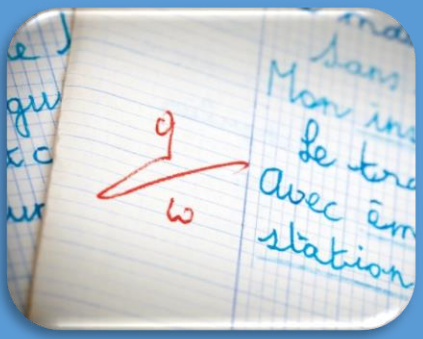
PROJET UML

Gestion de notes des étudiants d'une école d'ingénieurs



FAILLA Romain
ZUBER Thibault

Promo 2018

Table des matières

Choix logiciel

StarUML

StarUML est un logiciel de modélisation UML, il gère la plupart des diagrammes spécifiés dans la norme UML 2.0.

Une version 2.0 est proposée, nous avons d'ailleurs utilisé celle-là.

Qt Creator

Qt Creator est un environnement de développement intégré multiplate-forme faisant partie du framework Qt. Il est donc orienté pour la programmation en C++.

Il intègre directement dans l'interface un débogueur, un outil de création d'interfaces graphiques, des outils pour la publication de code sur Git et Mercurial ainsi que la documentation Qt. L'éditeur de texte intégré permet l'autocomplétion ainsi que la coloration syntaxique. Qt Creator utilise sous Linux le compilateur gcc. Il peut utiliser MinGW ou le compilateur de Visual Studio sous Windows.

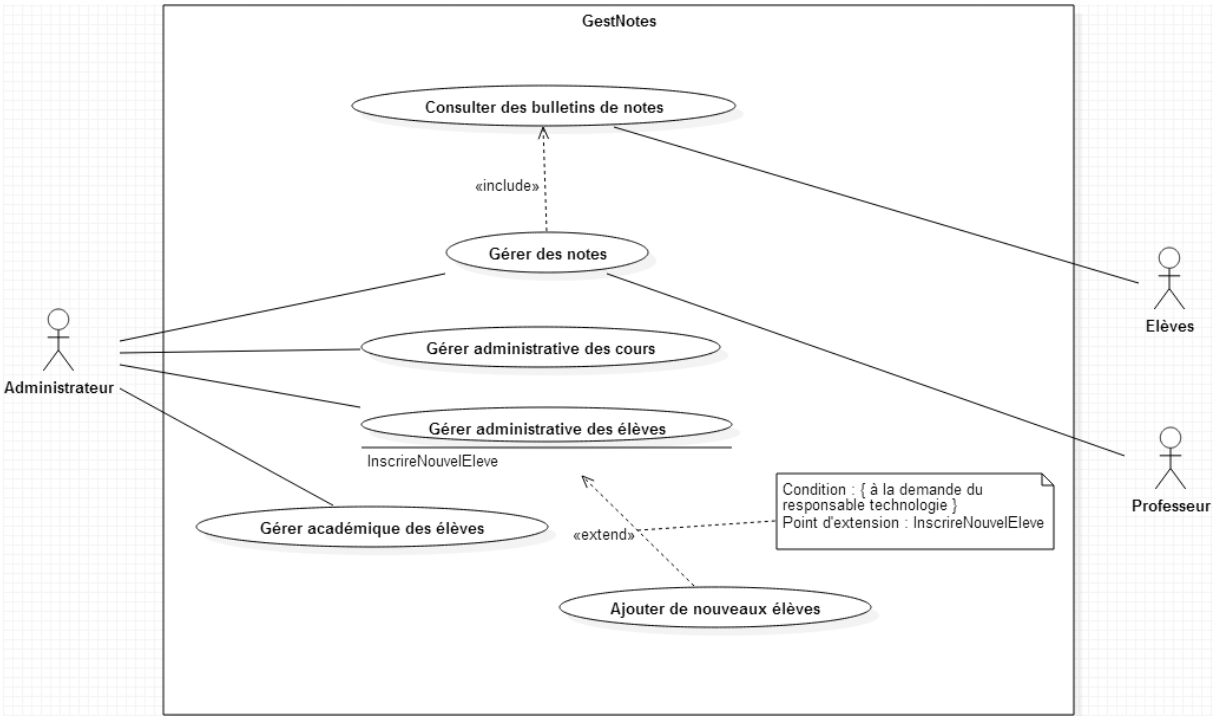
Les diagrammes UML

UML est utilisé pour spécifier, visualiser, modifier et construire les documents nécessaires au bon développement d'un logiciel orienté objet. UML offre un standard de modélisation, pour représenter l'architecture logicielle. Les différents éléments représentables sont : les activités d'un objet/logiciel, les acteurs, les processus, les schémas de base de données, les composants logiciels

Afin de réaliser les différents diagrammes UML nécessaires à l'étude de ce projet, nous avons utilisé StarUML. Nous avons modélisé les diagrammes les plus importants tels que cas d'utilisation ou classes.

Diagrammes de cas d'utilisation

Le diagramme de cas d'utilisation va nous permettre de donner une vision globale du logiciel que l'on souhaite faire fonctionner. Il va nous permettre de voir les liaisons entre les différents cas d'utilisation et les acteurs.



Nous pouvons nous rendre compte qu'il y a trois acteurs, l'administrateur, le professeur et l'élève.

L'administrateur possède tous les pouvoirs, la gestion des notes et des élèves ainsi l'ajout de nouveaux élèves. Les professeurs eux ne peuvent que gérer les notes des élèves. Les élèves ne peuvent que consulter leurs propres notes.

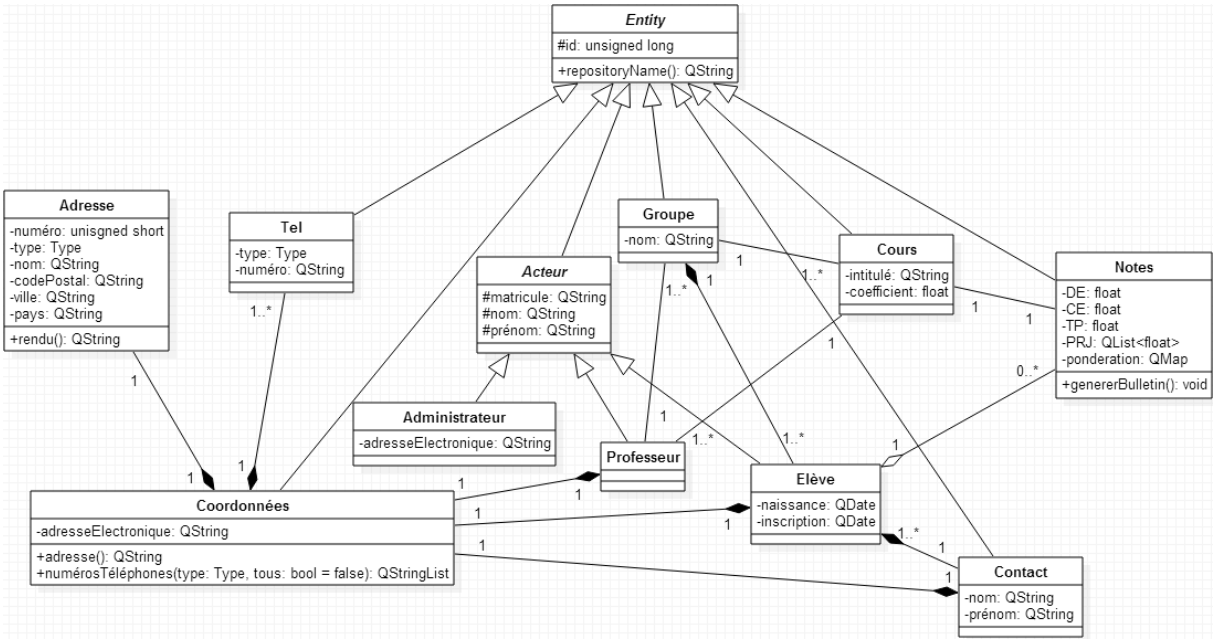
Modèle de la base de données

Le diagramme de classes est considéré comme le plus important de la modélisation orientée objet, il est le seul obligatoire lors d'une telle modélisation.

Il s'agit d'une vue statique car on ne tient pas compte du facteur temporel dans le comportement du système. Le diagramme de classes modélise les concepts du domaine d'application ainsi que les concepts internes créés de toutes pièces dans le cadre de l'implémentation d'une application.

Les principaux éléments de cette vue statique sont les classes et leurs relations : association, généralisation et plusieurs types de dépendances, telles que la réalisation et l'utilisation.

C'est le diagramme de classe que nous allons respecter pour la base de données.



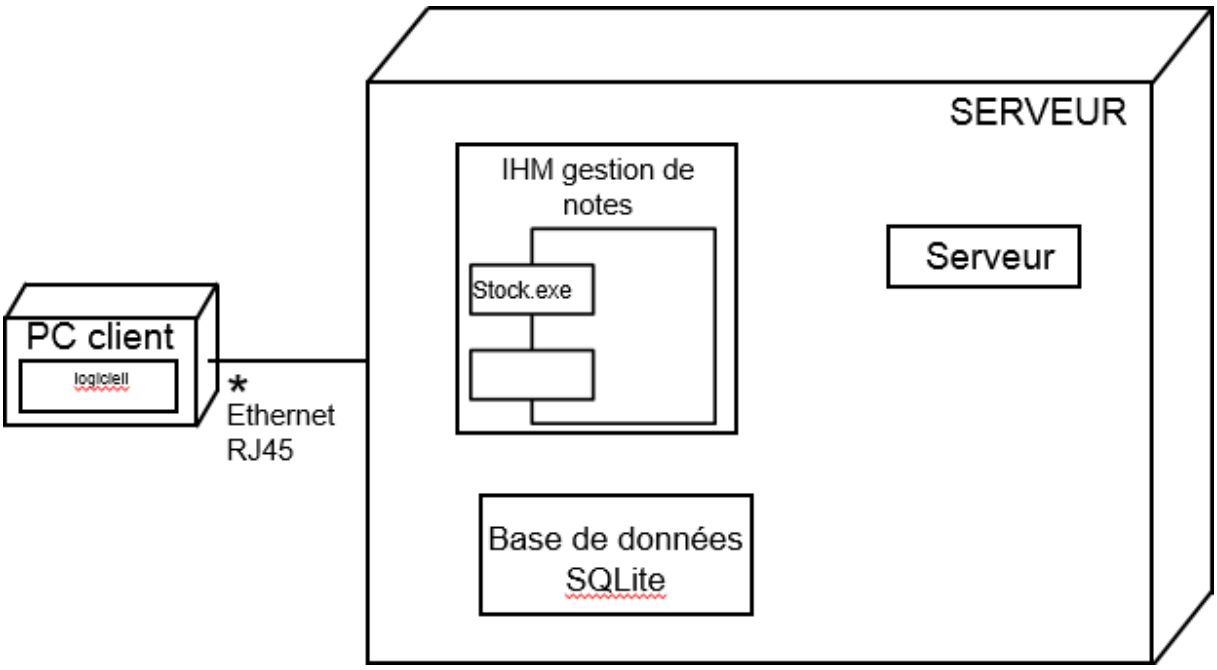
Ici nous pouvons remarquer que la plupart des classes héritent d'une même classe *Entity*.

De plus, la classe *coordonnées* est une classe importante car si cette classe est « détruite », les relation de composition font que les classe *adresse* et *tel* disparaîtront. Il en est de même pour cette dernière, si les classes *Professeur*, *Elève* ou *Contact* sont détruitent, la classe *coordonnées* n'est plus.

Il ne peut y avoir qu'une seule adresse pour une « coordonnée », mais aussi, qu'un seul professeur, élève ou contact pour une « coordonée ».

Diagramme de déploiement

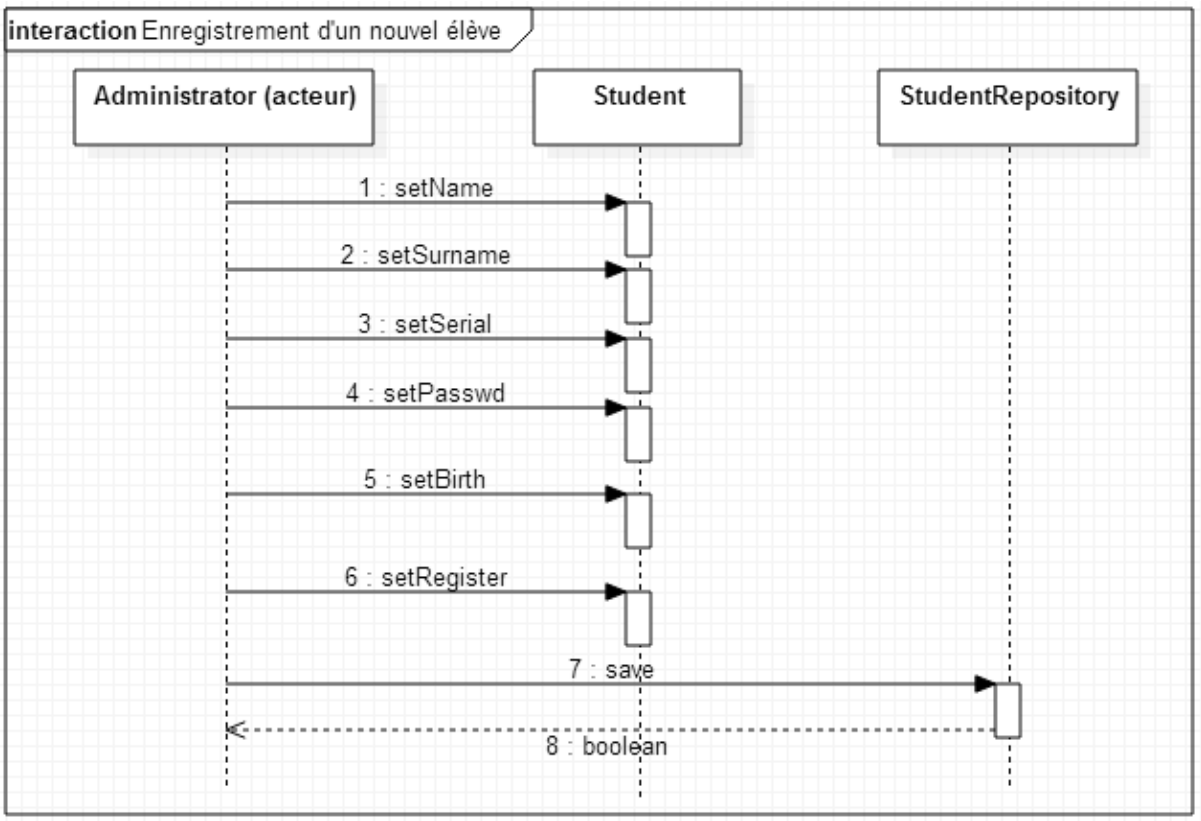
Un diagramme de déploiement décrit la disposition physique des ressources matérielles qui composent le système et montre la répartition des composants sur ces matériels. Chaque ressource étant matérialisée par un noeud, le diagramme de déploiement précise comment les composants sont répartis sur les noeuds et quelles sont les connexions entre les composants ou les noeuds.



Ici, les différentes ressources matérielles utilisées pour accéder au « logiciel » sont les PC clients relier par des cables RJ45. Notre base de donnée (sous SQLite) ainsi que l'IHM « gestion de notes » sont lié à un serveur principal.

Diagramme de séquence

C’est digramme d’interaction qui permet de représenter les collaborations entre objets selon un point de vue temporel en mettant l'accent sur la chronologie des envois de messages.

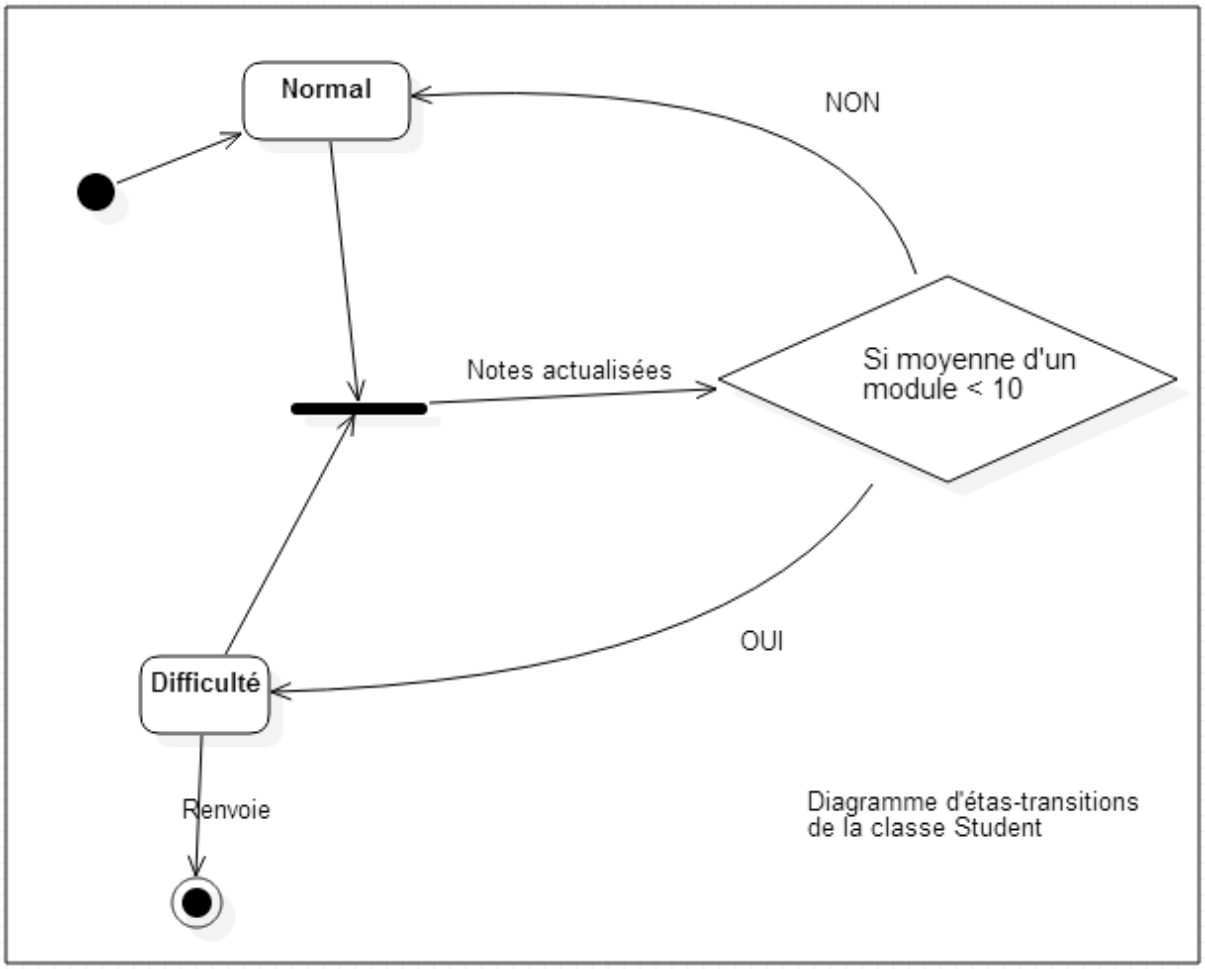


Ici, notre diagramme de séquence porte sur l’enregistrement d’un élève dans la base de sonnées. Ainsi, l’administrateur insère toutes les informations dont il a besoin pour rendre l’élève légitime ; c’est-à-dire le nom, le prénom, le matricule, le mot de passe et la date d’enregistrement.

Diagramme d’états-transitions

Les digrammes d’états-transitions permettent de décrire les changements d'états d'un objet ou d'un composant, en réponse aux interactions avec d'autres objets/composants ou avec des acteurs.

Un digramme d’états-transitions se focalise sur le comportement des objets guidé par l’occurrence d’événements.



Ici nous pouvons apercevoir l’état d’un élève normal au départ. Si sa moyenne de module est supérieure à 10 alors l’élève reste « normal ». Si l’élève à une moyenne de module inférieure à 10 alors l’état de l’élève devient élève en « difficulté ». Ainsi quand l’élève passe en état « difficulté », le professeur tuteur est averti immédiatement.

Développement informatique

Une fois les bases théoriques posées grâce à la conception UML du projet, nous avons pu nous tourner vers le développement informatique dans toute sa splendeur. Pour se faire, nous avons tâché de nous rapprocher du pattern MVC qui nous a permis de séparer le développement de la gestion de la base de données de la conception de l'interface graphique.

Conception du modèle

La plus grande part du développement informatique a été orientée vers la mise en œuvre du modèle conçu précédemment. En effet, nous nous sommes du fonctionnement d'ORM, comme Doctrine ORM, pour la mise en place du modèle. Avec cette conception, nous avons distingué le répertoire, c'est-à-dire la table de données de la BDD, de l'entité, une ligne de la table sélectionnée.

Le répertoire n'est pas présent dans notre modèle car, c'est lui qui gère le chargement et la persistance des données ; c'est lui qui, par l'intermédiaire de requêtes SQL, va charger les données de la table et les modifier. Pour ce faire, il va se servir des entités qui vont stocker les informations de chaque ligne traitée à l'issue des requêtes. Le répertoire va également permettre d'effectuer des recherches dans la table sans exécuter de requête SQL. En effet, une fois que les entités sont créées, elles sont stockées dans une liste possédée par le répertoire. Il y a un répertoire par type d'entité.

Les entités, quant à elle, respecte le modèle UML précédemment créé. Elles ont simplement des méthodes d'accès et de modifications aux attributs de la ligne représentées et des méthodes qui permette de faciliter la génération d'une adresse postale par exemple. Toutes les entités qui sont utilisées héritent d'une classe abstraite sobrement appelée « Entity » qui leur accorde un attribut _id qui indique la position de la ligne dans la table.

Cette architecture nous a permis d'avoir une utilisation des données stockées au sein de la base de donnée très simple car, nous n'utilisons aucune autre requête SQL après l'implémentation du mécanisme de répertoire et nous permettre de rester concentrer sur des aspects plus spécifiques à la vue ou au contrôleur.

Conclusion

En définitive, ce projet nous a permis de découvrir une utilisation concrète de la norme UML. En effet, une grande partie du projet a été un travail d'analyse qui a permis de tirer des modèles et des schémas de conception. Ces derniers nous ont permis de formaliser notre développement et de ne pas partir dans toutes les directions.

Bien que nous ayons mal géré notre temps, nous avons pu toutefois voir l'efficacité de la méthode UML en implémentant, seulement à partir de nos modèles, l'ensemble de la base de données en un temps record ce qui nous a permis d'avoir un exécutable « potable » lors de la soutenance orale. Grâce à cette expérience, nous pensons que la norme UML est indéniablement nécessaire pour n'importe quel type de projet, que ce soit en informatique ou dans un autre domaine.