DE WIT Apolline
GENIET Florent

pour le 17 mai 2018

Projet informatique
Rapport de synthèse
GESTION D'UNE ACADEMIE

Notre application est un outil de gestion d'une académie. Elle répond à la fois aux besoins logistiques (gestion des salles, répartition des enseignants dans les collèges) et administratifs (entrer et modifier les données personnelles des acteurs de l'académie, les inscrire dans un établissement) du commanditaire. Plus qu'une application pour l'administration, nous avons choisi d'y inscrire des fonctionnalités à l'usage des enseignants et des étudiants pour que chacun ait accès à ses informations personnelles. Elle devient ainsi un outil à l'usage de tous, permettant aux enseignants d'ajouter les notes de leurs étudiants.

1. Bilan du projet

1. Organisation

Nous avons tout d'abord analysé les différents besoins de l'Académie lors des premières séances afin de définir les grandes fonctionnalités de notre programme. Nous avons réalisé les diagrammes UML qui ont clarifié nos objectifs premiers dans l'écriture du code.

Dès la deuxième séance, nous avons commencé à coder les classes générales de notre programme et leurs méthodes présentes dans le package Donnees tout en continuant l'analyse afin de préciser les possibilités de chaque utilisateur de l'application.

Ensuite, nous avons écrit le package User, qui implémente les actions possibles pour les utilisateurs de l'application, ainsi que le main pour tester les méthodes de notre programme et coder l'enchaînement des différentes fonctions..

Enfin, nous nous sommes à nouveau partagé le travail, l'un de nous continuant d'ajouter des méthodes au programme, tandis que l'autre réalisait une interface bureautique.

Sa réalisation a conduit à modifier certaines de nos hypothèses.

Pour faciliter l'avancée de notre travail, l'interface graphique étant très dépendante du code nous avons utilisé Github afin de partager facilement nos réalisations.

2. Objectifs réalisés et apports personnels du sujet

Nous avons réussi à implémenter les fonctionnalités attendues du commanditaire. Celui-ci peut facilement interagir avec la base de données, et peut ainsi y entrer et modifier les informations concernant les enseignants et les étudiants. Nous avons appris à utiliser PgAdmin 4.

Nous avons découvert le logiciel Modelio pour réaliser nos diagrammes UML.

Comme l'accent était mis sur le calcul de distance, nous avons transformé notre projet en projet Maven afin de gérer les dépendances et de pouvoir utiliser l'API Google. Cela nous a posé quelques difficultés, et nous a donné un premier aperçu de l'utilisation d'API avec Java.

Nous avons choisi de réaliser une application bureautique afin de finaliser notre travail et de visualiser toutes les méthodes implémentées. Nous nous sommes familiarisés avec les bibliothèques swing et awt de Java grâce au site internet Openclassrooms. Il nous a permis de trouver les outils correspondant le mieux à nos besoins.

3. Les changements opérés

Pour l'écriture de la base de données, nous n'avons pas trouvé de fichier existant contenant des informations sur les collèges que nous puissions facilement inscrire dans la base de données. Nous avons donc entré quelques collèges choisi arbitrairement, ainsi que des étudiants et professeurs, pour tester notre application.

Il y a eu de nombreux changements effectués par rapport à nos diagrammes UML. Nous avons tout d'abord rajouté un package Connection qui gère la connexion à la base de données afin de faciliter son accès, très fréquent dans le code.

Et surtout, le choix de créer une application bureautique avec une interface graphique développée, regroupant la plupart des fonctionnalités proposées, a considérablement complexifié notre code. Nous avons pour cela rajouté de nombreux packages et classes. Nous avons du changer les arguments de certaines méthodes, et la façon dont on utilise les constructeurs : au lieu de rentrer le nom et le prénom, nous utilisons l'id des utilisateurs. Nous avons simulé le fonctionnement des constructeurs utilisant le nom d'une entité dans le code bureautique.

Nous pensions tout d'abord gérer la connexion selon 3 statuts : administrateur, étudiant ou enseignant et créer une option « gérer le département » dans la fenêtre des options de l'enseignant. Nous avons finalement opté pour une connexion spécifique en tant que responsable du département, car nous supposons que les actions réalisées sont suffisamment différentes d'un statut à l'autre pour que l'enseignant responsable d'un département ne les effectuent pas en même temps. Cela évite ainsi d'avoir une option « Gérer un département » inutile pour les enseignants qui ne sont pas responsables d'un département.

4. Améliorations

Dans les fonctions de l'administrateur, les fenêtres de choix d'un étudiant ou d'un enseignant auraient pu hériter de classes abstraites, de par leurs similarités.

Nous aurions pu gérer davantage les erreurs en vérifiant que les données rentrées sont au bon format.

Nous n'avons pas eu le temps de développer toutes les fonctionnalités de l'application bureautique. Seul un administrateur a accès à toutes les options prévues.

2. Fonctionnement

1. Exécution du programme

Pour exécuter notre programme, il faut choisir comme workspace l'emplacement de notre projet (ou le placer dans le workspace de l'application Java). Ensuite, il faut importer notre projet Maven (« Existing Maven Projects »). Dans la fenêtre suivante, il faut choisir « Academie » qui se trouve dans « ProjetInfo » dans le Root Directory.

Ensuite, il faut aller dans Build Path → Configure Build Path, sélectionner « postgresql-42.2.2.jar » et le supprimer (Remove). Puis, dans Add External JARs, il faut sélectionner l'archive postgresql-42.2.2 jointe au projet.

2. Description des packages

Package AppliBureautique :

Ce package regroupe les premières fenêtres de l'application bureautique. On y trouve la fenêtre de connexion (classe Fen) à l'application selon le statut de l'utilisateur (la classe Fen) ainsi que les fenêtres correspondant aux options qui s'offrent aux utilisateurs en fonction de leur statut : les classes FenAdmin, FenEleve, FenEns (pour l'enseignant) et FenRespo (pour le responsable d'un département).

Package ConnectionJdbc :

Ce package ne comporte qu'une classe ConnectionJdbc qui est la connexion (connection en anglais) à la base de données. Cela simplifie l'écriture du code car elle est souvent utilisée.

Package Donnees :

Ce package regroupe les 8 classes dont il est question dans notre sujet. Il permet de récupérer les données dans la base de données et de les organiser.

Packages FonctionAdmin, FonctionEleve, FonctionEnseignant, FonctionRespo :

Ces packages de l'application bureautique correspondent aux choix proposés à l'utilisateur selon son statut lorsqu'il se connecte. Ce sont les différentes fenêtres qui s'ouvrent et interagissent avec la base de données selon ce que l'utilisateur effectue comme action.

Package Main :

Ce package ne comporte que la classe Main qui permet de lancer le programme. Il ouvre l'application bureautique. Nous avons conservé le main initial en commentaire, qui permet de tester dans la console Java notre programme.

Package User :

Ce package définit les méthodes de chaque utilisateur de l'application, en lien avec le package Donnees, et avec la base de données.

3. Conclusion

Le développement de notre projet nous a amené à réaliser une interface graphique afin de le mener à son terme. Nous avons pu voir la complexité du développement d'une application utilisant une navigation entre plusieurs fenêtres. Cela nous a conduit à écrire de nombreuses classes, ce qui peut très vite désorganiser le code et le rendre confus.