BOULALOUA Fethi
MECHERI Akram
M1 Informatique

Projet Java avancé - Emploi du temps
Année 2015-2016

Table des matières

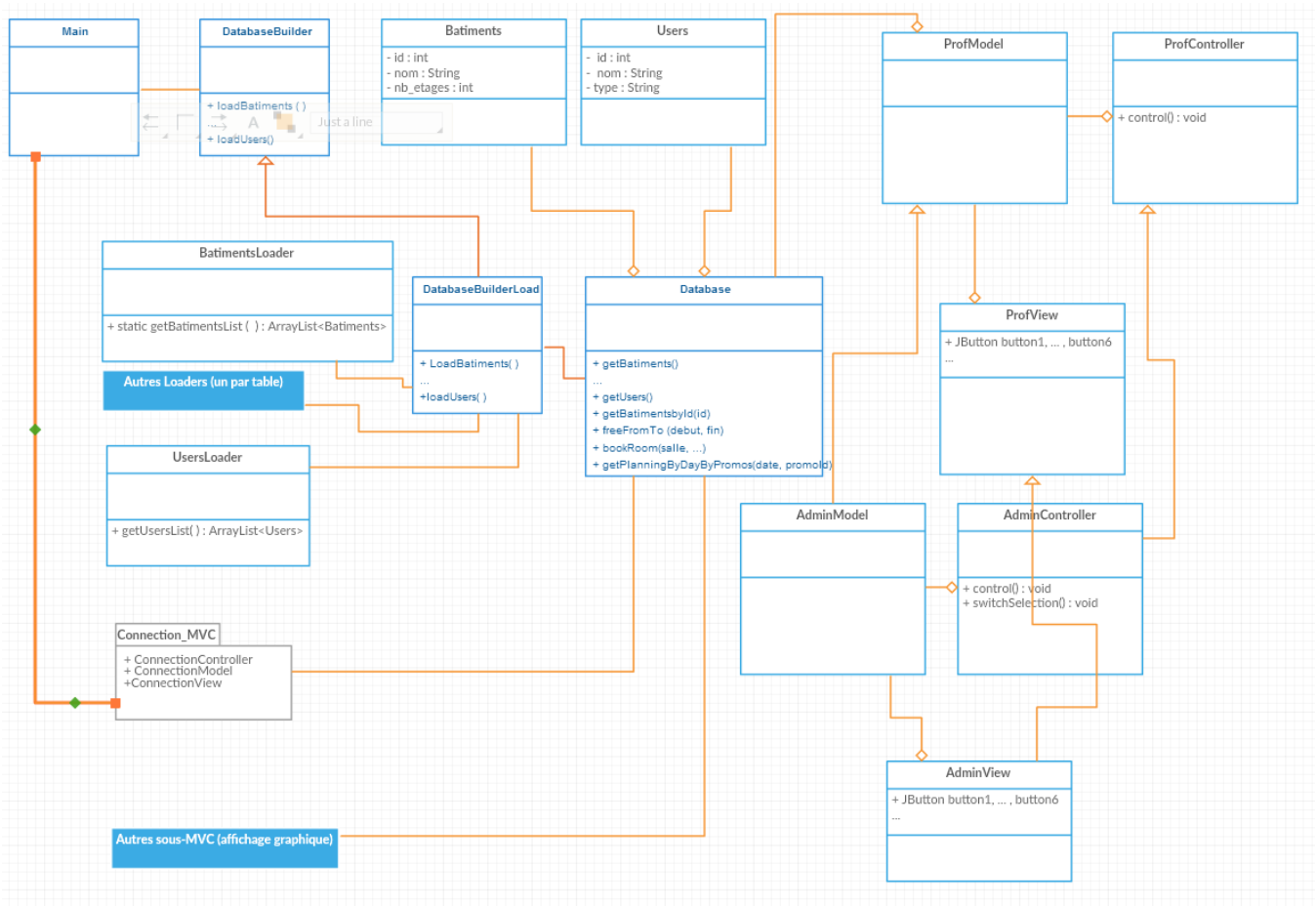
I . Structure du code

Le projet nécessitait l'utilisation d'une base de données, pour laquelle Akram a pu trouver un hébergeur distant. Il était donc nécessaire de construire notre modélisation autour d'elle.
Nous nous sommes entendus sur la création d'un objet Database qui contiendrait les différentes tables sous formes d'ArrayList et qui serait rempli avec le contenu de la base SQL à chaque lancement du programme.
Pour ce faire, le pattern Builder a été utilisé afin d'y intégrer séparément les différentes tables.

Le package memoryDatabaseExchange contient les classes nécessaires à la connexion à la base de données, et à la récupération de celle-ci pour créer notre objet, contenu dans le package allTables.
Un des objectifs du projet étant d'évaluer notre capacité à utiliser les fonctionnalités de Java 8, nous avons usé et abusé des streams pour manipuler les données contenus dans notre base, utilisant en particulier la fonction filter() pour simuler les "WHERE" des fonctions SQL.
La plupart des fonctions de manipulation des données sont directement contenues dans l'objet Database lui-même, par souci de facilité d'accès.

Les différentes fenêtres du programme possèdent chacune un MVC différent, pour des raisons qui seront discutées dans le IV.

Voici le diagramme de classes (non exhaustif) du projet :



Ainsi que le diagramme de cas d'utilisation :

II . Répartition des tâches

De manière sommaire, Akram a écrit le gros des interfaces graphiques ainsi que les parties qui utilisaient des requêtes pour extraire des données de la base SQL vers du Java. Il s'est également chargé de créer celle-ci, après que nous ayons débattu du meilleur format pour la modéliser.
Je me suis chargé du builder, des fonctions de récupération des données, d'une petite partie de l'interface graphique, et de la rédaction du présent rapport.
A mesure que le projet progressait cependant, la frontière devenait plus floue et nous travaillions un peu partout, le travail tournant à de la correction de code plutôt qu'à son écriture.

III . Amélioration envisagées

En l'état, plusieurs choses viennent à l'esprit pour améliorer le projet.
Les interfaces graphiques pourraient être regroupées sous un MVC, une certaine répétition étant présente entre les codes.
L'objet Database pourrait faire l'objet d'un pattern Singleton, puisqu'il n'est pas supposé en exister plus d'une.
Les types particuliers d'Users (Profs et Admins) sont pour l'instant différenciés par un champ String. Ceci dans une volonté de coller au maximum à la base SQL dont on extrait les données.
Une légère optimisation serait de faire hériter une classe Profs des Users et d'affecter le champ String à la bonne valeur. Cela éviterait une ligne de filter lorsque l'on veut extraire uniquement les enseignants de la liste des utilisateurs.

Les cours sont, pour l'heure, affichés sous forme d'un String listant les attributs. On peut imaginer une mise en forme un peu plus propre de ceux-ci.
Les administrateurs peuvent ajouter un élément aux tables, il serait également idéal de leur permettre d'en supprimer un.

IV . Difficultés rencontrées

La principale embûche lors du projet fut le partage de fichiers.
Des dysfonctionnements de git, potentiellement causés par notre inexpérience avec le logiciel, ont fini par m'empêcher d'ajouter des commits, nous obligeant à nous organiser différemment hors du répertoire git et revenir push régulièrement "à la main". Un peu de code a également dû être réécrit suite à des problèmes avec les branches.

Notre projet initial était d'écrire un unique MVC pour l'affichage graphique, avec un contrôleur changeant de vue suivant la situation. Le modèle aurait été l'objet Database, ou un objet le contenant.
Cependant, l'affichage a présenté des problèmes et refusait de charger la nouvelle vue lorsque l'on quittait une fenêtre. En conséquence, opérant sous des contraintes de temps assez drastiques à cause de la présence des partiels et d'un autre projet durant la période de développement, nous avons dû improviser rapidement la solution d'écrire un MVC pour chaque vue.
Enfin, je n'ai pas réussi à forcer mon NetBeans à ouvrir des fichiers de test JUnit, menant au report et finalement à l'abandon de la procédure.