Génie Logiciel 2 : Rapport technique

Christian Ingouff
Pierre-Alexandre Tyndal
Sonia Seddiki
Yann Charbonnier

EISTI

2013/2014
Semestre 2
Jalon 4

Table des matières

Partie 1

Introduction

Ce livrable va effectuer le bilan des dernières étapes du projet. Jusqu'à présent, nous avons explicité le cahier des charges, avant de le traduire grâce aux outils d'analyse UML. Les nombreux diagrammes que nous en avons tirés nous apportent une base solide pour la programmation.

Il nous reste à détailler la conception du logiciel avant de passer à sa programmation. Ce sont ces deux étapes que nous verrons ici.

Partie 2

Rappel : Analyse UML

Dans les livrables précédents, nous avons détaillé les différents diagrammes de l'analyse UML, à savoir :

- Diagramme de cas d'utilisation
- Diagramme de classes
- Diagramme d'activités
- Diagramme de séquence
- Diagramme d'états-transitions

Toute la partie analyse a consisté entre des allers-retours permanents entre les différents diagrammes, afin de les affiner, les corriger, les modifier.

2.1 Diagramme de cas d'utilisation

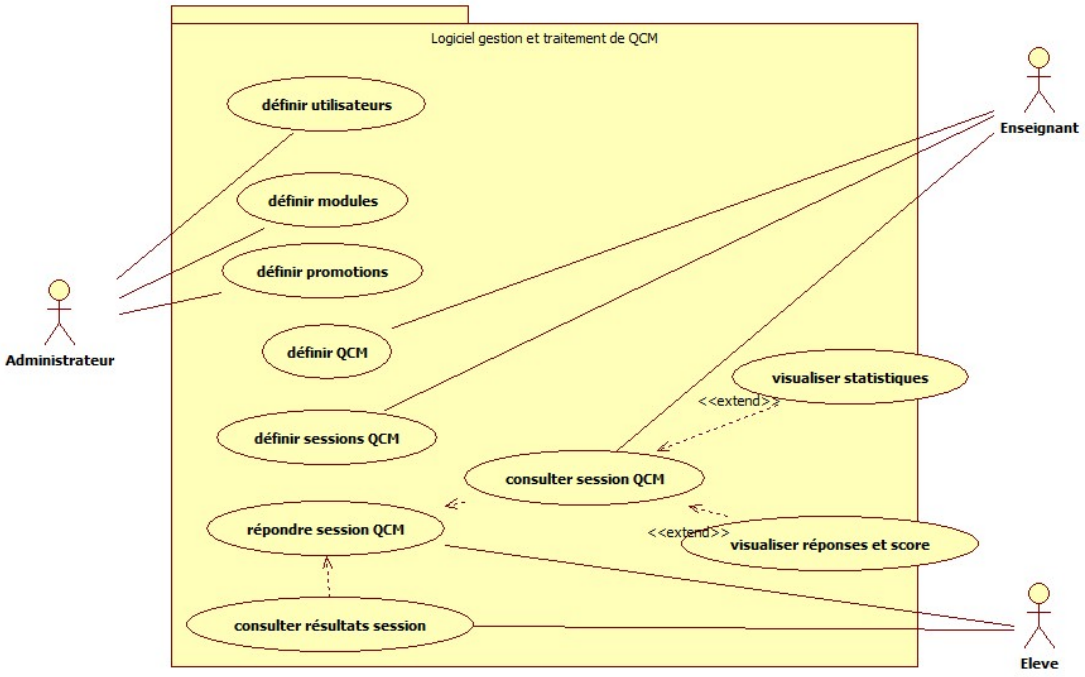

FIGURE 2.1 – Diagramme de cas d'utilisation

Ce diagramme liste les différents acteurs ainsi que les différentes opérations qu'ils peuvent effectuer sur le logiciel.

2.2 Diagramme de classes

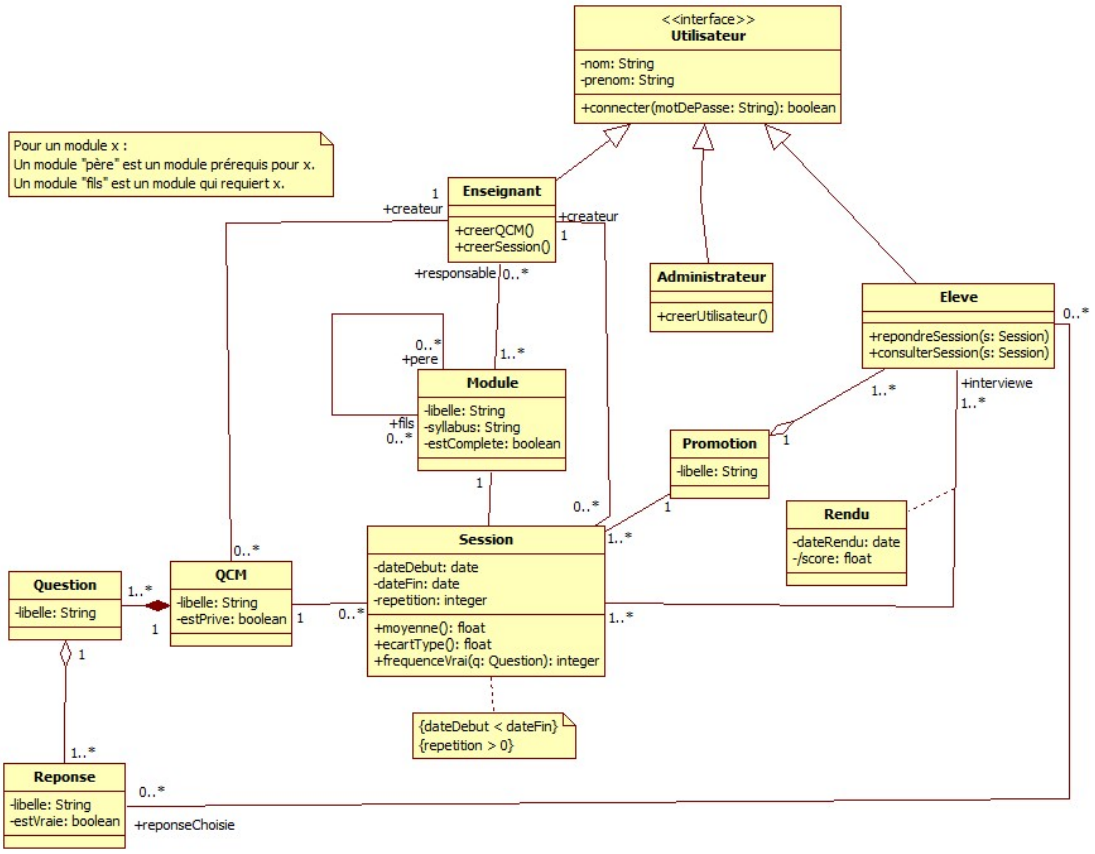

FIGURE 2.2 – Diagramme de classes

Ce diagramme montre les différentes classes avec leurs attributs et méthodes, ainsi que les liens entre ces classes. Il a fait l'objet de plusieurs modifications tout au long du projet. Toutefois la version proposée ici est l'ancienne version, les modifications ayant été faites à la volée directement dans le code. En effet, lors de sa première version, nous l'avions conçu avec la même approche que pour concevoir un MCD (cf. premier projet Génie Logiciel) : éviter la redondance de données. Autrement dit, si une donnée peut être déduite du diagramme, il est inutile de la mentionner comme attribut dans une classe. Or ce n'est pas l'objectif recherché par le diagramme de classes. Le but est de préparer le travail de programmation en faisant la liste de toutes les classes, attributs et méthodes nécessaires au codage du logiciel. Nous détaillerons quelques exemples par la suite.

2.3 Diagramme d'activité

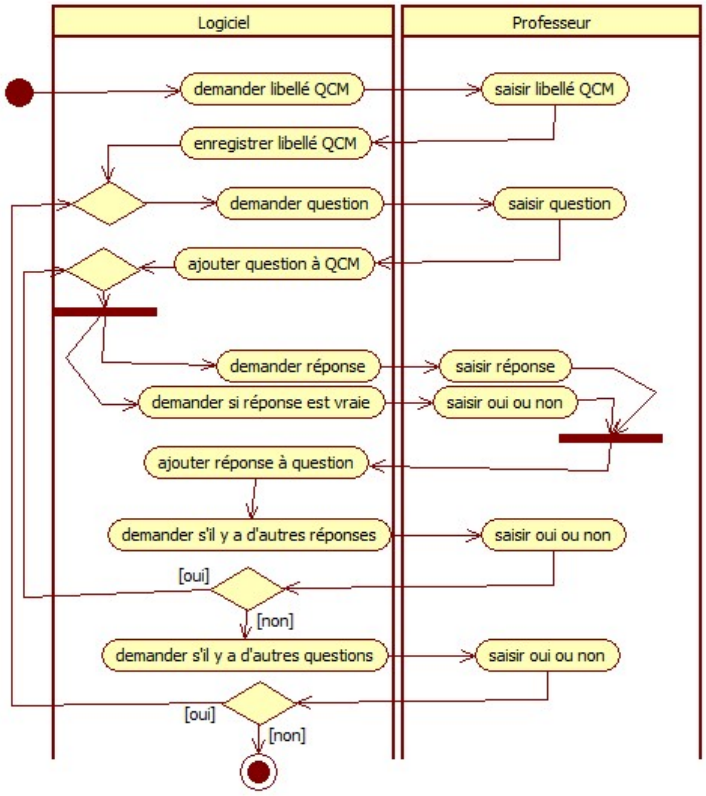

FIGURE 2.3 – Diagramme d'activité - Définir QCM

Ce diagramme est une première approche sur la façon de procéder pour le cas "Définir QCM".

2.4 Diagramme de séquence

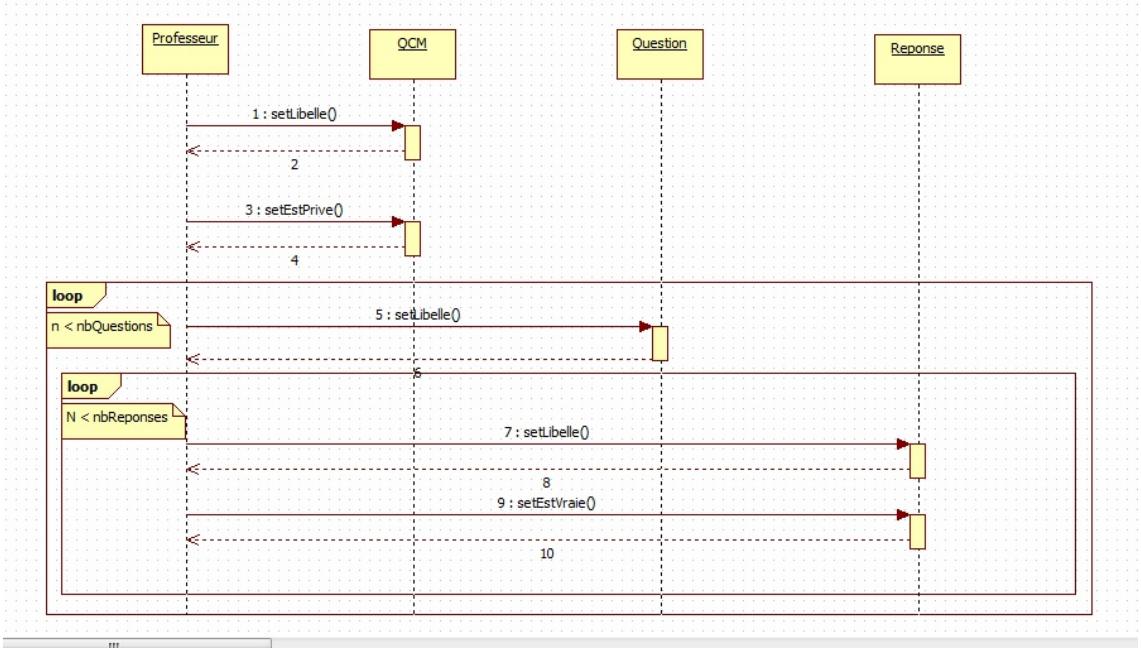

FIGURE 2.4 – Diagramme de séquence - Définir QCM

Le diagramme de séquence est une version plus proche du code du diagramme d'activité. Il montre le processus décrit dans le diagramme d'activités, mais avec les classes et méthodes définies dans le diagramme de classes. C'est ici que nous avons constaté quelques lacunes dans notre diagramme de classes. En effet, nous avons remarqué que les variables nbQuestions et nbReponses (qui sont respectivement le nombre de questions du QCM et le nombre de réponses d'une question donnée) étaient nécessaire, du moins pour réaliser le diagramme comme on l'entendait. Or ces deux données n'apparaissaient dans aucune classe. Nous avons donc par la suite rajouté nbQuestions comme attribut de QCM et nbReponses comme attribut de Question.

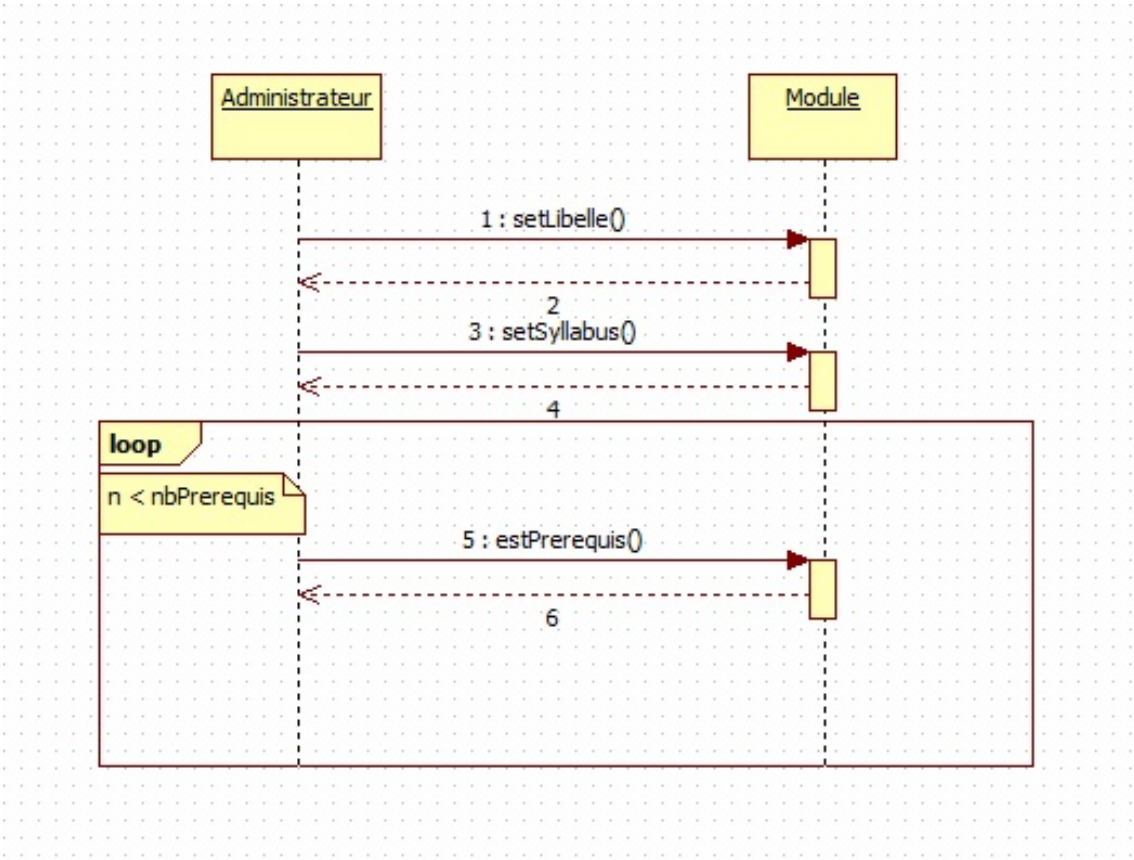

FIGURE 2.5 – Diagramme de séquence - Définir Module

Nous avons rencontré un problème similaire ici. En effet, dans le diagramme de classes, la classe Module est associée à elle-même, et chaque extrémité est renommée "père" ou "fils" suivant que le module a ou non un statut de prérequis dans une configuration donnée. Par soucis d'éviter la redondance de données, nous avions prévu de ne pas rajouter d'attribut booléen déterminant si le module était un prérequis ou non. Mais lors de la réalisation du diagramme de séquence, nous avons constaté que l'ajout d'un attribut booléen "estPrerequis" était nécessaire pour programmer le statut de prérequis d'un module. Cette modification a été donc prise en compte par la suite dans la programmation.

2.5 Diagramme d'états-transitions

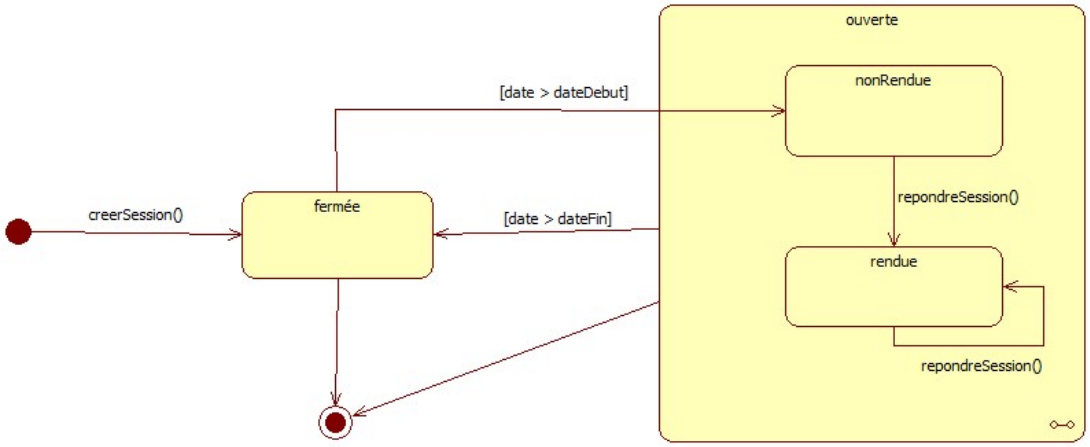

FIGURE 2.6 – Diagramme d'états-transitions - Session de QCM

La réalisation de ce diagramme a pu être effectuée sans ajout d'attributs et de méthodes supplémentaires dans le diagramme de classes. Il état conforme à notre modélisation du problème.

Partie 3

Conception

La partie conception comportera les différentes décisions prises relativement à la création du logiciel. Nous exposerons les différents choix techniques utilisés dans le code par la suite.

3.1 Le pattern MVC

MVC est un patron de programmation dont l'acronyme signifie Model-View-Controller. Ce design pattern sépare les différentes classes dans 3 packages nommés comme précédemment de la manière suivante :

- Le package Model : c'est le coeur du logiciel. Il regroupe toutes les classes contenant ses fonctionnalités principales (i.e. celles définies dans le diagramme de classes)
- Le package View : c'est l'interface du logiciel. Il regroupe donc toutes les classes qui créent cette interface (par exemple, les différentes fenêtres Elève/Professeur/Administrateur) permettant a l'utilisateur de voir ce qu'il fait et communiquer avec le logiciel. Une réalisation graphique à été tentée mais étant donné que nous n'arrivions pas a rafraîchir la page, nous avons été dans l'obligation d'arréter l'idée afin de respecter au mieux les délais
- Le package Controller : permet de faire le lien entre Model et View. Il détecte les actions effectuées sur l'interface (un clic sur un bouton par exemple), contrôle leur validité, et envoie les modifications à opérer au Model (le lancement de telle ou telle fonction associé au clic sur le bouton correspondant)

Les interactions entre les différents packages sont résumées sur le schéma suivant :

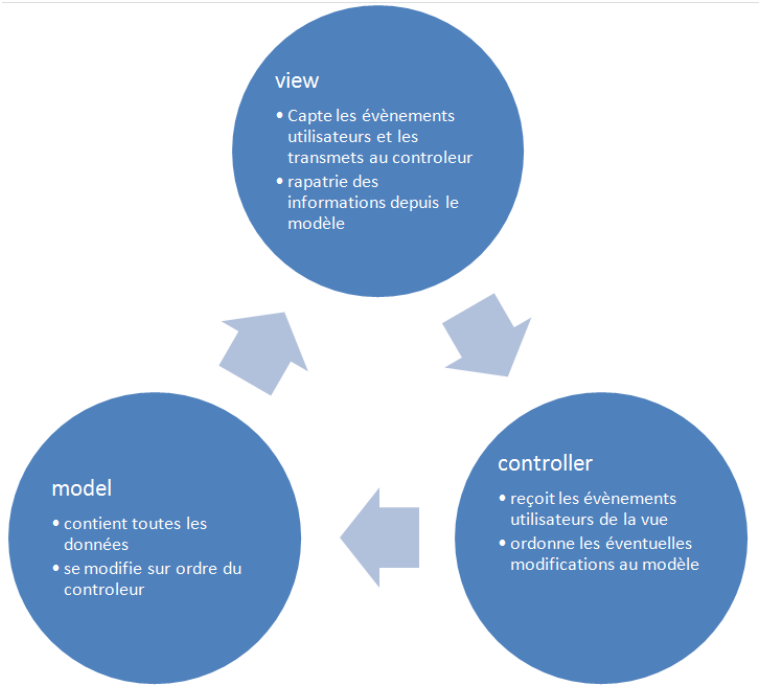

FIGURE 3.1 – Le pattern MVC

Le pattern MVC est une des exigences de conception imposées par le cahier des charges.

3.2 Sérialisation

Plutôt que d'utiliser la sérialisation classique de Java avec l'interface Serializable ou l'interface JDBC (Java DataBase Connectivity), nous avons décidé de sauvegarder nos données en les écrivant sur un fichier XML à part. Ceci nous permet de séparer clairement les différentes données, ainsi que d'avoir un rendu mieux structuré. Ci-dessous un exemple d'un fichier XML généré par le programme :

L'exemple présenté ici est ue sauvegarde de différents utilisateurs. Pour chaque utilisateur, on voit immédiatement son statut (Administrateur, Elève ou Enseignant), ainsi que ses différents attributs (nom, prénom, login et mot de passe, dont nous parlerons dans la partie "Authentification").

FIGURE 3.2 – Sauvegarde de quelques utilisateurs

3.3 Authentification

L'authentification n'était pas mentionnée dans le cahier des charges. Toutefois, la nécessité de l'implémenter est évidente.

Nous avons opté pour une authentification classique basée sur un identifiant "login" et un mot de passe "motDePasse", lesquels sont associés aux nom et prénom de l'utilisateur via la classe concernée. Pour des raisons de sécurité, nous avons contemplé la possibilité de chiffrer les mots de passe avec la fonction de hachage MD5. Le mot de passe qui apparaît dans le fichier XML serait alors déjà crypté, ce qui empêche de le récupérer directement depuis le fichier XML. La comparaison avec l'entrée utilisateur se ferait en chiffrant son entrée et en confrontant les résultats.

Cependant, l'implémentation du cryptage ne donnait pas forcément le même résultat pour une même entrée, ce qui nous a contraints à nous recaler sur une approche normale.

Partie 4

Programmation

4.1 Détail du programme

Nous avons implémenté les objectifs du projet dans un programme Java qui fonctionne en console. L'organisation du travail s'est développée sous le pattern MVC, par lequel nous structurons la partie de ce programme.

4.1.1 Model

Le dossier "src/model" contient l'ensemble des classes définies dans le diagramme de classes, ainsi que des classes de manipulation de listes d'objets sérialisés dans "src/model/list".

Plus en détail, les "Handlers" font à la fois office de conteneurs pour les listes d'objets sérialisés, et aussi d'interpréteurs/exporteurs vers des fichiers XML servant de bases de données. Les "Handlers" sont construits grâce à une importation de fichiers xml (contenus dans "bin/data/") au début du programme et sont actualisés à chaque fois qu'un utilisateur termine une gestion d'objet sérialisable.

Par exemple, un administrateur peut engendrer une sauvegarde des promotions en quittant le menu de gestion de promotions. Cela va ainsi pour toutes les classes affectées par un "Handler".

4.1.2 View

Le dossier "src/view" contient la vue du programme, l'interface que l'utilisateur du programme est capable de voir.

Nous avions effectué une tentative d'implémentation d'IHM dans la dernière semaine de rendu que vous pouvez trouver dans le dossier "src/view/ihm". Cette tentative a échoué à cause d'un problème de rafraichissement.

Le dossier "src/view/console" regroupe l'ensemble des textes itérés dans l'interface console. La plupart des fonctions à l'intérieur des classes ainsi présentées sont en "static", ce qui permet leur utilisation sans créer d'instance de classe. Le code est compartimenté selon sa fonctionnalité (i.e. une fonction pour une saisie).

4.1.3 Controller

Le dossier "src/controller" contient la partie contrôle : c'est ce qui permet au programme de transitionner d'une vue à une autre. Il fait ainsi en appelant les méthodes de la vue adéquates au fur et à mesure que l'utilisateur entre ses données.

Nous comptions contenir le contrôle de l'IHM dans un dossier "src/controller/ihm", à savoir tout ce qui concerne les listeners et les changements d'états, dans l'envisagement d'une implémentation graphique.

Dans le dossier "src/controller/console" se trouvent l'ensemble des contrôles correspondant à chaque fenêtre de vue. Ainsi, à chaque vue qui s'affiche devant l'utilisateur se déroule le contrôle correspondant en parallèle pour la changer.

4.2 Jeu de test

Pour tester si notre programme fonctionne correctement, il est nécessaire de créer un jeu de test. Celui se composera de :

- Un administrateur
- Deux enseignants
- Deux élèves appartenant à une promotion
- Un module avec un prérequis
- Un QCM de 5 questions (4 réponses par question) appartenant à ce même module
- Une session de QCM et son rendu

Ainsi, nous aurons un objet représentatif de chaque classe de notre modèle.

Ces jeux de test se trouvent dans le dossier "bin/data/". Ce sont des fichiers XML structurant les données sérialisées.

Partie 5

Mode d'emploi du programme

Dans ce livrable, vous trouverez non seulement notre code source, mais aussi un "programme.jar" qui permet à l'utilisateur d'exécuter le programme en effectuant la commande au bon dossier à la ligne de commande :

```
java -jar programme.jar
```

Pour information, le fichier principal est le fichier "src/Main.java". A noter que le programme requiert le dossier "bin/data/" pour la base de données.

Le programme accueille l'utilisateur avec un invité de connexion. Pour tester les différentes interfaces, nous donnons accès à plusieurs comptes utilisateurs de différents rôles :

- Administrateur : login "ingouffchr", mot de passe "azerty"
- Enseignant : login "cb", mot de passe "azerty"
- Eleve : login "rouanet-l", mot de passe "azerty"

Nous structurons le programme en 3 parties, correspondant à chaque type d'utilisateur.

5.1 Administrateur

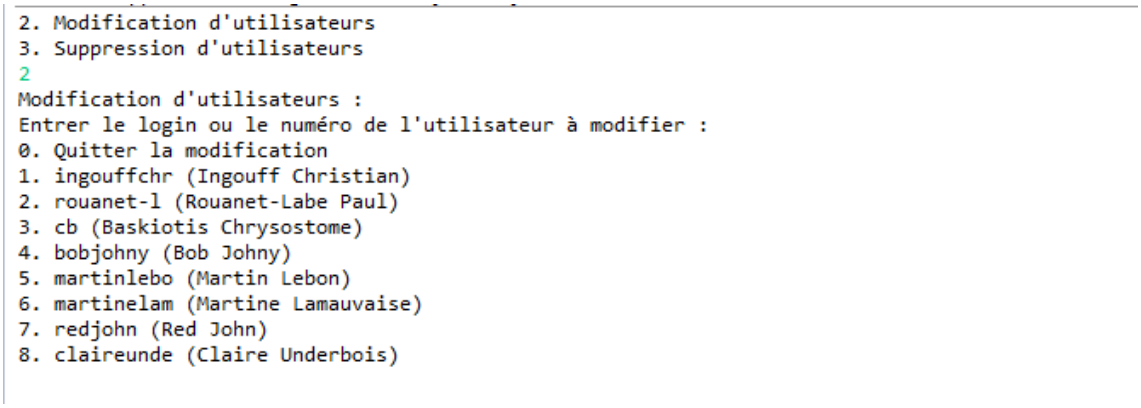

FIGURE 5.1 – Interface administrateur

L'administrateur a accès à plusieurs gestions (création, modification, suppression) :

- Gestion des utilisateurs
- Gestion des modules
- Gestion des promotions

5.1.1 Gestion des utilisateurs

Création d'utilisateurs

Une création d'utilisateurs demande à l'administrateur d'entrer le login et le mot de passe du nouvel utilisateur successivement. Elle lui demande ensuite de spécifier son rôle (1 pour Administrateur, 2 pour Enseignant, 3 pour Elève).

Si l'utilisateur est un enseignant, la création lui demande de spécifier la liste des modules gérés par celui-ci. La saisie fonctionne ainsi : l'administrateur tape "+<numéro>" pour ajouter un module à la liste courante et "-<numéro>" pour retirer un module de la liste courante.

L'exécution se termine avec un retour au menu Gestion. Le login et le mot de passe sont générés automatiquement et peuvent être modifiés ultérieurement.

Modification d'utilisateurs

La modification d'utilisateurs demande à l'administrateur de choisir un utilisateur à modifier. Elle lui demande ensuite de choisir l'attribut à modifier : nom, prénom, login, mot de passe ou éventuellement modules gérés pour un Enseignant.

La sélection d'un attribut mène l'administrateur à des menus similaires à la création. Leurs exécutions se terminent toutes avec un retour au menu précédent.

Suppression d'utilisateurs

L'administrateur peut également supprimer des utilisateurs. La suppression lui demande l'utilisateur à supprimer, lui demande confirmation, à laquelle l'utilisateur confirme par "o" ou infirme par "n". La fin de suppression ramène l'utilisateur sur la liste des utilisateurs, ce qui lui permet de supprimer en chaîne.

5.1.2 Gestion des modules

Création de modules

La création de modules demande à l'administrateur le libellé et le syllabus du module. Elle demande ensuite l'ajout de modules pré-requis (modules pères) par la manière analogue à l'ajout d'utilisateur.

L'exécution se termine avec un retour au menu Gestion.

Modification de modules

La modification de modules demande à l'administrateur de choisir un module à modifier. Elle lui demande ensuite de choisir l'attribut à modifier : libellé, syllabus ou modules pères.

La sélection d'un attribut mène l'administrateur à des menus similaires à la création. Leurs exécutions se terminent toutes avec un retour au menu précédent.

Suppression de modules

L'administrateur peut également supprimer des modules. La suppression lui demande le module à supprimer, lui demande confirmation, à laquelle l'utilisateur confirme par "o" ou infirme par "n". La fin de suppression ramène l'utilisateur sur la liste des modules, ce qui lui permet de supprimer en chaîne.

5.1.3 Gestion des promotions

Création de promotions

La création de promotions demande à l'administrateur le libellé de la promotion. Elle demande ensuite l'ajout d'élèves existants dans la liste de la manière habituelle.

L'exécution se termine avec un retour au menu Gestion.

Modification de promotions

La modification de promotions demande à l'administrateur de choisir une promotion à modifier. Elle lui demande ensuite de choisir l'attribut à modifier : libellé ou liste d'élèves.

La sélection d'un attribut mène l'administrateur à des menus similaires à la création. Leurs exécutions se terminent toutes avec un retour au menu précédent.

Suppression de promotions

L'administrateur peut également supprimer des promotions. La suppression lui demande la promotion à supprimer, lui demande confirmation, à laquelle l'utilisateur confirme par "o" ou infirme par "n". La fin de suppression ramène l'utilisateur sur la liste des promotions, ce qui lui permet de supprimer en chaîne.

5.2 Enseignant

```
0. Quitter la modification
1. DS de Mathématiques
2. Le bal des Eistiens
2
QCM 'Le bal des Eistiens' (par cb (Baskiotis Chrysostome)) :
Choisir un numéro de question ou une tâche :
0. Quitter le QCM
1. Modifier le libellé du QCM
2. Passer la visibilité du QCM à : PUBLIC
3. Ajouter une question

4. Cherchez l'intrus
5. Où sont les câbles coupés ?
6. Je code :
```

FIGURE 5.2 – Interface enseignant

L'enseignant a accès à plusieurs fonctionnalités :
- Gestion des QCM
- Gestion des sessions de QCM
- Consultation des statistiques de sessions

5.2.1 Gestion des QCM

Création de QCM

La création de QCM demande à l'enseignant le libellé du QCM. S'en suit alors l'ajout de questions : demande du libellé suivie de l'ajout des réponses à la question. Les réponses demandent le libellé et leur caractère vrai ou non par rapport à la question.

L'ajout de réponses et de questions se fait jusqu'à ce que l'utilisateur décide de l'arrêter par la confirmation de l'invité.

L'exécution se termine avec un retour au menu Gestion.

Modification de QCM

La modification de QCM demande à l'enseignant de choisir un QCM à modifier. Elle lui demande ensuite de choisir l'attribut à modifier : libellé, ou une question au choix. L'utilisateur peut également ajouter des questions dans ce menu.

L'enseignant ne peut modifier que les QCM publics ou ses propres QCM privés.

Le sous-menu sousjacent propose à l'enseignant de modifier les réponses, d'en ajouter ou de supprimer la question. Le menu encore sousjacent d'une réponse permet de supprimer la question ou de modifier son libellé ou son caractère.

La sélection d'un attribut mène l'enseignant à des menus similaires à la création. Leurs exécutions se terminent toutes avec un retour au menu précédent.

Suppression de QCM

L'enseignant peut également supprimer des QCM. La suppression lui demande le QCM à supprimer, lui demande confirmation, à laquelle l'utilisateur confirme par "o" ou infirme par "n". La fin de suppression ramène l'utilisateur sur la liste des QCM, ce qui lui permet de supprimer en chaîne.

L'enseignant ne peut supprimer que les QCM publics ou ses propres QCM privés.

5.2.2 Gestion des sessions

Création de sessions

La création de sessions demande à l'enseignant une date de début de session, une date de fin, un nombre maximum de rendus pour la session (nombre de répétitions), un QCM concerné parmi ceux disponibles, le module concerné et une promotion prédéfinie.

La date de fin ne peut être ni ultérieure à la date d'aujourd'hui, ni ultérieure à la date de début de session.

L'exécution se termine avec un retour au menu Gestion.

Modification de sessions

La modification de sessions demande à l'enseignant de choisir une session à modifier. Elle lui demande ensuite de choisir l'attribut à modifier : date de début, date de fin, répétition, promotion, QCM et module.

L'enseignant ne peut modifier que ses propres sessions. La date de fin suit les mêmes règles qu'à la création.

La sélection d'un attribut mène l'enseignant à des menus similaires à la création. Leurs exécutions se terminent toutes avec un retour au menu précédent.

Suppression de sessions

L'enseignant peut également supprimer des sessions. La suppression lui demande la session à supprimer, lui demande confirmation, à laquelle l'utilisateur confirme par "o" ou infirme par "n". La fin de suppression ramène l'utilisateur sur la liste des sessions, ce qui lui permet de supprimer en chaîne.

5.2.3 Consultation des statistiques de sessions

L'enseignant peut consulter les statistiques de ses propres sessions. Le programme invite l'enseignant à choisir la session à examiner. Chaque choix successif permet d'afficher les statistiques de la session choisie.

Les statistiques proposées sont la moyenne, la variance et la fréquence de bonnes réponses par question. A noter que s'il y a 2 bonnes réponses dans une question, les deux sont comptés dans le compteur pour la question.

5.3 Elève

```
Sessions finies :
2. QCM 'Le bal des Eistiens' ('Grec' pour CPI1) du 01-01-2012 au 01-01-2014
2
Consultation de résultats :
Session du QCM 'Le bal des Eistiens' ('Grec' pour CPI1) du 01-01-2012 au 01-01-2014
Votre note est de 15.0 sur 20.
Retour au menu principal...
Choisissez votre tâche :
0. Quitter le programme
Sessions en cours :
1. QCM 'Le bal des Eistiens' ('Grec' pour CPI1) du 01-02-2013 au 01-01-2015

Sessions finies :
2. QCM 'Le bal des Eistiens' ('Grec' pour CPI1) du 01-01-2012 au 01-01-2014
```

FIGURE 5.3 – Interface élève

L'élève se voit afficher la liste des sessions auxquelles il est affecté :
- Les sessions en cours peuvent être concourrues
- Les sessions finies peuvent être consultées pour leurs résultats
- Les sessions qui n'ont pas encore commencé ne sont délibérément pas affichées

5.3.1 Sessions en cours

Les sessions en cours permettent à l'élève de répondre à un QCM à condition qu'il n'aie pas rendu pour la session plus de fois que le nombre de répétitions défini pour la session.

Le programme propose à l'élève de répondre au QCM en lui proposant de cocher des réponses à chaque question avant de valider (avec le choix "0"). Il fait ceci jusqu'à la fin de la série de questions.

L'aboutissement de la session amène l'élève au menu principal et enregistre le rendu.

5.3.2 Sessions terminées

Les sessions terminées peuvent être examinées par l'élève pour qu'il connaisse le score qu'il a fait à une session. S'il a effectué plusieurs rendus, le maximum pour la session est retourné.