'enseignant de voir ce qu'il a saisi. Sur cette partie, il pourra constater si tout a été prise en compte depuis le début de la création. Si par erreur de saisi, le professeur pourra soit modifier sa création soit annuler (ce qui effacera la création elle-même et de récréer une autre).

d. Modification du questionnaire

Cette partie consiste à modifier le questionnaire. Elle est accessible soit depuis la récapitulation du questionnaire soit depuis le détail du questionnaire. Le professeur pourra donc revenir à sa création sans perdre les données déjà saisies. Il pourra bien sûr en rajouter autant de questions que de propositions, mais aussi de les supprimer. On peut également modifier le titre et la description du quiz. Une fois que les modifications soient validées, elles seront envoyées à nouveau à la page récapitulative.

e. Génération en PDF

On peut générer le questionnaire en PDF depuis la page récapitulative ou bien depuis le détail du questionnaire. En cliquant sur le bouton « **aperçu** », un pop-up affiche le détail du questionnaire, et en cliquant sur « **générer en PDF** », on peut l'imprimer ou enregistrer en PDF. Dans le paramètre de l'impression, on peut modifier la mise en page, choisir quelle page à imprimer, ou encore dans « plus de paramètres », on peut choisir les imprimantes. Tout cela permettra de faire un test sur papier.

f. Détail du questionnaire

Cette section est similaire à la page de récapitulation mais il n'y a pas de bouton « **valider** ». Il est accessible depuis la page « home » de l'enseignant, dans la section « **Mes derniers questionnaires créés** ». Le détail contient toutes les informations concernant le questionnaire. On peut également le générer en PDF, ou bien même le modifier si toutefois on veut apporter une amélioration à notre questionnaire.

2.3.2 Gestion d'examen

Cette partie consiste à regrouper les questionnaires créés en un examen pour une classe donnée.

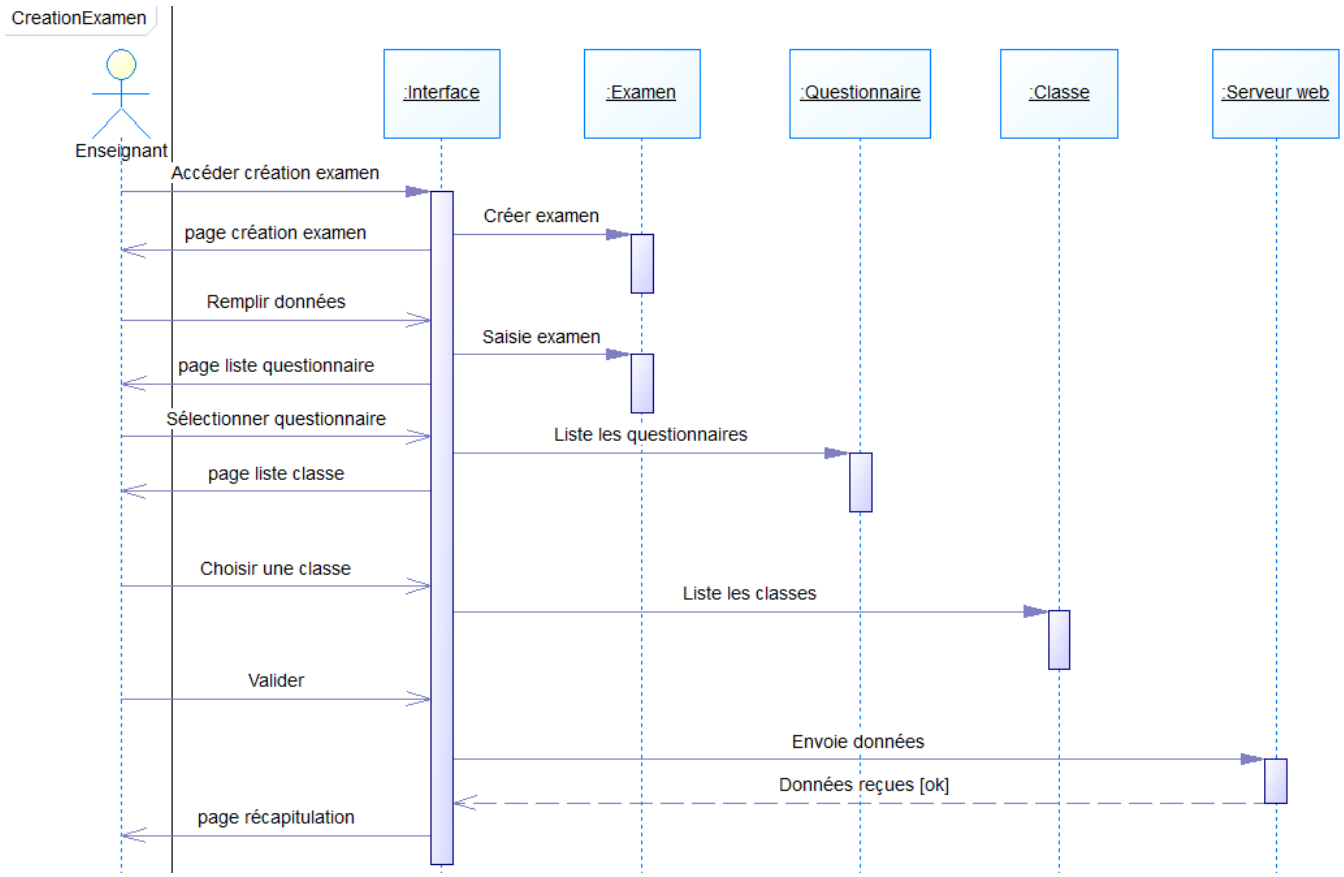

FIGURE 2.9 – Diagramme de séquence – Création examen

a. Création de l'examen

Cette création est accessible depuis l'interface « home » de l'enseignant. Cela permet aux enseignants de faire des évaluations notées dans un temps imparti. Une fois que les premières données sont remplies, il sera alors redirigé vers une page contenant la liste des questionnaires ayant été créés. Il doit en cocher au moins un questionnaire dans la liste. On peut tout cocher/décocher avec la case « **Cocher|Décocher tout** ». Un questionnaire coché fera partie de l'examen. Ensuite, une fois validé il sera alors redirigé vers une autre page pour choisir qu'une seule classe visée parmi la liste des classes. Une fois cette dernière validée, il sera redirigé vers une page récapitulative de sa création.

b. Création de la récapitulation

Cette partie consiste à résumer la création de l'examen. Il contient les informations nécessaires reliées à sa création, comme le titre, les questionnaires sélectionnés (avec ses

questions et propositions) et la classe visée pour cet examen. Cela permet aux professeurs de revoir ce qu'ils ont choisis, et de relire la totalité des questions. Dans cette page, il pourra aussi générer en PDF en cliquant sur le bouton « **Aperçu** ». On pourra modifier les paramètres avant tout impression. En outre, l'enseignant pourra annuler sa création et perdre toutes les données sélectionnées s'il désiste. Une fois que le professeur soit satisfait de sa création, il pourra alors valider et l'examen sera sauvegardé.

c. Détail de l'examen

Cette partie est similaire à la page récapitulative mais il n'y a rien à valider. Cela permet de revoir le résumé de l'examen avec ses questionnaires. Il est accessible depuis l'interface « **home** » du professeur. On pourra voir les détails de chaque examen. On pourra encore générer l'examen en PDF s'il ne l'a pas encore fait depuis la page de récapitulation. Mais aussi, le professeur pourra gérer ses examens en cliquant sur le bouton « **Tous mes examens** ». Il apercevra tous les examens qu'il a créés avec la date de création, et peut les consulter mais aussi de les supprimer.

2.3.3 Gestion de sondage

Cette partie consiste à réaliser un sondage juste et adapté pour recueillir des informations précises sur une classe visée.

a. Création du sondage

Cette première étape est la création du sondage, elle est accessible depuis la page « home » du professeur. Elle est formée d'une seule question à partir de laquelle les étudiants auront le choix de voter parmi les options proposées. Le sondage permet au professeur de voir ou de constater des résultats par rapport à une seule question à laquelle pourrait être rajouter une multitude de suggestion. Par exemple, si l'enseignant voudrait faire un sondage sur le nombre des étudiants qui ont leurs stages.

b. Création de la classe de sondage

Bien évidemment, pour faire un sondage, le professeur l'assigne à une classe. Il ne peut choisir qu'une seule classe pour un sondage créé. Cette assignation permet au professeur de faire une statistique sur la classe choisie. Pour faire simple, l'exemple précédent pourrait être servi que pour la classe « Master 1 Java » comme exemple.

c. Détail du sondage

Une fois que le sondage a été créé, le professeur pourra alors voir depuis la page « **home** » sa nouvelle création dans la section « **Mes derniers sondages** ». Chaque fois que le professeur crée un sondage, il pourra le voir instantanément sa création dans cette liste. Le détail du sondage contient les informations que le professeur a saisies et aussi le suivi de vote pour chaque option qu'il a proposé. Au tout début de la création, le nombre de vote est à 0, et une fois qu'un étudiant ait voté, le nombre sera incrémenté suivant son choix. Contrairement aux questionnaires, ici l'étudiant ne sera pas limité à un temps donné et il pourra répondre à son rythme.

2.4 Gestion des interfaces et d'ergonomie dans la partie Web et Mobile

« Rédigé et réalisé par Tony VO »

2.4.1 Interface Authentification et inscription

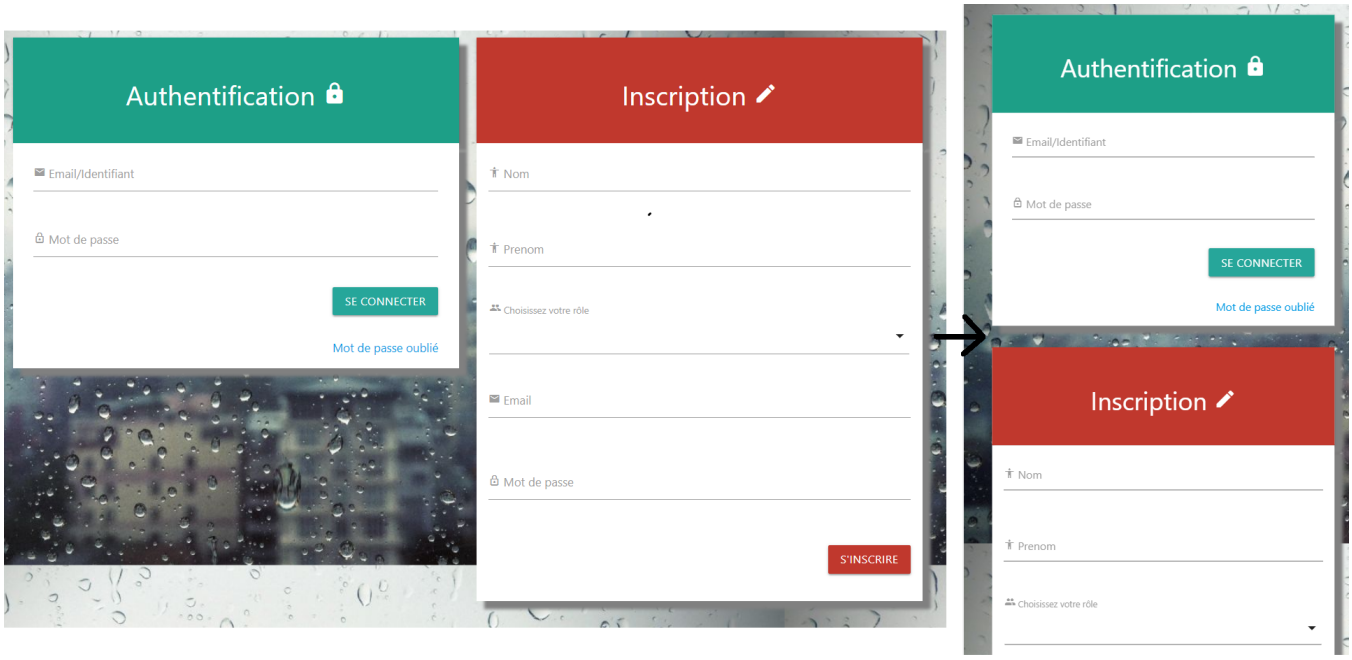

FIGURE 2.10 – Authentification version web et mobile

L'authentification et l'inscription ont été relié dans un seul fichier (l'inscription). La tache qui m'a été donnée est de le rendre ergonomique et responsive. Le Framework principalement utilisé était Materialize CSS . Afin de faire ces fonctionnalités, il fallait associer placer des « id » dans le code html afin de pouvoir modifier son apparence à sa volonté grâce au code CSS en utilisant le charactère « # » avec l'id de l'élément que l'on veut modifier. Pour le rendre responsive. Il fallait utiliser le système de grille (que le Framework Bootstrap avait popularisé avec Twitter) pour que dans sa version mobile, les deux formulaires ne soient pas aplatis de gauche à droite, mais mis l'un au-dessus de l'autre.

2.4.2 Interface Accueil

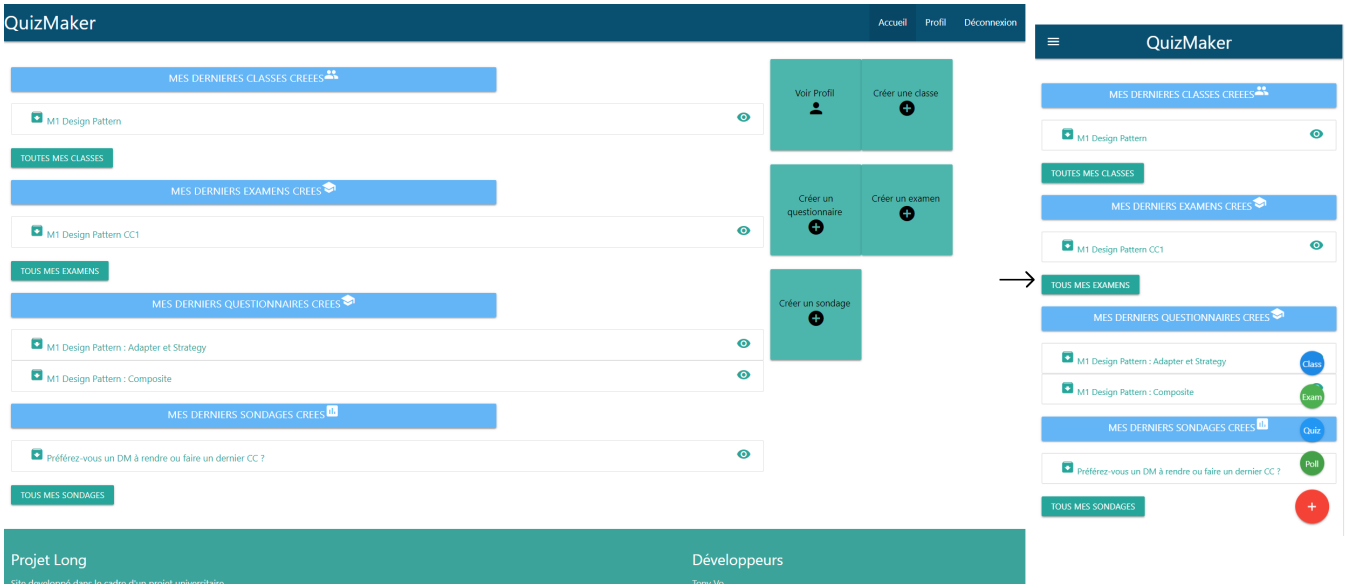

FIGURE 2.11 – Home de l'enseignant version WEB et Mobile

Le home enseignant est constitué de son contenu, de la navbar (présente dans tous les pages sauf à l'authentification/inscription) et du footer. Ce dernier ajoute juste un peu de style pour présenter ses développeurs et le « copyright » du projet Long. La barre de navigation horizontale est constituée de fonctionnalités classiques comme : retourner à l'accueil, consulter son profil et se déconnecter mais lorsqu'elle est utilisée dans la version mobile, un bouton supplémentaire apparaît sur la gauche pour permettre de dérouler une barre de navigation latérale qui contiendra en plus une photo de profil et différentes fonctionnalités

selon le rôle de l'utilisateur. Il peut également l'afficher en glissant du doigt de la gauche à la droite.

Ansi, Nous avons utilisé une de caractéristique d'Angular qui est le rooting pour naviguer d'une page à l'autre et non par des liens href. Le rooting, établi dans notre fichier app.rooting, permet au page de se charger plus rapidement et donc une navigation plus fluide. La majorité des boutons d'action déclenche des pop-ups qui sont également du code Javascript appelé par la classe « modal » dans le code HTML.

2.4.3 Interface Liste des Examens

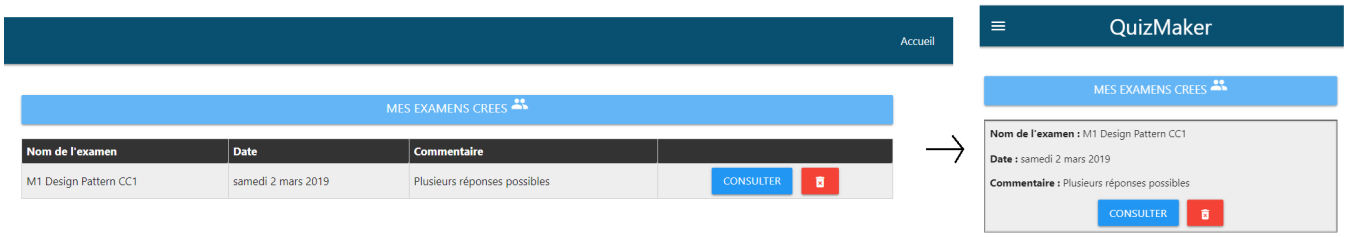

FIGURE 2.12 – Liste des examens créés version Web et mobile

Dans la liste des examens créées, l'interpolation est aussi utilisée, grâce à notre fichier Typescript où nous utilisons la fonction fetchAllExams(), qui utilise la fonction getAllExamen() de type examenService (se trouvant dans notre fichier Shared).

La liste des examens est responsive également, c'est-à-dire que dans la version mobile, la liste prend une autre forme plus adaptée (le header disparait et les entêtes se mettent à gauche des attributs au lieu d'être au-dessus.

Pour obtenir cette modification, au lieu de prendre le système de grille du home, il fallait utiliser dans le fichier CSS, le mot clé « @media » qui va déclencher des modifications visuelles à notre tableau selon la longueur minimum de l'écran (avec le mot clé « min-device-width ») et la longueur maximum (« max-device-width »). Tous les affichages de date ont été « piper » (c'est-à-dire qu'elles ont été mises dans le format « Jour Mois Année » et traduite en français). Seule la liste des soumissions d'examens s'est vu rajoutée, en plus de la date, l'heure en format h : :m : :s car la deadline d'un étudiant se joue souvent en seconde. La fonction « delete », qui sert à supprimer un examen de la base de données, est également

créée dans ces deux fichiers.

Dans le projet, toutes les pages comprenant une liste, utilisant ce design et généralement le même type d'interpolation m'ont été confié. En voici la liste exhaustive : :

Pour les professeurs :

- Liste classe créées.
- Liste examens créés.
- Listes des examens créés par classe.
- Listes des soumissions par examens.
- Liste des étudiants inscrits.
- Liste des sondages.
- Soumission des étudiants.

Pour les étudiants :

- la liste ses classes.
- la liste des examens à faire et déjà effectués.
- Liste des sondages à faire.

2.4.4 Interface Rechercher une classe

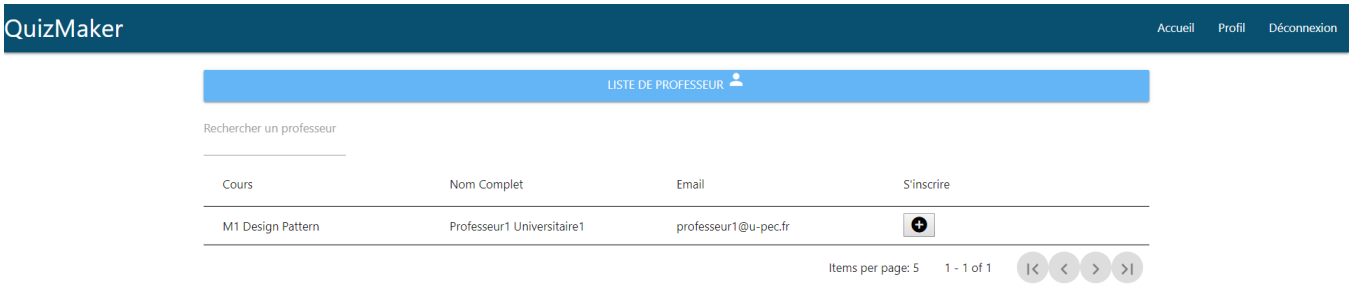

FIGURE 2.13 – Rechercher une classe

En tant qu'étudiant, En tant qu'étudiant, l'action « rechercher une classe » n'utilise pas le même tableau que ceux précédemment cités. Il utilise le module d'angular « angular Material » où il fallait l'importer dans le fichier Typescript pour pouvoir utiliser les fonctions de la barre de recherche et de la pagination, disponibles sur son site officiel (d'où la présence récurrente dans le code HTML du mot « mat »). Ces deux fonctions essentielles n'ont été

réalisables qu'avec l'utilisation de ce module.

2.4.5 Interface Résultat d'un examen

Lorsque l'étudiant termine de répondre un examen, cela le renverra dans la page de résultat, qui grâce à l'utilisation du « *ngif » dans le code html, renverra une couleur verte si la réponse est bonne, rouge si elle est fausse et orange si elle est incomplète.

2.4.6 Application Mobile

Pour la partie mobile, au départ, les tests ont été effectués via le débogueur de Chrome qui pouvait afficher la page web avec la taille d'un écran d'un smartphone (d'une hauteur supérieure à la longueur). C'est par cette méthode que le design a été recalibré pour la version mobile. Ensuite, lorsque la version avait l'air d'être concluant, l'application WEB a été converti grâce à Cordova en application mobile via le terminal Powershell de Windows et les tests ont été effectués, d'abord sans utiliser la base de données, sur mon Samsung Galaxy S8+.

Et à partir de là, de multiples erreurs avaient fait leur apparition, tels que la non fonctionnalité des boutons (bug corrigé en mettant les fonctions javascript qui active le modal dans le fichier Typescript correspondant) et la duplication de page lorsque l'on utilisait le bouton retour arrière du smartphone (bug ayant pris toute la journée à utiliser car la première piste avait été d'utiliser le « child rooting » qui permettait de séparé le cheminement des pages tel un arbre, mais la solution avait été seulement de supprimer une ligne de code (qui appelait le script Cordova) dans le fichier index.html ».

Les tests ont ensuite été effectués avec la base de données en utilisant modifiant l'adresse IP et le port dans le fichier environment.ts. Les tests sont facilement réalisables chez soit lorsque nous utilisons l'adresse IP de notre box mais sont plus ardus, une fois arrivé à l'université où il fallait rechanger l'adresse IP du fichier environment.ts, mettre un de notre smartphone en routeur et d'y connecter notre ordinateur contenant la base de données ainsi que notre smartphone contenant l'application mobile. Ce dernier récupère bien les données de la base de données mais des micro coupures de réseau sont fréquemment présentes, ce qui force la déconnexion de l'utilisateur de l'application. C'est pour cela que nous avions essayé

de négocier avec notre responsable de projet Mme Sovanna Tan, afin de pouvoir obtenir un serveur de l'université.

Une fois cette préparation terminée, et plusieurs simulations faites, l'application Web s'est révélée concluante et plutôt prometteur (de notre point de vue).

2.5 Gestion des soumissions et statistiques « Rédigé et réalisé par Junior Mabika »

2.5.1 Répondre aux questions

Après création d'un examen, les étudiants ont la possibilité de répondre aux questions de celui-ci. La première solution élaborée pour rendre cette partie fonctionnelle, était d'afficher chaque question ainsi que les options qui lui étaient associées sur une seule et même page. Mais le property binding d'Angular rendait ardue la récupération des informations des champs sélectionnés. La seconde solution consistait à répondre au questionnaire, question par question, avec un bouton permettant de naviguer à la question suivante et de finir le test.

Nous affichons également sur cette page le temps écoulé. À chaque réponse, les identifiants des choix de l'étudiant, par rapport à une certaine question, sont récupérés via une méthode onsubmit() dans le fichier typescript ; puis empilés dans une liste de réponse qui sera envoyée vers la base de données et traitée pour la correction .

L'examen est soumis à une classe par le professeur et s'affiche sur la page d'accueil des étudiants inscrits à cette dernière dans le champ ''mes derniers examens ''. Au l'instant où l'étudiant décide d'accéder au test, il est soumis à une contrainte de temps. Ensuite dès qu'il finit de répondre, le bouton 'soumettre' entraîne une redirection vers la page du résultat du test. Sur cette page nous avons la note de l'étudiant ainsi qu'un récapitulatif de ses réponses.

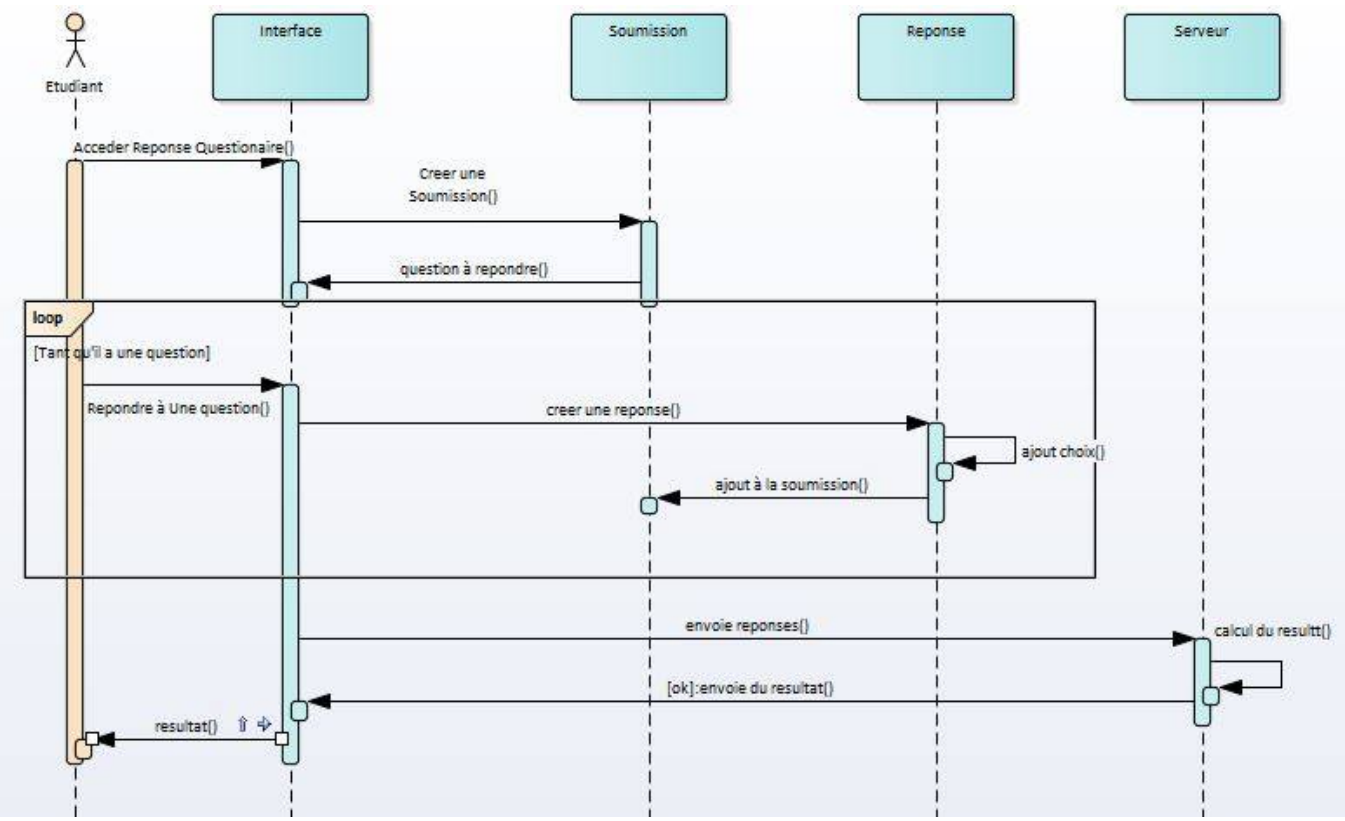

FIGURE 2.14 – Diagramme – Répondre aux questions

L'étudiant accède à l'examen, et répond au test, question par question. À l'instant où il passe à la question suivante, ses choix sont stockés dans un objet de type 'answer' qui dans son modèle, contient une liste de choix. Après la soumission, l'objet de type 'answer' est sauvegardé dans une 'soumission' qui sera envoyée au serveur pour la correction. Pour finir, le résultat est envoyé à l'étudiant via la page résultat.

2.5.2 Répondre au sondage

Un sondage étant une seule question, le traitement de celui-ci est beaucoup plus simple qu'un questionnaire, l'affichage et la récupération des options sélectionnées étaient évidents après avoir implémenté la partie réponse aux examens. Nous affichons la question et ses options, et récupérons les options choisies par l'étudiant. De même que pour les examens, elle devient disponible après création sur la page d'accueil de l'étudiant dans le champ, mes derniers sondages. Elle n'est pas soumise à une contrainte de temps.

2.5.3 Statistique : Chart.JS

Chart.js est un outil qui permet de faire des images et des diagrammes dans le style HTML5 en un tour de main à partir de données, c'est-à-dire sans aucun plugin supplémentaire ou un autre outil similaire. Et ce grâce à JavaScript ! Pourquoi Chart.js ? C'est l'un des modules java script le mieux adapté pour représenter nos statistiques. Il fonctionne Angular. La transmission des données, tout comme la configuration des diagrammes, se fait en utilisant JSON. Il y a deux choses à préparer avant de dessiner un diagramme avec Chart.js :

— Les données que nous voulons montrer

— L'HTML tout autour

Pour notre projet, nous avons utilisé les Pie chart du module chart.js. Ils vont nous servir à représenter les données de sondages sur que nous allons extraire du backend. Après une réponse aux sondages par l'étudiant, les données sont représentées dans un diagramme circulaire dans ''tous mes sondages" coté professeurs.

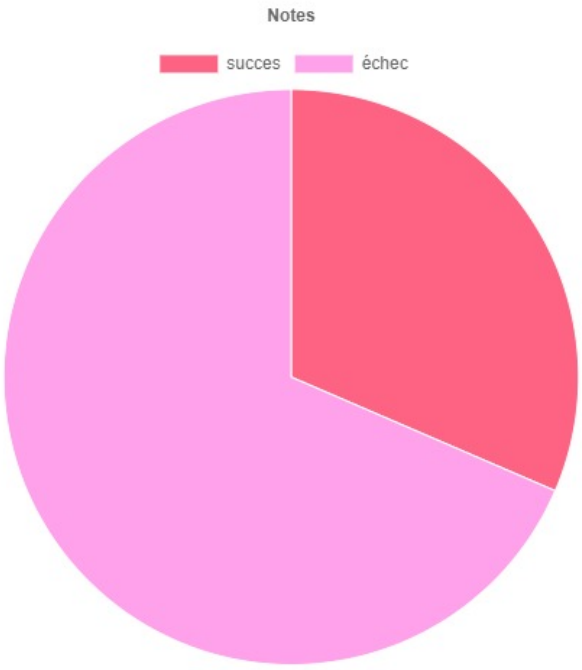

FIGURE 2.15 – Pie chart

Chapitre 3

Organisation du travail

« Rédigé par Ghassen Belkhir »

3.1 Organisation du groupe

L'organisation est un facteur primordial dans le travail collaboratif qui joue un rôle important sur l'efficacité et l'agilité d'une équipe. C'est pourquoi nous avons choisi la méthode Scrum pour travailler sur ce projet.

Pour appliquer la méthode Scrum, on a dû préciser les rôles de chaque membre de l'équipe selon cette stratégie qui consiste à avoir les acteurs suivants :

— Le Product Owner : celui qui a la vision du produit à réaliser.Et dans le monde professionnel, c'est le client qui demande le produit.
— Le Scrum Master : assure la productivité de chaque membre de l'équipe et suit l'avancement du projet, ainsi il assure l'application des règles définies. Mais aussi, il est un développeur qui participe à la production.
— Les développeurs : Ce sont tous les membres du groupe.

La méthode Scrum consiste aussi à définir les besoins sous forme des users stories ou en d'autres termes, ce sont les fonctionnalités que le système doit offrir. Ses users stories sont regroupés dans ce qu'on appelle backlog. Nous avons préparé notre backlog durant notre première réunion.

Durant cette courte période avec les cours qui se déroulaient en même temps que le projet, nous n'avons pas pu appliquer entièrement la méthode Scrum en ce qui concerne la programmation des réunions. Mais à l'aide de certains outils, nous avons assuré la continuité de notre travail même à distance.

3.2 Outils de travail

Notre outil principal est le puissant GitLab, le gestionnaire de version. Cet outil nous permet de :

- Garder les historiques de nos modifications pour pouvoir revenir en arrière.
- Fusionner les codes écrits par chaque membre du groupe.
- Aider à résoudre les conflits en indiquant les codes qui causent ce conflit.

Pour que nous puissions bien utiliser GitLab, nous devions mettre une stratégie de travail pour éviter les ambigüités et la désynchronisation entre les membres de l'équipe.

Donc nous avons décidé de suivre la bonne pratique et d'appliquer une partie de la stratégie GitFlow et l'adapter à notre situation. À travers ses principes, l'équipe peut travailler en collaboration sans aucun conflit qui soit irréversible.
Afin de mettre en place cette stratégie nous avons mis les règles suivantes :

- Aucun développeur ne doit modifier directement la branche principale master et la branche de développement nommée beta sauf le Scrum Master.
- La branche master contient notre version finale.
- La branche beta contient une nouvelle version avec des nouvelles fonctionnalités
- Quand un développeur commence à ajouter une nouvelle fonctionnalité, il doit créer une nouvelle branche à partir de la branche beta, ensuite il fait son travail sur sa branche puis il finit par merger sa nouvelle version avec la branche beta et de supprimer la branche créée.

3.3 Répartition des tâches

Notre système est composé d'une partie Backend et Frontend, où nous avons divisé les tâches comme suit :

Tous les membres du groupe participent à la spécification de besoin et à la conception. Un membre s'occupe de la création du serveur web et de la base de données, puis le reste du groupe s'occupe de la partie Frontend en divisant entre eux les interfaces qu'ils doivent créer. Nous avons choisi cette répartition des tâches selon le poids de chaque fonctionnalité car la partie Frontend demande un énorme travail pour produire une interface conviviale. C'est pourquoi nous avions attribué trois membres du groupe pour s'en occuper. En outre, chaque membre a choisi sa tâche selon sa propre volonté.

Le tableau suivant représente les responsabilités de chaque membre de l'équipe :

Membres de l'équipe		
Nom	Role	Responsabilités
Ghassen Belkhir	Scrum Master	Gestion du projet (*) Développement de la partie Backend Intégration des services à la partie Frontend Documentation du Api
Kevin Low Pein	Développeur	Développer les fonctionnalités suivantes (Frontend) : Gestion des questionnaires Gestion des examens Gestion des sondages Documentation du code
Tony VO	Développeur	Développer les fonctionnalités suivantes (Frontend) : Répondre aux questions Interface Accueil Interface Liste des Examens Interface Rechercher une classe Interface Résultat d'un examen Application Mobile Ecrire Manuel d'utilisation Documentation du code
Junior Mabika	Développeur	Développer les fonctionnalités suivantes (Frontend) : Répondre aux questionnaires Répondre aux sondages Gestion de statistiques Documentation du code

(*) Gestion du projet : il s'agit de la préparation d'une liste de fonctionnalités (backlog) qui doivent être faites dans une durée déterminé et ensuite de les proposer à l'équipe pour confier les tâches. Il s'agit aussi de suivre l'avancement de la réalisation des tâches afin de respecter les durées estimées, ainsi d'assurer la synchronisation entre tous les membres.

Même si chacun a sa tâche respective, nous coopérons pour résoudre les problèmes ren-

contrés, et nous nous basons notre collaboration sur la critique constructive et le travail et dans la bonne humeur.

Chapitre 4

Conclusion

« Rédigé par Kevin Low Pein »

Au terme de ce projet, nous avons atteint l'objectif qui répondait aux besoins nécessaires de l'enseignant. Les fonctionnalités proposées ont été réalisées et sont fonctionnelles pour l'application Web et mobile. Le système de QCM intelligent en ligne est mis en place. Plusieurs améliorations pourront être apportées mais cela sortirait du contexte attendu.

Par ailleurs, le travail en groupe nous a permis de partager les connaissances, les tâches pour gagner du temps, de savoir s'organiser, de s'entraider en cas de blocage, mais aussi de savoir travailler à distance sans perdre les améliorations des autres grâce au logiciel de gestion de version Git. Malgré l'abandon d'un membre, on a su se répartir les tâches pour combler son absence.

En outre, ce projet nous a été avantageux en termes de connaissances, il nous a permis d'avoir des expériences dans la technologie MEAN Stack, de nous fortifier avec de nouvelles techniques mais aussi de consolider les connaissances déjà acquises.

Chapitre 5

Références

— Git (lien vers notre projet sur Git) :
https ://git-etudiants.lacl.fr/ghassen.belkhir/projet_long
— OpenClassroom (Développer des applications avec Angular) :
https ://openclassrooms.com/fr/courses/4668271-developpez-des-applications-web-avec-angular
— Node.JS :
https ://nodejs.org/en/download/
— Logiciel pour créer des diagrammes : https ://umbrello.kde.org/
— Tutoriels sur Mean Stack :
http ://www.codaffection.com/
— https ://cordova.apache.org/
— https ://www.mongodb.com/download-center/community
— https ://materializecss.com/getting-started.html
— https ://www.w3schools.com/css/