**Gestion de l’emploi du temps de l’université 21/03/12**

[Sous-titre du document]

Anthony Morales, Chen-Yang Gao, Irvin genieys, Simon Maby

**Résumé**

Rapport de projet de conception et d’implémentation de la solution pour la gestion de l’emploi du temps de l’université

Table des matières

I. Introduction 4

II. Conception de la Base de donnée 4

a. Modèle conceptuel des données 4

b. Vues et types 4

III. Conception logiciel 5

a. Diagramme des classes 5

b. Diagrammes de séquence 5

c. Persistance des données 9

IV. Rapport technique 10

V. Tests 11

VI. Résultats 12

VII. Discussion 13

VIII. Conclusion 14

IX. Annexes 15

# Introduction

# Conception de la Base de donnée

## Modèle conceptuel des données

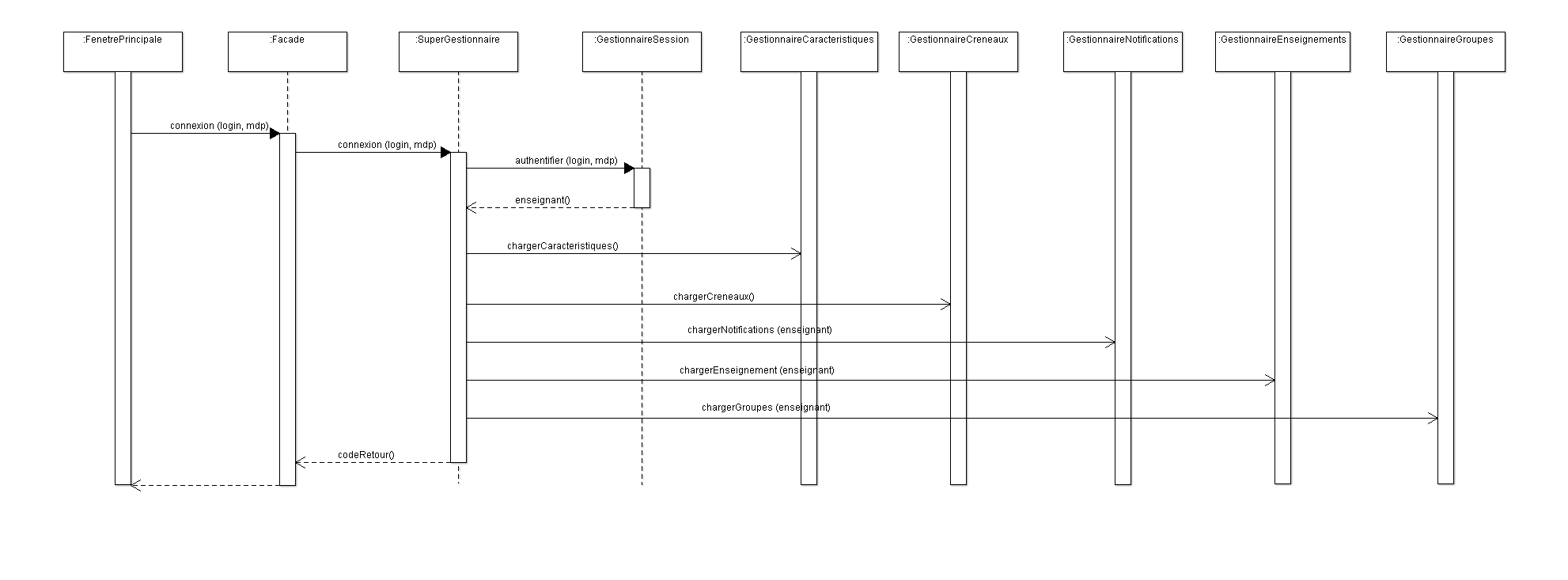
## Vues et types

# Conception logiciel

## Diagramme des classes

## Diagrammes de séquence

Connexion :

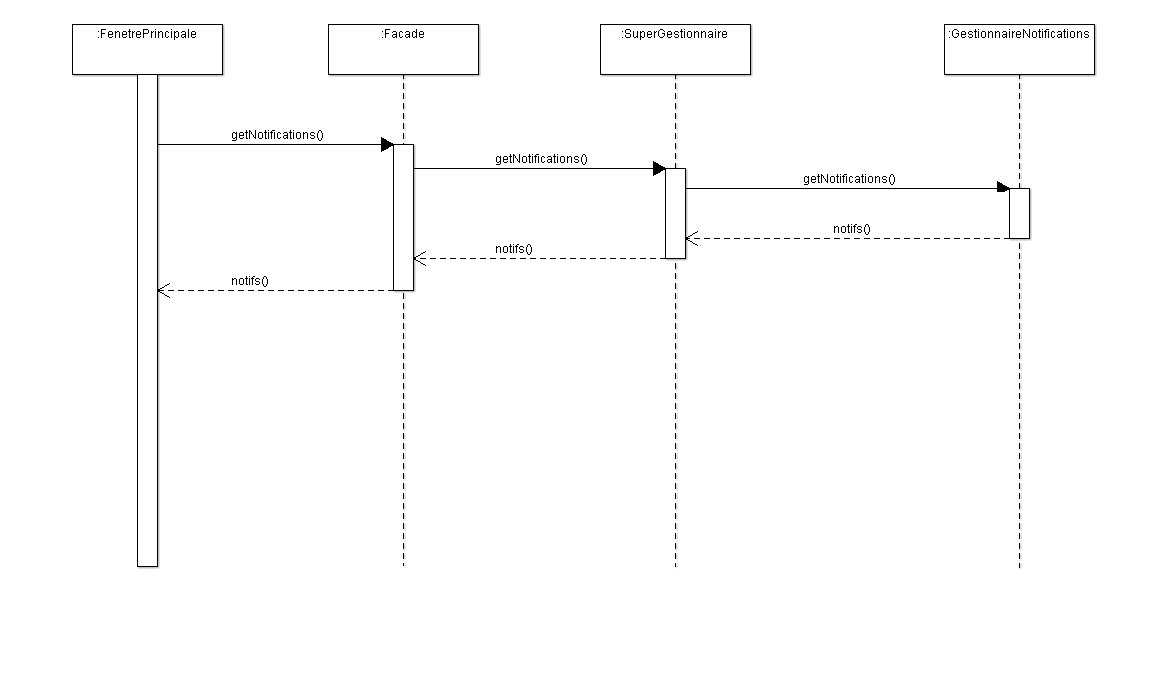


Pour se connecter un utilisateur, la classe FenetrePrincipale appelle chez la façade la méthode « connexion » dont le seul but est de demander au super gestionnaire la connexion de l’utilisateur en lui donnant son identifiant et son mot de passe. Le super gestionnaire prend ensuite le relais et demande au gestionnaire de session d’authentifier l’utilisateur en allant vérifier dans les données persistantes l’existence de celui-ci et le cas échéant construit un objet Enseignant et le donne au super gestionnaire.

Si le gestionnaire de session a accompli l’authentification avec succès nous allons charger en mémoire les données dont aura besoin l’utilisateur et qui ne changerons pas au cours de l’exécution c’est-à-dire les caractéristiques des salles, les créneaux, les enseignements, les groupes et les notifications de l’utilisateur. Ces chargements sont délégués par le super gestionnaire aux gestionnaires responsables de ces objets.

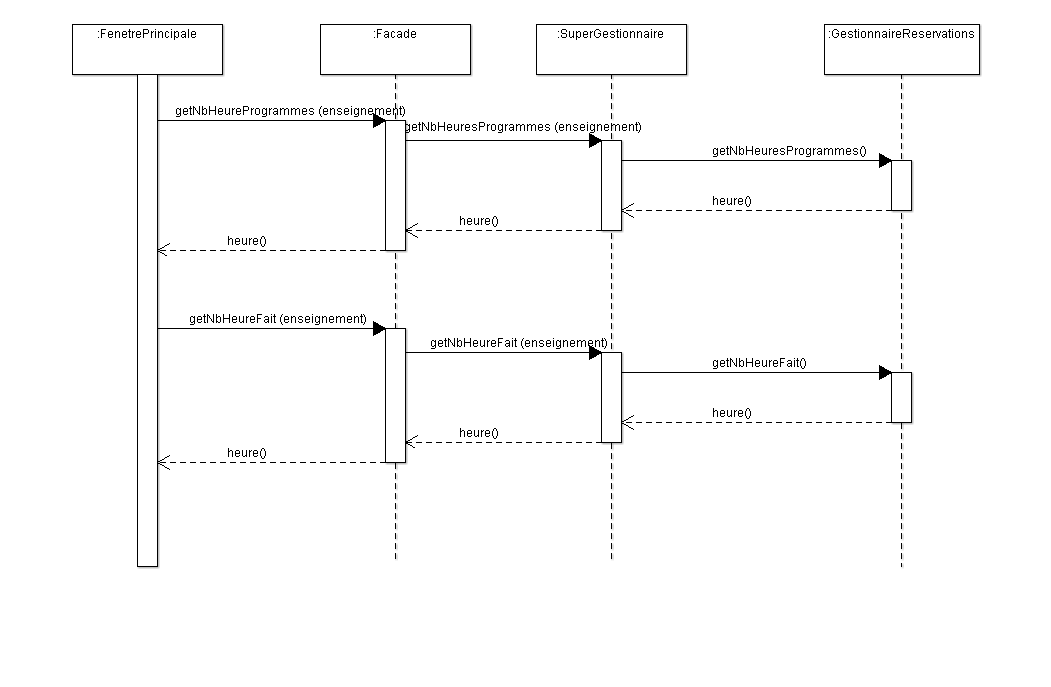
Après cela le super gestionnaire examine l’enseignant retourné par le gestionnaire de session et envoi un code de retour à la façade pour qu’elle puisse communiquer à la fenêtre principale si l’authentification a réussi et si oui, si l’enseignant est une responsable.

Visualisation des notifications :



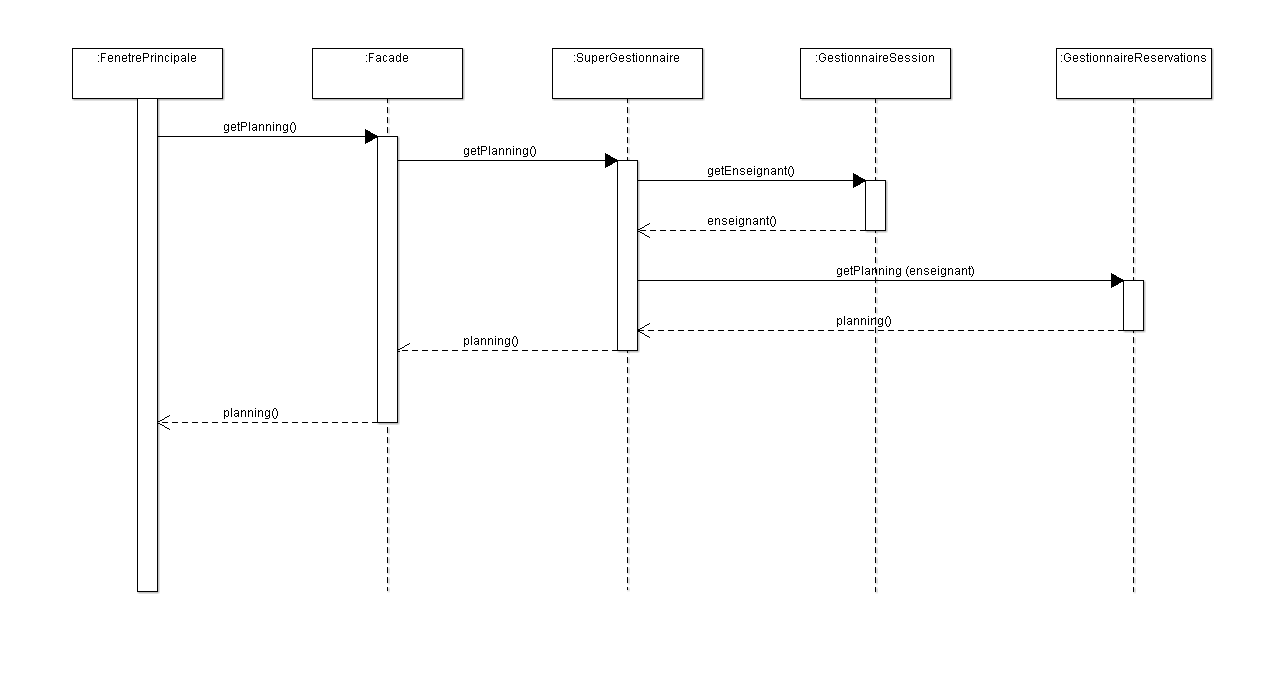
Pour visualiser les notifications, l’interface utilisateur va demander à la façade la liste des notifications de l’enseignant via la méthode getNotifications(). La façade n’étant qu’un point d’entrée pour l’utilisation des classes métiers elle demande simplement au super gestionnaire de lui donner la liste des notifications de l’enseignant connecté. Etant donné que les notifications font partie des objets que nous chargeons dès la connexion le super gestionnaire n’a qu’à les récupérer en les demandant au gestionnaire de notification qui garde ces connexions en mémoire depuis la connexion.

Visualisation de l’état du service :



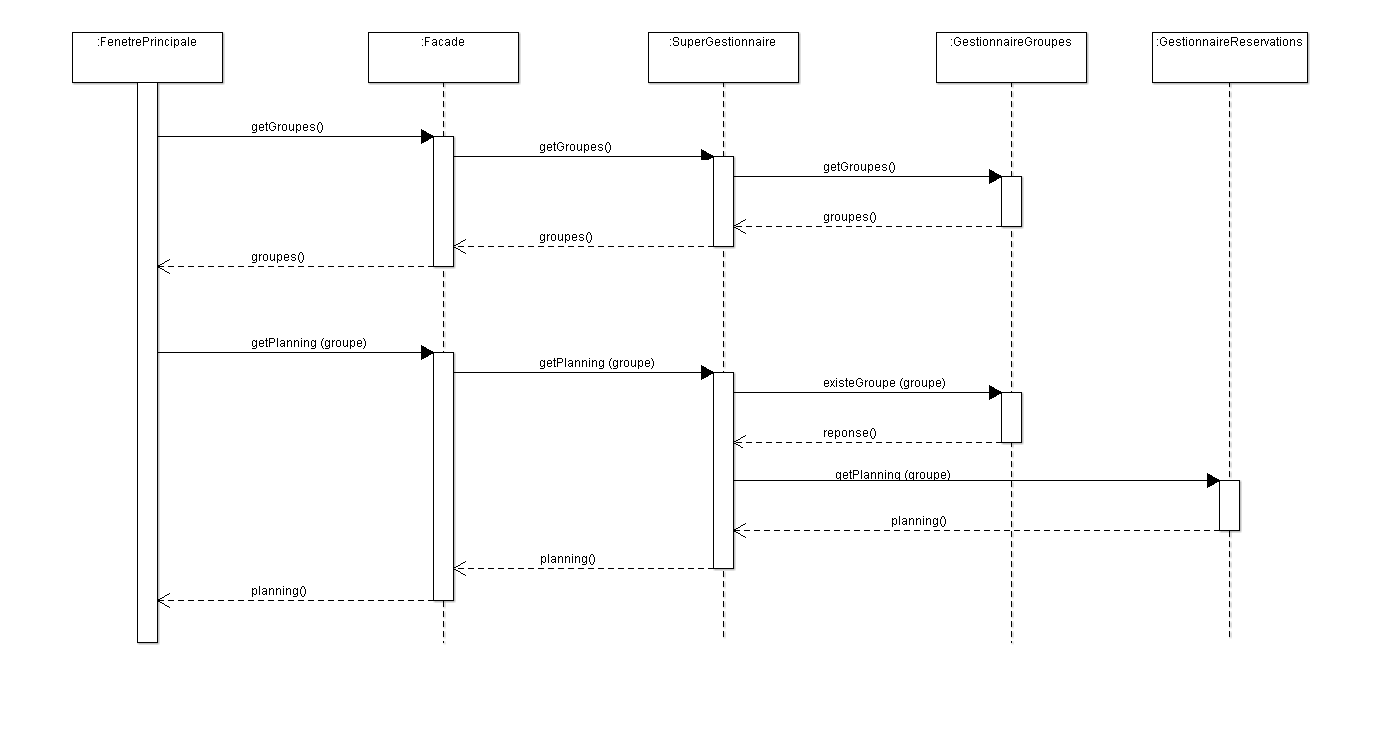
Pour visualiser l’état du service d’un enseignant, la façade fournit les méthodes getNbHeureProgramme et getNbHeureFait qui donnent respectivement pour un enseignement le nombre d’heure qui est programmé dans le planning et le nombre d’heure déjà effectué. Pour utiliser ces fonctionnalités l’interface utilisateur doit préalablement demander la liste des enseignements grâce à la méthode getEnseignements qui donnent les enseignements de l’enseignant connecté, en effet ces enseignements ont été chargés à la connexion dans le gestionnaire d’eneignements.

Visualisation du planning :



Pour visualiser le planning de l’enseignant connecté, l’interface utilisateur dispose de la methode getPlanning dans la facade qui appelle simplement la methode getPlanning chez le super gestionnaire. Le super gestionnaire demande ensuite au gestionnaire de session de lui donner l’objet représentant l’enseignant connecté et demande au gestionnaire de reservation de lui donner toutes les réservations programmées de cet enseignant. Pour cela le gestionnaire de réservations va chercher dans les données persistantes ces réservations et les donne au super gestionnaire qui va les fournir à l’interface utilisateur via la facade. L’interface utilisateur pourra ensuite afficher à l’utilisateur son planning grâce à la liste de réservations reçues.

Visualisation du planning d’un groupe :



L’application doit aussi permettre d’afficher le planning d’un groupe d’étudiant. Comme nous avons pu le voir dans la description de la connexion, le super gestionnaire demande, lors de l’authentification de l’enseignant, au gestionnaire de groupes de charger en mémoire tous les groupes de l’enseignant connecté. L’interface utilisateur peut donc ensuite demander ces groupes grâce à la méthode getGroupes qui récupère ces données. La façade fournit une méthode getPlanning qui attend un groupe en paramètre et demande au super gestionnaire le planning de ce groupe. Celui va ensuite vérifier si le groupe donné existe bien parmi les groupes chargés et demandera ensuite le planning de ce groupe au gestionnaire de réservations.

Comme pour la visualisation du planning personnel l’interface utilisateur pourra afficher le planning de ce groupe grâce à la liste des réservations correspondantes renvoyée par la méthode.

## Persistance des données

# Rapport technique

## Outils utilisés

ArgoUml :

Pour réaliser la conception objet de notre projet nous avons choisi d’utiliser le programme argoUml qui est un outil graphique de conception objet. Ce logiciel nous a permis par la suite de générer automatiquement du code à partir de notre conception ce qui a permis un aide de temps considérable.

Eclipse :

Pour développer notre projet nous avons utilisé l’environnement de développement intégré eclipse. Nous avons choisi cet outils car c’est l’IDE le plus utilisé et le plus fiable pour développer des projets en Java et de plus il fournit de nombreux plug-in qui nous ont été d’une grande aide.

Subversion/Subclipse

Afin de gérer au mieux le travail collaboratif en équipe nous avons décidé d’utiliser le gestionnaire de version subversion. Pour stocker nos données nous avons utilisé l’hébergeur de dépôt svn assembla et notre projet est accessible à l’url <http://subversion.assembla.com/svn/gestionedt/>.

Nous avons de plus choisi d’utiliser le plug-in subclipse d’eclipse qui fournit un client svn intégré à l’éditeur et nous a permis de gérer plus simplement le gestionnaire de version.

Le framework JDBC :

Pour interagir avec les données stockées en mémoire persistante nous avons utilisé le framework java JDBC qui permet une utilisation simple d’une base de données dans un programme java.

Le framework JDOM :

Pour parcourir des fichiers xml nous avons utilisé le framework JDOM qui fournit une API pour la gestion des fichiers xml.

Swing4eclipse :

Afin de faciliter le développement de l’interface utilisateur graphique nous avons utilisé le framework swing4eclipse qui fournit un outil visuel pour développer des interfaces graphiques en java.

Javadoc :

Pour documenter notre code nous avons utilisé l’outil javadoc, de plus eclipse fournit un outil de génération de javadoc.

# Tests

# Résultats

# Discussion

# Conclusion

# Annexes