Vincent Alves
Pierre Merlin d'Estreux de Beaugrenier
Thibault Gimenez
Benjamin Legrand

Projet Génie Logiciel 2: Livrable Final

31 mai 2014

Table des matières

1 Introduction

Ce projet Génie Logiciel 2 que l'on nous a assigné doit aboutir à la réalisation d'un gestionnaire de QCM.
Le but du projet a été de développer un logiciel de gestion et de traitement de QCM : Questionnaires à choix multiples. Ce logiciel est destiné à une école et a pour but d'évaluer les élèves sur des modules en leur faisant passer des QCM. Il a également pour but de donner le ressenti des élèves à travers des QCM sur les modules. L'équipe en charge de ce projet est formée de Pierre Merlin d'Estreux de Beaugrenier, Thibault Gimenez, Benjamin Legrand et Vincent Alves.

Dans ce document, nous rappellerons brièvement l'analyse des besoins du client que nous avions effectuée, puis parlerons de notre implémentation de cette analyse en java.

2 Rappel de l'analyse des besoins

En se basant sur les demandes du client, nous avons établi les listes de besoins suivantes :

2.1 Besoins fonctionnels

- Trois types d'utilisateurs : L'administrateur, les professeurs, et les élèves.
  - Créer des niveaux d'accès différents pour les trois types d'utilisateur : L'administrateur peut accéder à plus de ressources que le professeur, qui peut accéder à plus de ressources que l'élève.
- L'administrateur peut définir les utilisateurs et les promotions. Les promotions sont des listes d'élèves. Les élèves sont définis par leur nom et prénom.

- L'administrateur peut définir les modules enseignés à l'école par leur nom, la liste des modules prérequis et le syllabus du module.

- Un professeur peut créer des QCM, qui sont un ensemble de questions avec un libellé accompagné d'une liste non vide de réponses fermées.
  1. Les QCM définis peuvent être privés(utilisable par le professeur uniquement) ou publics(utilisable par tous les professeurs)
  2. Chaque réponse d'un QCM est définie par un libellé et une information précisant si elle est vraie ou fausse.

- Un professeur peut créer des sessions de QCM. Une session est définie par ses dates de début/fin (attribut de la session) ainsi que le module et la promotion auxquelles elle est associée.
  1. Un professeur qui a créé une session de QCM peut consulter à tout moment les résultats partiels(si elle n'est pas terminée) ou définitifs de cette session. Il peut voir les résultats de chaque élève et les statistiques.
  2. Les résultats demandés par le professeur peuvent être les scores individuels des élèves participants, ou des statistiques sur l'ensemble des élèves : Moyenne, écart-type, fréquence de bonnes réponses par question.

- On doit pouvoir connaître la liste des modules qu'enseigne un professeur.

- Un élève peut répondre à un QCM s'il est inscrit à la session et si la session est ouverte.
  1. Un élève peut refaire une session de QCM tant qu'il n'a pas atteint un nombre de répétitions précisé par le professeur.
  2. Un élève peut consulter les résultats d'une session à laquelle il a participé, à la condition qu'elle se soit terminée.

- Le client nous impose l'utilisation de JAVA 1.7.

2.2 Besoins non-fonctionnels

- Le logiciel doit avoir un temps de réponse inférieur à 10 secondes pour toute action.

- Le logiciel doit être facilement utilisable par des étudiants et professeurs n'ayant pas nécessairement de connaissances avancées en informatique.

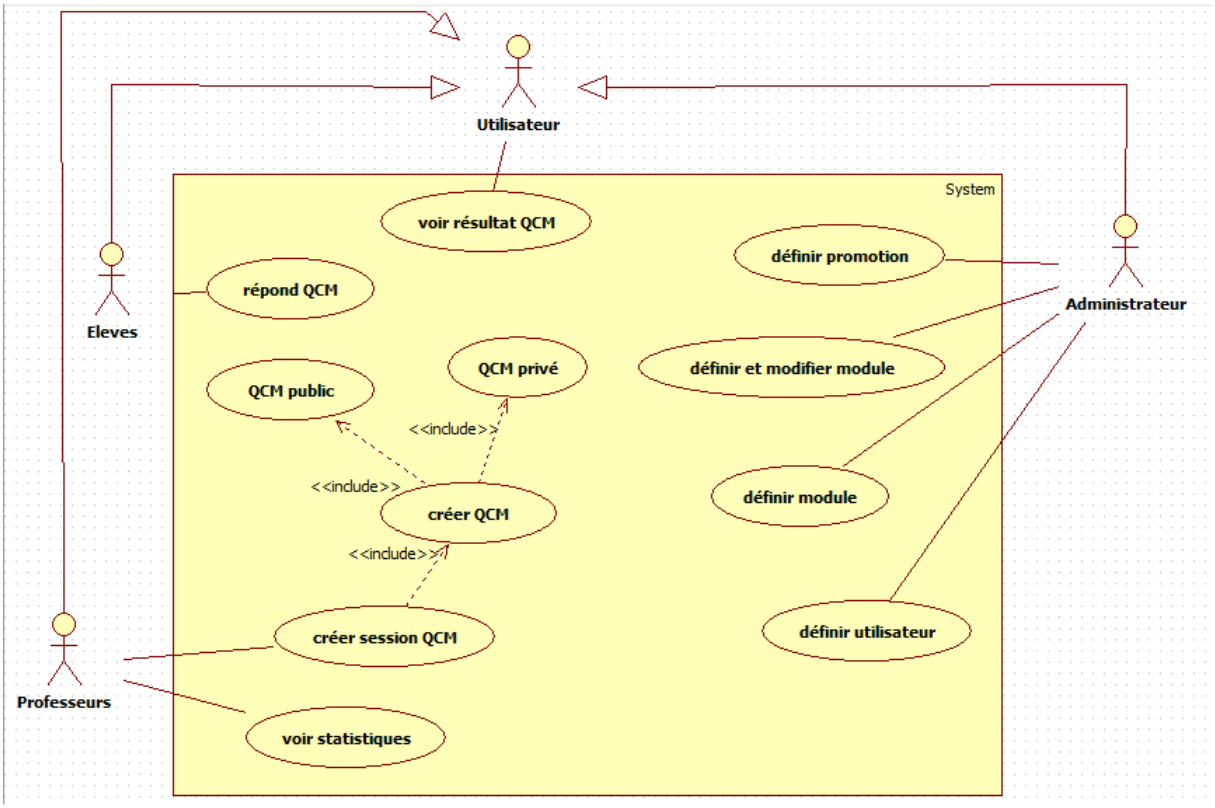
- La réalisation du logiciel doit être divisée en plusieurs jalons.
- De ce fait, la communication avec le client doit être fréquente à tous les jalons pour vérifier que la réalisation va dans la direction souhaitée par le client.

- Le logiciel doit pouvoir fonctionner sous Windows ou GNU/Linux.

2.3 Diagramme de cas

Le diagramme de cas sert à identifier :
- Le système
- Les acteurs qui interagissent avec le système
- Les actions des acteurs sur le système

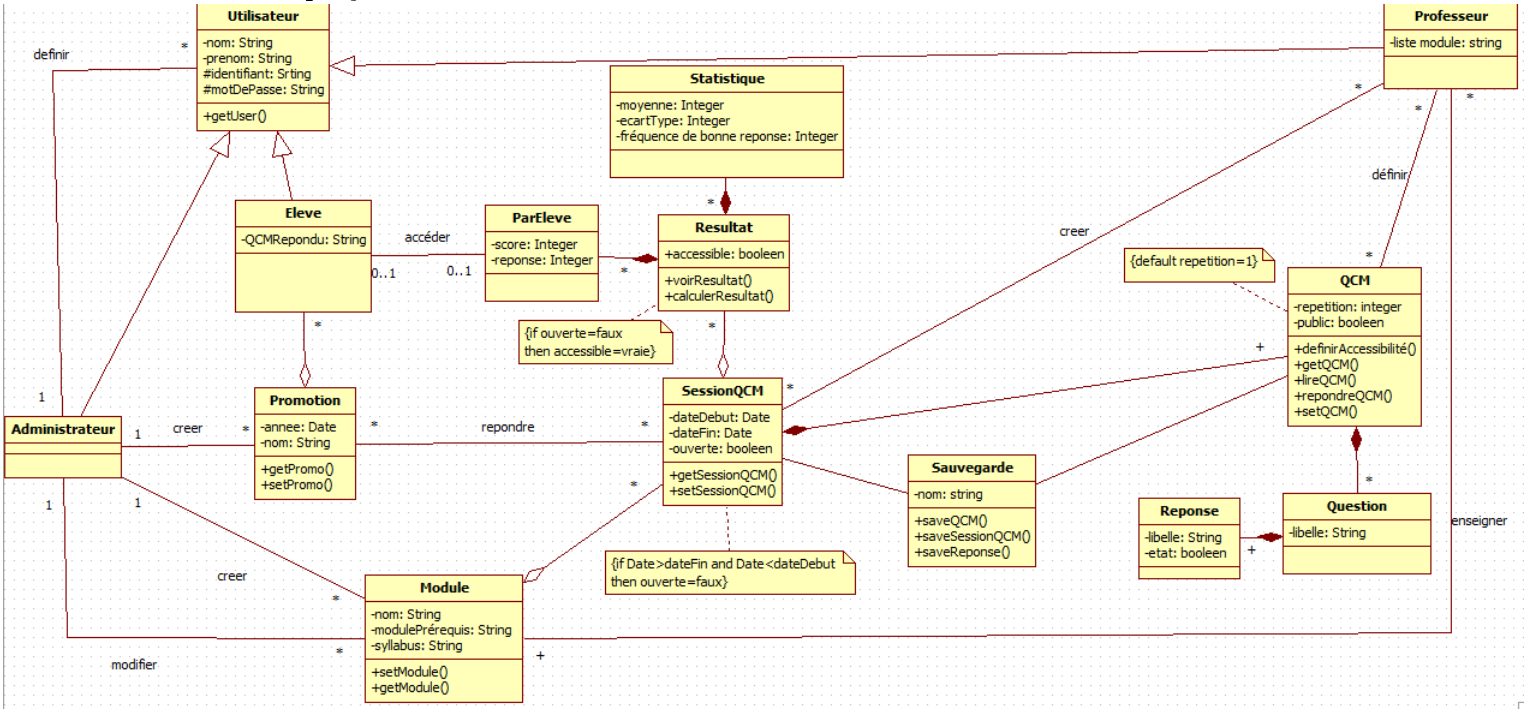
Ce qui nous a permis d'établir le diagramme de cas d'utilisation suivant :



Un acteur est une entité externe (personne, machine...) qui interagit avec le système. On distingue ainsi trois types d'utilisateur (administrateur, professeur et élève) avec des actions différentes. Tous les détails des actions des utilisateurs ne sont pas représentés pour une question de lisibilité.

2.4 Diagramme de classe

Le diagramme de classe est le diagramme le plus indispensable à la programmation en Java. Le diagramme de classes est un schéma utilisé en génie logiciel pour présenter les classes et les interfaces des systèmes ainsi que les différentes relations entre celles-ci. Une classe est un ensemble de fonctions et de données (attributs) qui sont liées ensemble par un champ sémantique. Le diagramme de classe du projet est le suivant :



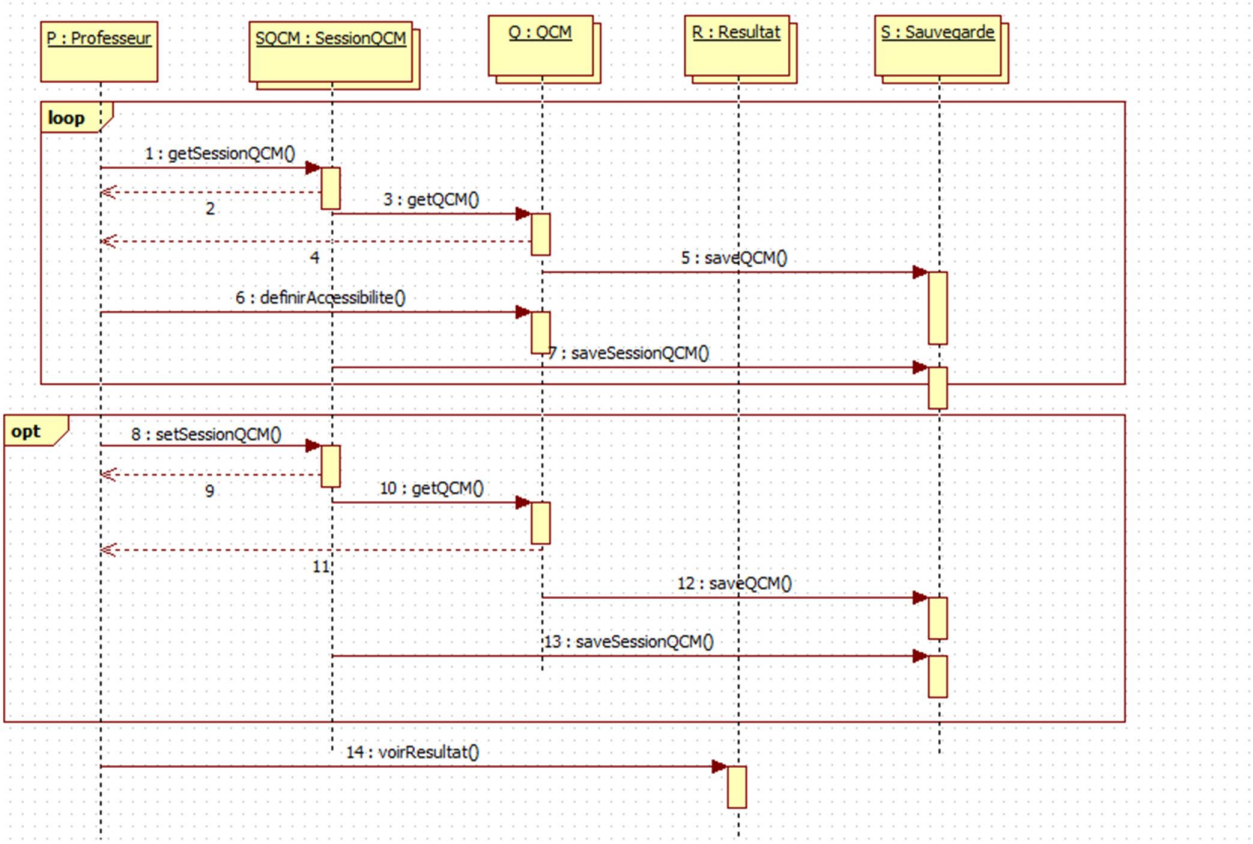
Le diagramme de classe est divisé en trois parties. L'une comprenant les utilisateurs (professeur, élève et administrateur). Une autre comprenant les QCM, les modules et les promotions. C'est-à-dire les données créées par l'administrateur et les professeurs. La dernière partie du diagramme de classe contient les résultats des QCM.

2.5 Diagramme d'activité

Voir le fichier joint « projet5 » pour les détails de ce diagramme. Il a été choisi de faire des boucles pour mettre en évidence le fait que l'utilisateur puisse effectuer une nouvelle action après avoir effectué une action. Des sauvegardes ont été rajoutées avant la déconnexion de l'utilisateur afin de sauvegarder les modifications de l'utilisateur. Nous avons choisi une boucle pour la connexion avec trois essai de saisie de mot de passe afin d'assurer la sécurité des données du logiciel.

2.6 Diagramme de séquence

Ci dessous, le diagramme de séquence pour le professeur.



Pour les autres diagrammes de séquence voir les pièces jointes. Le diagramme de séquence donne l'ordre de l'action effectuée par l'utilisateur. Il a été choisi de mettre des retours après la création et la modification des QCM pour indiquer à l'utilisateur que les modifications ont été effectuées.

3 Notre implémentation sous JAVA

Nous avons utilisé l'environnement de développement Eclipse pour implémenter ce projet.

3.1 Séparation en modules

Le projet a été séparé en 4 modules principaux :

- Utilisateurs : Les classes ayant rapport aux utilisateurs : Utilisateur, Administrateur, Promotion, etc...
- QCM : Les classes ayant rapport aux QCMs : Sessions, Questions, Réponses, etc...
- Interface : La classe Menu, qui gère l'interaction avec l'utilisateur.
- Bases de Données : Les classes sauvegardées dans des fichiers et qui contiennent toutes les données.

Ces modules sont indépendants les uns des autres, ce qui permet de mieux organiser le code et d'éviter des erreurs dans un module dues à un autre. (Par exemple, une erreur de l'interface suite à une absence de QCMs)

3.2 Fonctionnement général

La classe Menu recherche dans la UserDatabase le login/password entré par l'utilisateur, et le renvoie sur un menu correspondant à ses droits(ou pas de menus si le login n'existe pas.)

Depuis ces menus, différentes choses sont possibles selon les droits de l'utilisateur :

- Ajouter des Utilisateurs : Une nouvelle instance d'Utilisateur est crée et ajoutée à la base de données
- Ajouter des QCMs : Une nouvelle instance de QCM est crée et ajoutée à la base de données
- Ajouter des Sessions : On liste les QCMs crées et on prend l'un deux pour faire une session
- Participer à une Session : On liste les sessions créees. Si celle sélectionnée est ouverte et qu'il reste des essais à l'élève, on démarre la session de réponse.
- Observer les résultats des Sessions : On récupère tous les résultats à une session et on effectue des tests et calculs dessus.
- Modifier des Sessions : On récupère dans la base de données la session correspondante et on la modifie.

Tous ces menus proposent également l'option logoff, qui si sélectionnée renvoie l'utilisateur à l'écran de login.

3.3 Sérialisation

Toutes les classes qui ont besoin d'être sauvegardées, c'est à dire les utilisateurs, les sessions et les QCMs, sont sérialisables via la définition

```
implements Serializable
```

Ces classes sont ensuite stockées dans deux classes principales du module Bases de Données :

UserDatabase : Utilisateurs, Modules, Promotions.

QCMDatabase : QCMs, Sessions, Réponses.

A chaque logoff d'un utilisateur, le programme sauvegarde les bases de données affectées(User pour l'administrateur, QCM pour les professeurs) dans un fichier par classe.

Ces fichiers sont par la suite chargés au prochain démarrage du programme, si ils existent.

3.4 Les résultats finaux

Nous n'avons malheureusement pas fini d'implémenter toute l'analyse, suite à plusieurs erreurs au niveau de la sérialisation et de la création des classes. Il nous manque donc les modules/promotions, ainsi qu'une réponse aux QCM fonctionnelle.

L'impémentation finale est proche de notre analyse originale, malgré certaines classes ajoutées(notamment les classes de bases de données)

4 Manuel d'instruction

Une fois le programme lancé via

```
java -jar "GL2.jar"
```

L'utilisateur est prompté de s'authentifier. Les identifiants ajoutés à la base de données par la suite fonctionneront, mais le programme vient avec deux comptes prédéfinis :

Un administrateur
`admin/admin`
Un professeur
`prof/prof`

Une fois authentifié sur un compte, l'utilisateur est présenté un menu d'options selon la classe du compte, comme vu dans l'analyse des besoins. Au logoff, il est possible de s'authentifier sur un autre compte.