

Rapport de conception

Projet SLAB

Chen-Yang Gao, Irvin Genieys, Simon Maby, Anthony Morales
22/03/2012

Introduction

Un projet de développement d'une telle application demande de faire des choix de conceptions aiguisés. Nous tenterons d'expliquer ses choix avec autant de rigueur possible afin de fournir un référentiel pour tout développeur actuel ou futur.

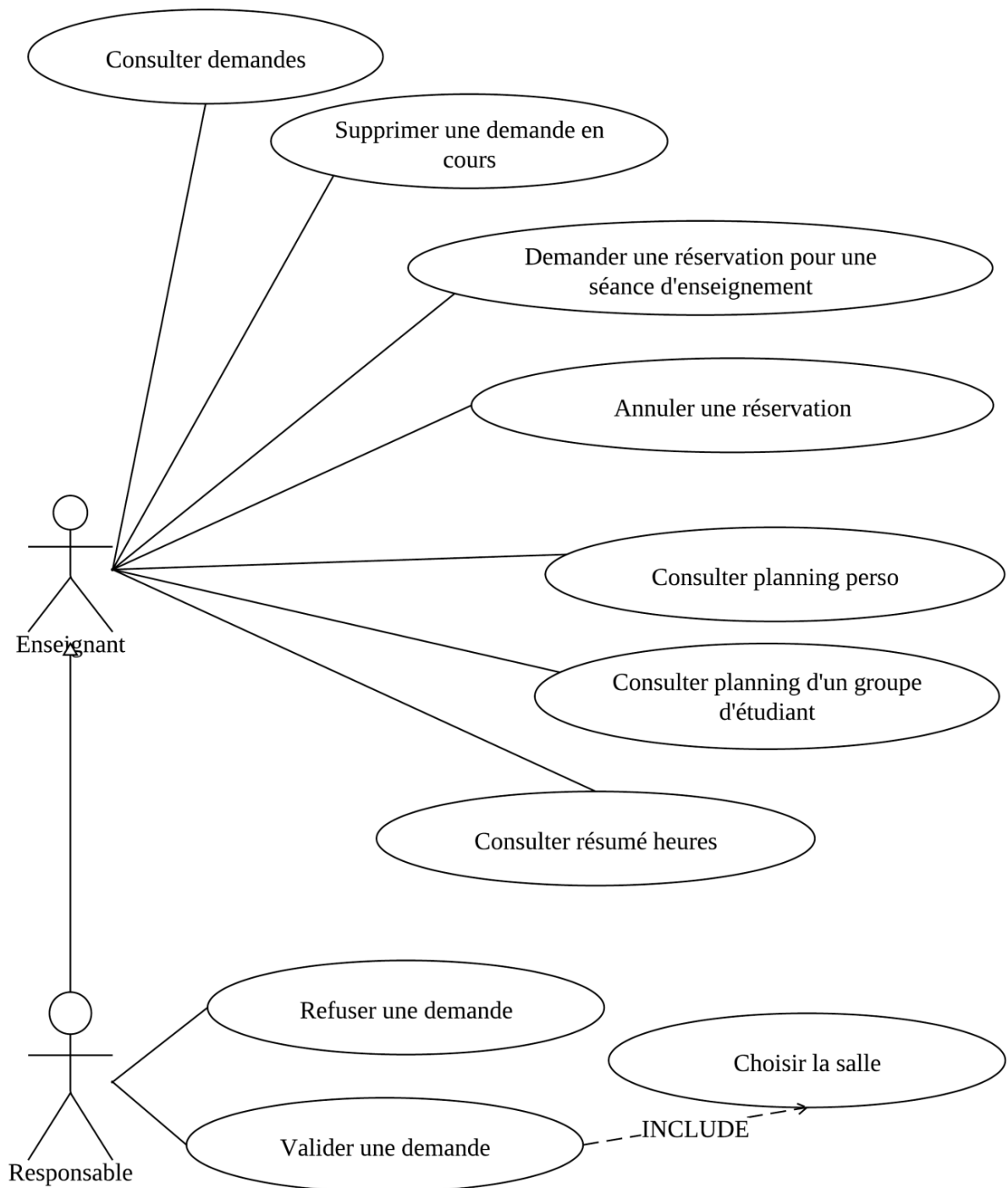
Nous allons vous présenter des cas d'utilisations très détaillés accompagnés de maquettes de la future interface. Ensuite, nous expliquerons le design choisi avec une présentation du diagramme de classe et des diagrammes de séquences. Enfin nous expliquerons nos moyens de persistance des données.

Table des matières

I. Use cases et maquettes	3
Explication des uses cases.....	4
II. Conception logiciel	19
a. Diagramme des classes	19
b. Diagrammes de séquence	23
c. Persistance des données.....	32
III. Conception de la Base de données	33
a. Schéma de la base de données	33
b. Vues et types.....	34
IV. Conclusion.....	35
V. Annexes	36

I. Use cases et maquettes

L'application a pour but de gérer la modification de réservation de salles en cours d'année. Deux utilisateurs possibles : Enseignant et Responsable. Le responsable a accès à toutes les fonctionnalités de l'enseignant plus celle de répondre à des demandes de réservations. Ces cas demandent une identification avec mot de passe.



Explication des uses cases

Nous allons détailler les uses cases enseignant. Chacun de ces use cases sera accompagné d'une maquette décrivant l'interface. L'utilisateur de type responsable possède ces même uses cases, ils ne seront donc pas abordés de nouveau. Les maquettes responsable sont les mêmes que les maquettes enseignant à la différence que la barre de menu contient un choix supplémentaire : Répondre aux demandes, qui est une fonctionnalité propre aux responsables. Nous terminerons en expliquant ce dernier cas, évidemment accompagné de la maquette associée ce qui montrera la différence évoquée concernant le menu.

Login

Identification avec mot de passe pour accès à l'application

Déroulement L'utilisateur rentre son Nom, son prénom, et son mot de passe dans les champs dédiés et se connecte.

Déroulement alternatif Si un ou plusieurs champs est incorrect un message d'erreur est affiché.

Post-conditions L'utilisateur est loggé sur le système. Si cet utilisateur est un enseignant il a accès aux fonctionnalités enseignant, si c'est un responsable il a accès aux fonctionnalités responsable.

L'écran d'accueil est affiché avec les notifications reçues. L'enseignant peut avoir des notifications de validation ou de refus d'une de ses demandes. Le responsable peut avoir des notifications de réception d'une nouvelle demande de la part d'un enseignant.

Evidemment l'interface chargée dépend du type d'utilisateur. La seule différence réside dans l'ajout dans le menu d'une fonctionnalité Répondre aux demandes pour le type responsable.

Accueil	Demandes	Supprimer une séance	Planings	Etat de mon service	Deconnexion
---------	----------	----------------------	----------	---------------------	-------------



Bienvenue sur l'espace de gestion des emplois du temps



- o Votre demande du 24/12 a été validée
- o Votre cours de Statistiques du 12/11/2013 avec les IG4 a été supprimé

Voir

Consulter mes demandes

L'utilisateur consulte les demandes qu'il a faite auparavant, qu'elles soient validées, refusées ou en cours.

Déroulement

Il clique sur Demandes qui regroupe deux onglets. L'onglet par défaut est Mes demandes et satisfait ce cas d'utilisation. Cet onglet est un bouton qui amène sur un écran affichant en liste les demandes effectuées dans l'ordre chronologique ascendant. La demande est résumée par :

- ✓ caractéristiques de salles
- ✓ groupe d'étudiant
- ✓ l'heure de début et l'heure de fin du créneau
- ✓ la date de la séance demandée

La date d'émission de la demande est indiquée ainsi que son statut :

- ✓ Validée
- ✓ En-cours
- ✓ Refusée

Enfin un bouton supprimer est à disposition pour supprimer la demande.

Déroulement alternatif

Post-conditions

Accueil

Demandes

Supprimer une séance

Plannings

Etat de mon service

Deconnexion

Mes demandes Faire une nouvelle demande

Demande

Salle TD pour IG 4 Groupe 1 le 17/12/2013 de 8h à 9h30

Salle Cours avec rétroprojecteur pour IG 4 le 19/12/2013 de 8h à 9h30

Date demande

10/11/2013

3/11/2013

Statut

En-cours

Validée

X supprimer

X supprimer

L'utilisateur fait une demande destinée aux responsables qui le concernent.

Déroulement Il clique sur « Demandes » qui regroupe deux onglets. L'onglet par défaut est Mes demandes. Il doit donc cliquer sur l'onglet « faire une nouvelle demande ». Cet onglet est un bouton qui amène sur un écran divisé en trois parties.

La première partie permet de créer la demande. Cinq listes déroulantes s'affichent :

- ✓ Caractéristiques
- ✓ Créneau
- ✓ Date
- ✓ Etudiants
- ✓ Matière

Les valeurs possibles pour chacune de ces listes dépendent du contenu défini par le dictionnaire de donnée du système d'information de l'université. Cependant ces valeurs sont restreintes selon l'enseignant connecté. Pour la liste étudiants les seules valeurs à afficher sont les étudiants auxquels il enseigne. Pour la liste matières, les seules valeurs à afficher sont celles qu'il enseigne.

Enfin, pour chaque liste une seule valeur est possible sauf pour les caractéristiques, qui sont cumulées.

La deuxième partie affiche le résumé de la demande qu'il est en train de créer.

La troisième partie est un bouton envoyer qui soumet la demande aux responsables concernés et leur envoie une notification de nouvelle demande.

Déroulement alternatif	Si la demande est incomplète un message d'erreur est affiché. Une demande est incomplète si il : <ul style="list-style-type: none">✓ Manque un créneau✓ Manque une date✓ Manque un groupe d'étudiant✓ Manque une matière Seul le champ caractéristique peut être nul ce qui correspond à aucune exigence spéciale.
-------------------------------	---

Post-conditions	Un message « votre demande a été envoyée » est affiché en cas de succès. Et la demande faite sera affichée dans l'onglet « Mes demandes »
------------------------	---

Mes demandes

Faire une nouvelle demande

Sélection



Caractéristiques

Rétroprojecteur

Ordinateur

Tableau interactif



Créneau



Date



Etudiants



Matière

La demande :

Salle cours

Rétroprojecteur

8h - 9h 30

IG 4 classe entière

Statistiques 2 TD

4 FEB 2008						
S	M	T	W	T	F	S
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	

Envoyer

Annuler une réservation

L'utilisateur supprime une de ses séances déjà programmée dans l'emploi du temps.

Déroulement Il clique sur Supprimer une séance. Un écran s'affiche, il se divise en deux parties. La première partie est un outil de sélection permettant de trouver parmi la multitude de séances celle que l'utilisateur veut supprimer. Quatre listes déroulantes s'affichent :

- ✓ Créneau
- ✓ Date
- ✓ Etudiants
- ✓ Matière

Les valeurs possibles de sélection sont les mêmes que pour le use case précédent.

Les critères de recherche sont résumés en dessous. Tous les critères sont acceptés, il n'est pas nécessaire d'effectuer une sélection sur toute les listes.

Un bouton rechercher permet de trouver les séances de l'enseignant dans l'emploi du temps qui correspondent aux critères.

Les résultats sont affichés dans un tableau offrant la possibilité de supprimer directement l'une des séances avec un bouton supprimer. Les séances sont résumées par leur date, leur matière, leur groupe d'étudiant, leur salle, et leur date de début et date de fin de leur créneau.

Déroulement alternatif	Si aucun résultat n'est disponible un message le précise
-------------------------------	--

Post-conditions

Gestion EDT

?

0

x

Accueil

Demandes


Supprimer une séance


Plannings


Etat de mon service


Deconnexion

Recherche

Matière

Etudiants

Créneau

Date

Statistiques 2

IG 4 classe entière

Rechercher

Résultats


12/ 11 / 13 Statistiques 2 IG 4 salle 101 8h-9h30

14/ 11 / 13 Salle 201

supprimer

supprimer

Un souci?



Voir mon planning

Consulter planning perso

L'utilisateur consulte son emploi du temps personnel.

Déroulement Il clique sur Plannings qui regroupe deux onglets. L'onglet par défaut est « Mon planning » et satisfait ce cas d'utilisation. Cet onglet est un bouton qui amène sur un écran affichant un emploi du temps classique hebdomadaire. Deux boutons fléchés semaine suivante et semaine précédente permettent d'afficher les emplois du temps associés. Un bouton de sélection d'une date particulière permet de sauter directement à la semaine contenant la date sélectionnée.

Déroulement alternatif

Post-conditions

Consulter planning perso

L'utilisateur consulte l'emploi du temps d'un groupe d'étudiant, notamment dans le but de vérifier leur disponibilité pour faire une demande de réservation intelligente.

Déroulement Il clique sur Plannings qui regroupe deux onglets. L'onglet par défaut est « Mon planning ». Il faut donc cliquer sur l'onglet « planning d'un groupe d'étudiant ». Cet onglet est un bouton qui amène sur un écran affichant un emploi du temps classique hebdomadaire. Deux boutons fléchés semaine suivante et semaine précédente permettent d'afficher les emplois du temps associés. Un bouton de sélection d'une date particulière permet de sauter directement à la semaine contenant la date sélectionnée.

Enfin, une liste déroulante nommée étudiants permet de sélectionner un groupe d'étudiant. Les valeurs sont limitées aux étudiants auxquels l'enseignant enseigne.

Déroulement alternatif

Post-conditions

Mon planning

Planning d'un groupe d'étudiants

12/ 03/ 2012



semaine

précédente



semaine

suivante

00MU1

08h30	lundi	mardi	mercredi	jeudi	vendredi
09h30	GEST RELAT COMMERCIA	FRANCAIS	ECONOMIE GENERALE	INFORMATIQ COMMERCIA	DROIT
10h30		ANGLAIS LV1		INFORMATIQ COMMERCIA	
10h45					
11h30	DEVELOP UNIT COMMERC	ECONOMIE D'ENTREPRISE	ANGLAIS LV1	GEST RELAT COMMERCIA	COMMUNICATION
12h30					ANGLAIS LV1
13h30					COMMUNICATION
13h45					
14h30	MANAG GEST UNIT COMM	GEST RELAT COMMERCIA	ALLEMAND LV2		
15h30		ESPAGNOL LV2	ESPAGNOL LV2		
16h30	FRANCAIS	ESPAGNOL LV2	MANAG GEST UNIT COMM		
17h30		DEVELOP UNIT COMMERC			
18h30					
19h30					
20h30					

Gestion EDT

?

0

x

Accueil

Demandes

Supprimer une séance

Planings

Etat de mon service

Déconnexion

Mon planning

Planning d'un groupe d'étudiants

12/03/2012

Etudiants

↑

semaine précédente

↓

semaine suivante

00MIU1

	lundi	mardi	mercredi	jeudi	vendredi
8h30	GEST RELAT COMMERCIA	FRANCAIS	ECONOMIE GENERALE	INFORMATIQ COMMERCIA	DROIT
9h30		ANGLAIS LV1		INFORMATIQ COMMERCIA	
10h30					
10h45					
11h30	DEVELOP UNIT COMMERC	ECONOMIE D'ENTREPRISES	ANGLAIS LV1	GEST RELAT COMMERCIA	COMMUNICATION ANGLAIS LV1 COMMUNICATION ANGLAIS LV1
12h30					
13h30					
13h45					
14h30	MANAG GEST UNIT COMM	GEST RELAT COMMERCIA	ALLEMAND LV2 ESPAGNOL LV2		
14h45		ESPAGNOL LV2			
15h30	FRANCAIS	DEVELOP UNIT COMMERC	MANAG GEST UNIT COMM		
16h30					
17h30					
18h30					

© 2011 Université de la Méditerranée

14 | Rapport de conception

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

Consulter résumé heures

L'utilisateur consulte l'état de son service actuel, c'est-à-dire le nombres d'heures par enseignement effectué, prévus dans l'emploi du temps, et restant à faire.

Déroulement

Il clique sur état de mon service dans le menu.

Un écran s'affiche sous forme de tableau en résumant le nombre à faire, programmées dans l'emploi du temps et les heures faites pour un enseignement donné.

Un enseignement se définit par une matière, un type de cours (TD, TP, cours), des étudiants.

Evidemment ils sont propres à l'enseignant connecté.

Déroulement

alternatif

Post-conditions

[illegible]

Nous avons décrit les uses cases des enseignants, aussi valable pour les responsables. Décrivons maintenant les cas propres aux responsables. La maquette associée sera l'occasion de montrer également la différence au niveau du menu entre l'interface enseignant et l'interface responsable.

Répondre à une demande

Un responsable répond à une demande de réservation de séance. Deux types de réponses sont possibles. Valider, ou refuser. Après réponse la demande associée est supprimée et la séance est ajoutée à l'emploi du temps en cas de validation.

Déroulement Il clique sur Répondre aux demandes. Un écran s'affiche récapitulant les demandes encore non traitées dans une liste. Il peut sélectionner une de ces demandes et cliquer sur un des deux boutons disponibles :

- ✓ Voir salles disponibles. Ce bouton affiche un nouvel écran qui liste les salles disponibles, résume la demande sélectionnée et propose deux boutons : Valider qui effectue la réservation dans le planning, supprime la demande, envoie une notification de validation à l'enseignant. Annuler qui ne fait rien sinon revenir à l'écran initial de « répondre à une demande ».
- ✓ Refuser. Ce bouton refuse la demande, envoyant une notification de refus à l'enseignant concerné et supprimant la demande.

Déroulement alternatif

Post-conditions

Accueil

Répondre aux demandes

Demandes

Supprimer une séance

Plannings

Etat de mon service

Deconnexion

Demandes reçues

Demande

Bernard Fallery : Salle TD pour IG 4 Groupe 1 le 17/12/2013 de 8h à 9h30

Michel Sala : Salle Cours avec rétroprojecteur pour IG 4 le 19/12/2013 de 8h à 9h30

Voir salles disponibles

Refuser

Gestion EDT

?

o

x

Accueil

Répondre aux demandes

Demandes

Supprimer une séance

Plannings

Etat de mon service

Deconnexion

Choisissez une salle

Bernard Fallery : Salle TD pour IG 4 Groupe 1 le 17/12/2013 de 8h à 9h30

Sc 101

Sc102

Sc201

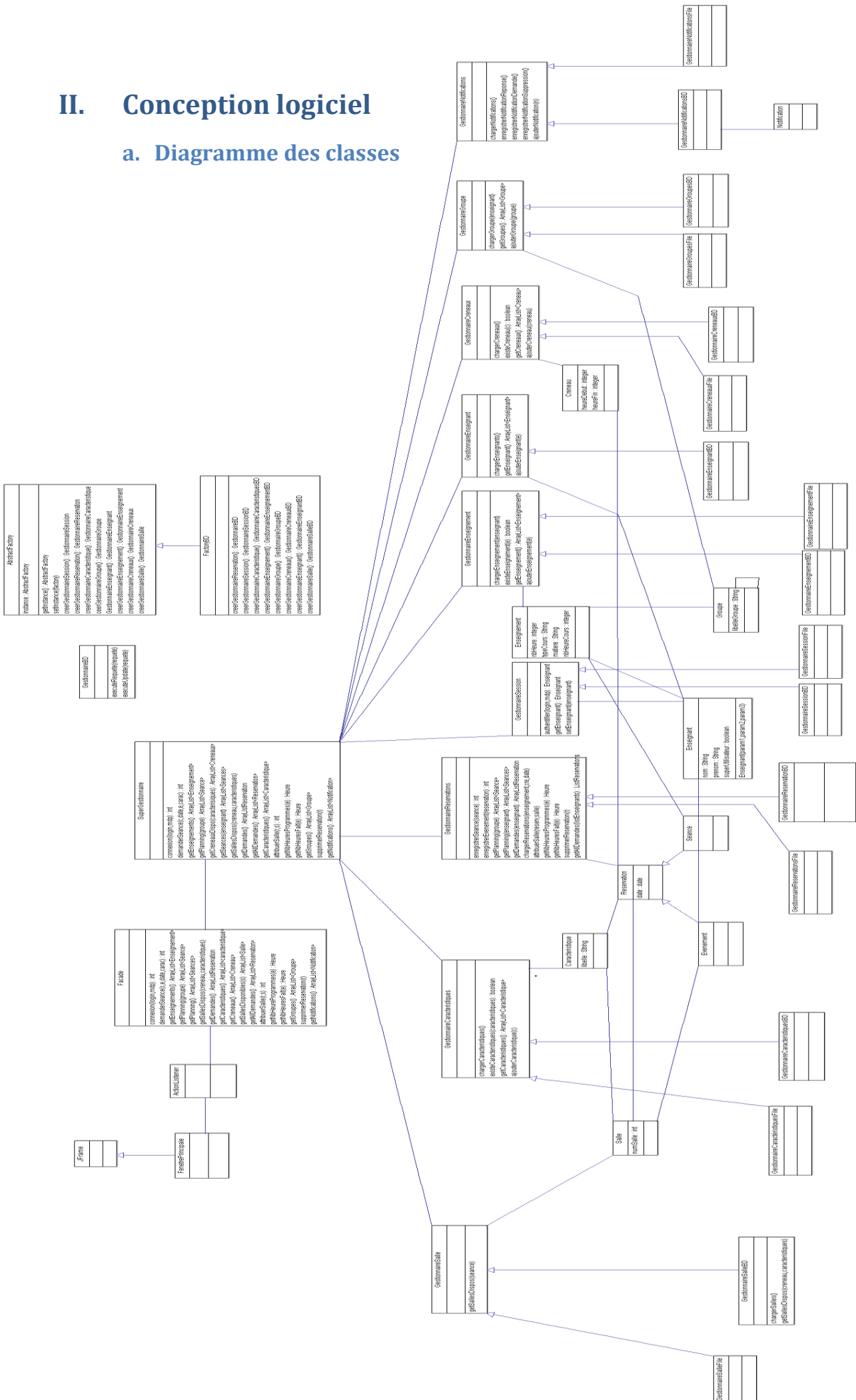
Sc202

Sc114

Valider

Annuler

a. Diagramme des classes



Nous décrirons dans cette partie les classes principales de notre modèle et leur rôle. Nous expliquerons notamment le principe des gestionnaire qui se calque sur le modèle des courtiers. Tous les gestionnaires sont abstraits et devront être implémentés pour un type de données persistantes spécifique, dans notre cas une base de données et des fichiers.

Facade :

Cette classe sert de point d'entrée dans la business logic, elle simplifie donc l'utilisation des différentes fonctionnalités de notre programme à une interface utilisateur.

Nous avons choisi de ne conserver qu'une seule façade pour notre application, en effet après de longues réflexions nous avons estimé que le but premier de cette classe est de simplifier l'utilisation des classes métiers et que le fait de la découper ne ferait que compliquer l'utilisation du programme par une interface utilisateur, en effet cette dernière aura accès à toutes les fonctionnalités via une seule classe.

De plus lors de la conception de nos maquettes nous avons remarqué que chaque fonctionnalité correspondait plus ou moins à un onglet et que toutes les fonctionnalités étaient liées et qu'il y a donc un sens à ce qu'elles soient accessibles au travers de la même interface. Cependant si nous devons développer de nouvelles fonctionnalités ne s'apparentant pas à la gestion de l'emploi du temps (ex : ajout d'une enseignant...) nous utiliserions dans ce cas une façade différente.

SuperGestionnaire :

La classe de SuperGestionnaire sert à superviser le travail de tous les autres gestionnaires, en effet le rôle de la façade étant simplement de fournir un point d'entrée pour accéder aux classes métier nous ne voulions pas que celle-ci ait du traitement à effectuer. Nous avons donc introduit une classe entre la façade et les différents gestionnaire de manière à ce qu'il existe une classe capable de contrôler le travail des gestionnaire et de faire en sorte de contrôler les données venant de l'interface utilisateur (via la façade) de façon à ce que les gestionnaires n'aient pas à se préoccuper de ce problème et de simplement faire les tâches qui leurs sont assignées.

La seconde raison pour laquelle nous avons choisi d'introduire cette classe est par souci de découplage. En effet nous voulions éviter que les gestionnaires se connaissent entre eux ce qui aurait introduit un couplage fort dans notre programme, ainsi seul le super gestionnaire a connaissance des autres gestionnaires et fait en sorte de leur donner les informations dont ils ont besoin.

Par exemple pour obtenir le planning de l'enseignant connecté à l'application, le super gestionnaire va d'abord demander au gestionnaire de session l'instance de l'enseignant connecté avant de la fournir au gestionnaire de réservations pour obtenir l'ensemble de ses séances.

GestionnaireSession :

Cette classe sert à garder une instance en mémoire de l'enseignant connecté à l'application. La présence de cette classe simplifie notamment l'utilisation des différentes fonctionnalités, par exemple pour avoir le planning personnel de l'enseignant connecté l'interface utilisateur n'aura pas à fournir l'enseignant mais le super gestionnaire ira chercher cette information dans cette classe. Sa principale fonction est donc de fournir l'enseignant connecté.

GestionnaireRéservations :

Cette classe sera utilisée pour effectuer des tâches en rapport avec tous les types de réservations comme les séances ou les événements. Elle permettra d'obtenir le planning d'un enseignant (l'ensemble de ses séances), la liste des demandes d'un enseignant mais aussi de supprimer une réservation ou d'attribuer une salle à une réservation.

GestionnaireCaracteristiques :

La classe GestionnaireCaractéristiques fournit des méthodes permettant de gérer les caractéristiques des salles. Son principal rôle est de stocker en mémoire l'ensemble des caractéristiques présentes dans les données persistantes, cela permettra de réduire la communication entre le programme et les données brutes, d'autant plus que les caractéristiques sont des données qui ne changeront pas au cours de l'exécution de l'application ce qui explique leur durée de vie dans notre programme. Le super gestionnaire pourra donc demander ces caractéristiques pour permettre à l'interface utilisateur de les afficher.

GestionnaireSalle :

Cette classe accomplit des tâches propres aux salles, la principale fonctionnalité de cette classe est de fournir la liste des salles disponibles pour une réservation donnée.

GestionnaireEnseignements :

La classe gestionnaire enseignement est chargé de stocker en mémoire les enseignements de l'enseignant connecté. De la même manière que pour les caractéristiques les enseignements sont des données qui ne changeront pas au cours de l'exécution, nous les chargeons donc en mémoire et leur durée de vie est celle de l'application.

GestionnaireEnseignants :

Cette classe fournit la liste des enseignants stockés dans les données persistantes, elle ne sera utilisée que si l'enseignant connecté est un responsable.

GestionnaireCreneaux :

La classe GestionnaireCreneaux conserve la liste des créneaux, cette liste sera chargé lors de la connexion de l'enseignant et conservé tout au long de l'exécution du programme.

GestionnaireGroupes :

Cette classe fournit l'ensemble des groupes liés à l'enseignant connecté qui seront chargés lors de son authentification.

GestionnaireNotifications :

La classe GestionnaireNotification est en charge des tâches se rapportant aux notifications, elle permet notamment d'enregistrer des notifications dans les données persistantes.

GestionnaireBD :

Cette classe fournit une connexion avec une base de donnée et fournit deux méthodes permettant d'effectuer des requêtes. La première sert à exécuter une requête d'interrogation et la seconde une requête de modification.

Pour maintenir une et une seule connexion au cours de l'exécution et être accessible par tous les gestionnaires cette classe suivra le patron de conception singleton.

AbstractFactory :

La classe AbstractFactory représente une fabrique abstraite d'objets et permettra de construire les gestionnaires appropriés selon la nature des données persistantes que l'on veut utiliser.

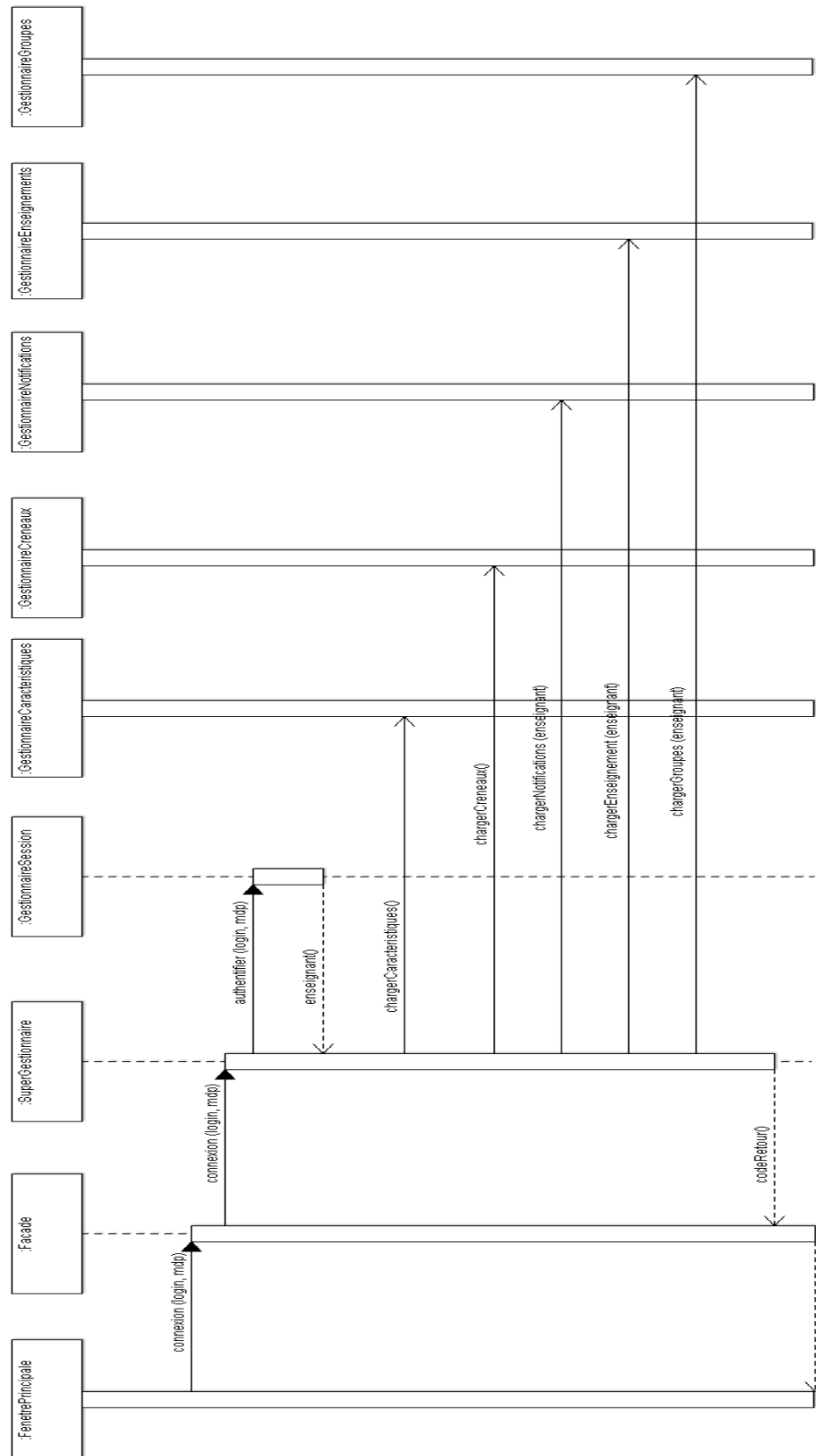
FactoryBD :

Cette classe implémente les méthodes de la classe AbstractFactory et permet de construire des gestionnaires capables d'interagir avec une base de données.

Enfin les classes Créneau, Salle, Enseignant, Enseignement, Notification, Caractéristique, Reservation, Seance, Evenement et Groupe servent à représenter en mémoire les données stockées en mémoire persistante.

b. Diagrammes de séquence

Connexion :

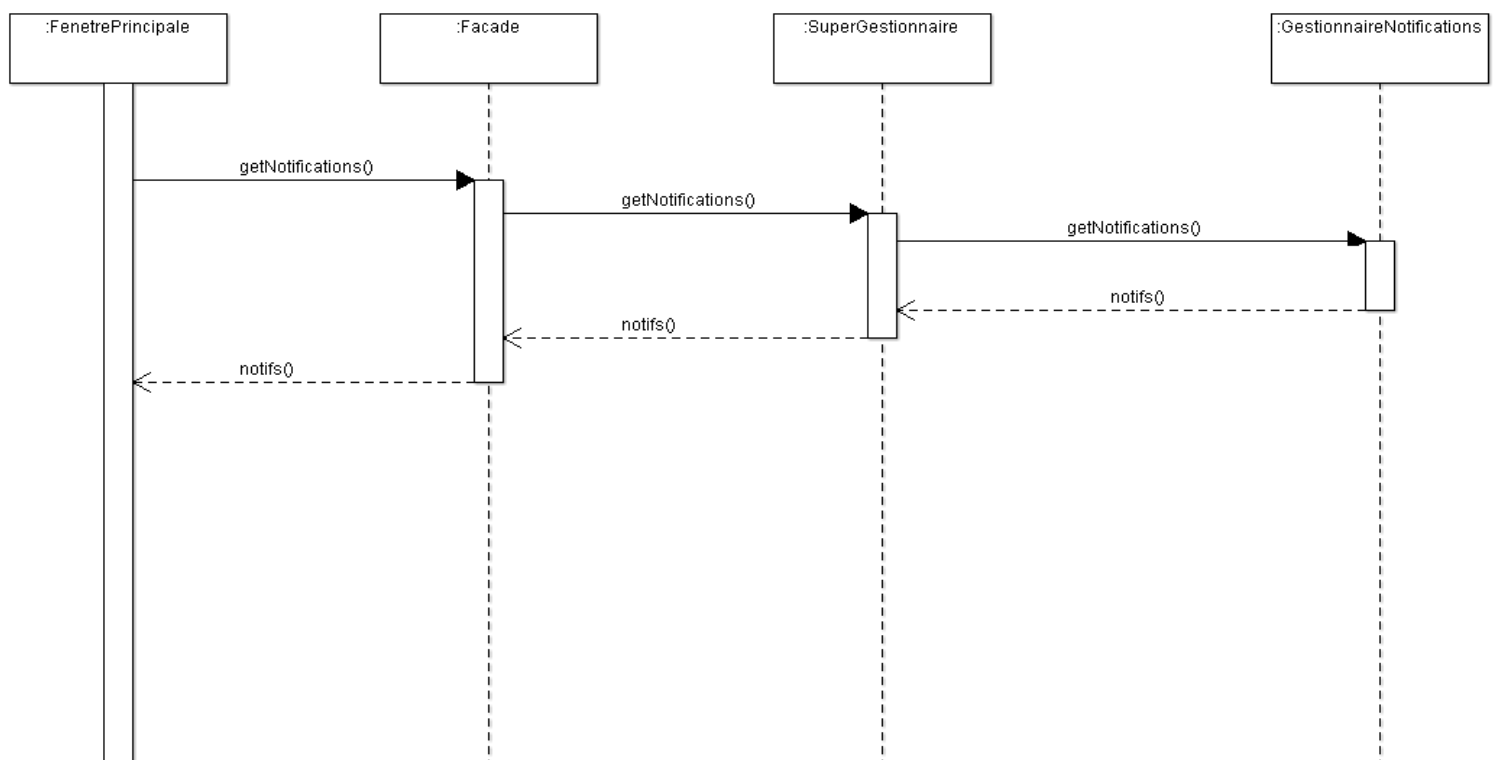


Pour se connecter un utilisateur, la classe FenetrePrincipale appelle chez la façade la méthode « connexion » dont le seul but est de demander au super gestionnaire la connexion de l'utilisateur en lui donnant son identifiant et son mot de passe. Le super gestionnaire prend ensuite le relais et demande 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 changeront 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.

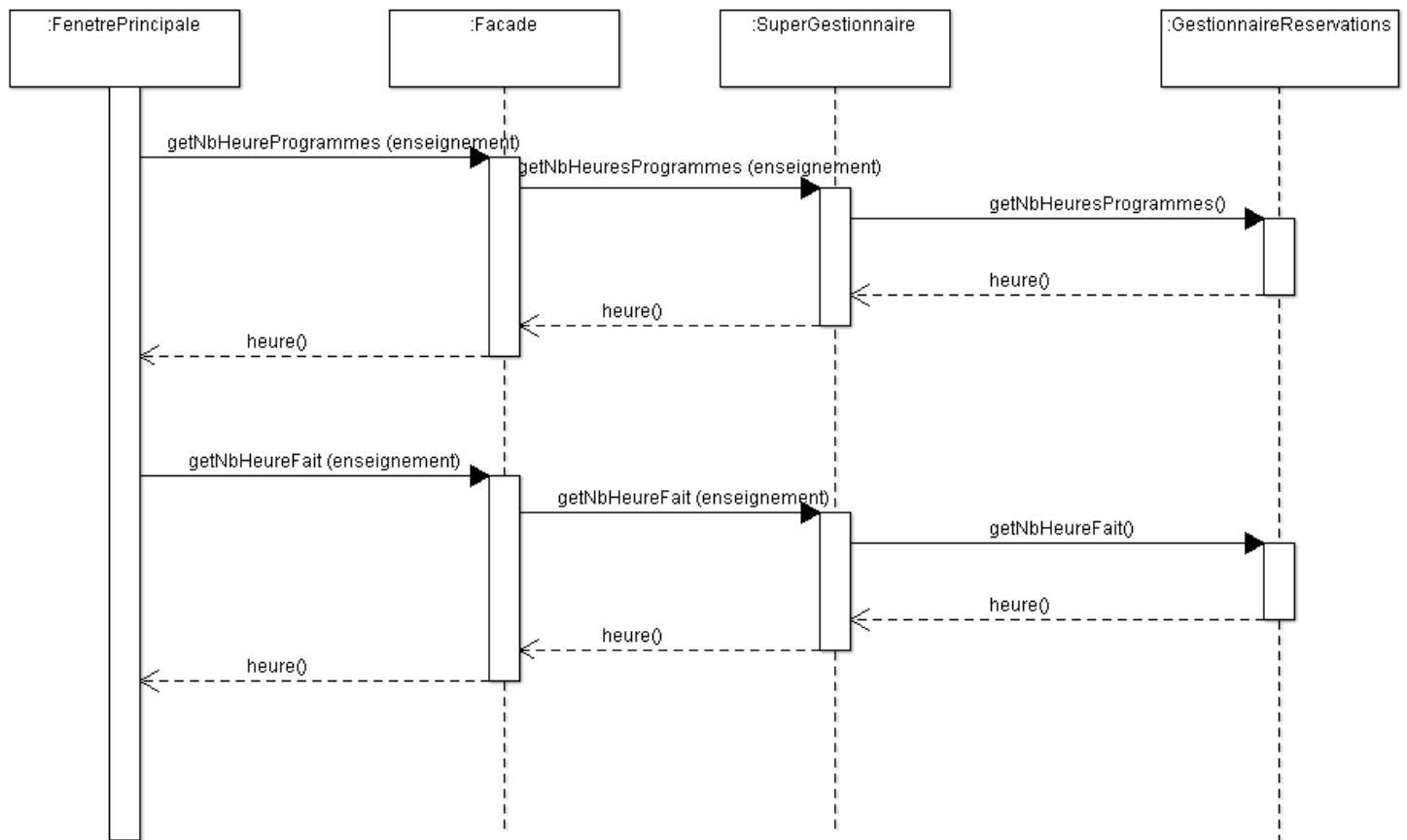
Après cela le super gestionnaire examine l'enseignant retourné par le gestionnaire de session et envoie un code de retour à la façade pour qu'elle puisse communiquer à la fenêtre principale si l'authentification a réussi et si oui, si l'enseignant est une responsable.

Visualisation des notifications :



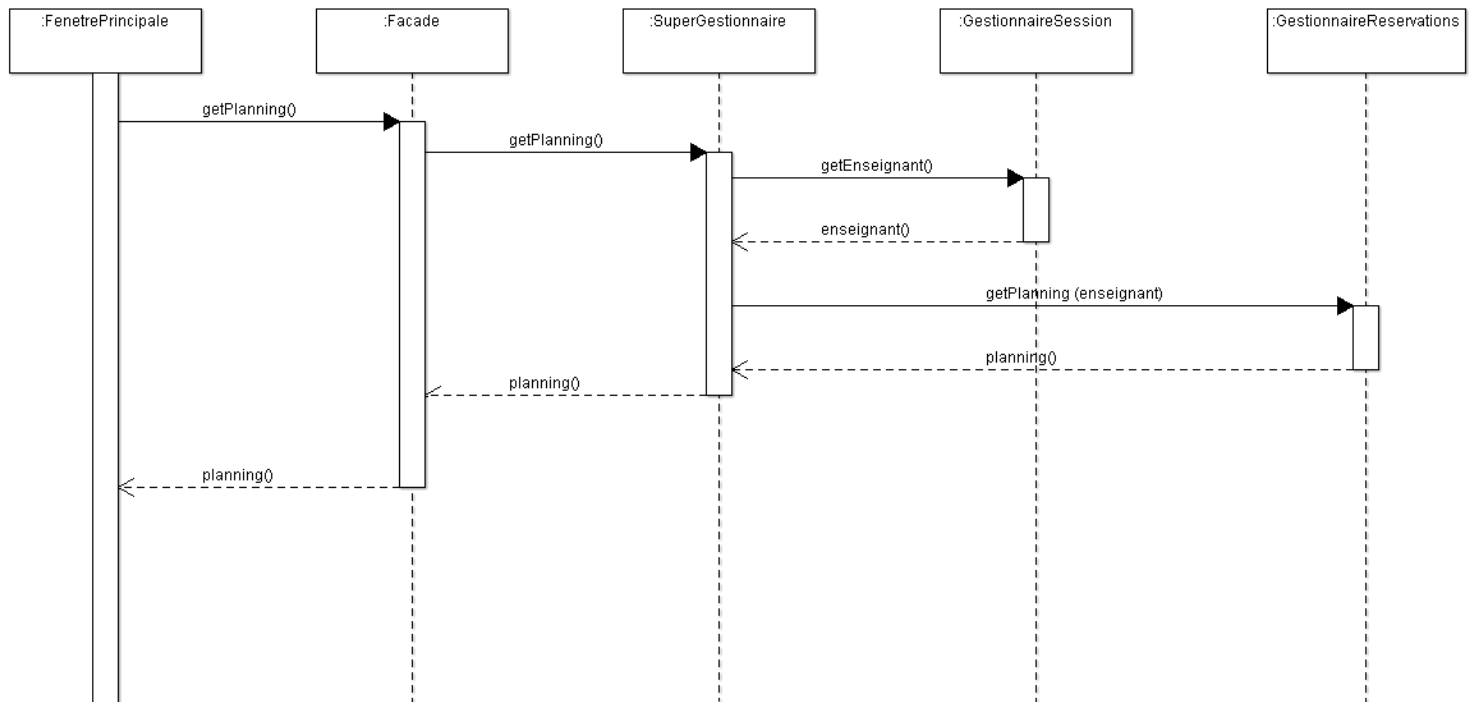
Pour visualiser les notifications, l'interface utilisateur va demander à la façade la liste des notifications de l'enseignant via la méthode getNotifications(). La façade n'étant qu'un point d'entrée pour l'utilisation des classes métiers elle demande simplement au super gestionnaire de lui donner la 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 :



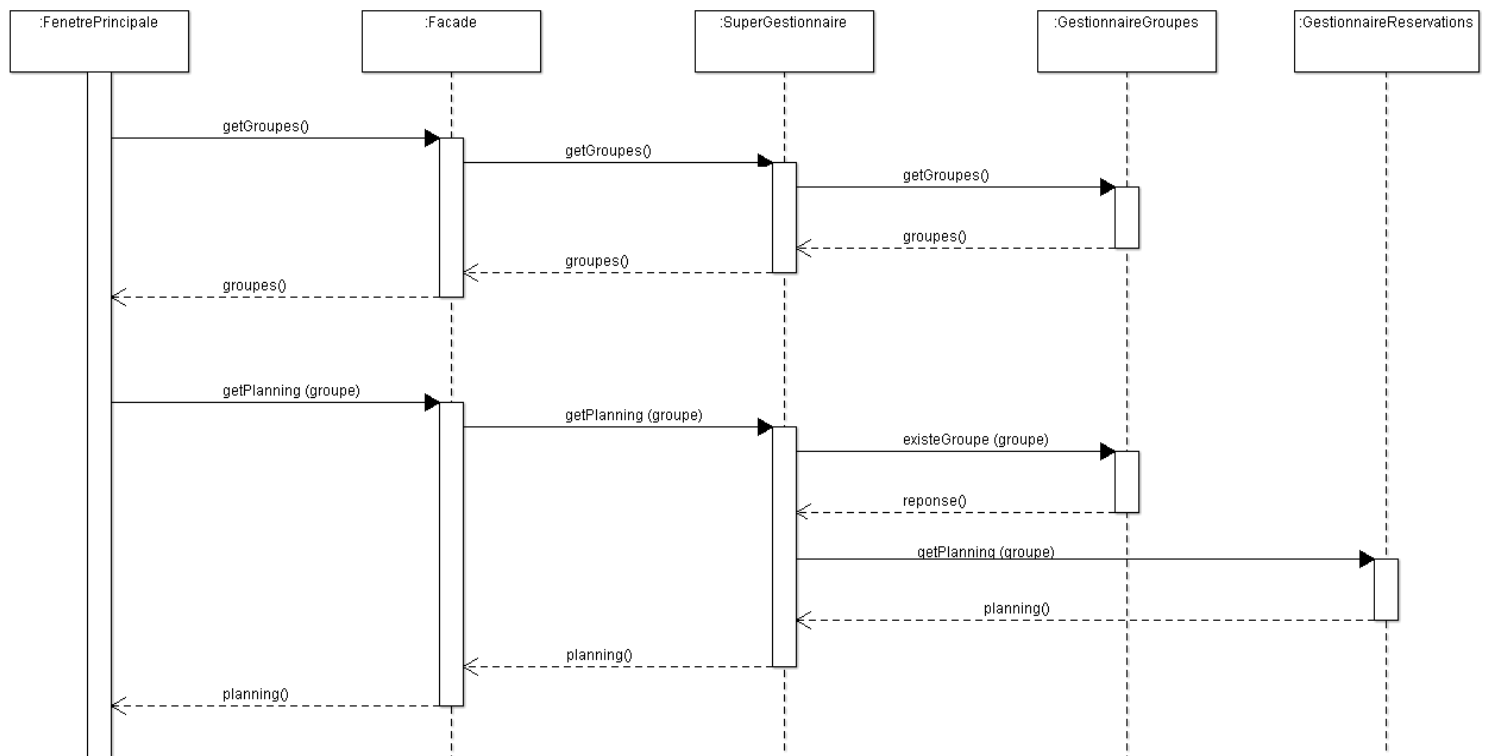
Pour visualiser l'état du service d'un enseignant, la façade fournit les méthodes `getNbHeureProgramme` et `getNbHeureFait` qui donnent respectivement pour un enseignement le nombre d'heure qui est programmé dans le planning et le nombre d'heure déjà effectué. Pour utiliser ces fonctionnalités l'interface 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'enseignements.

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.

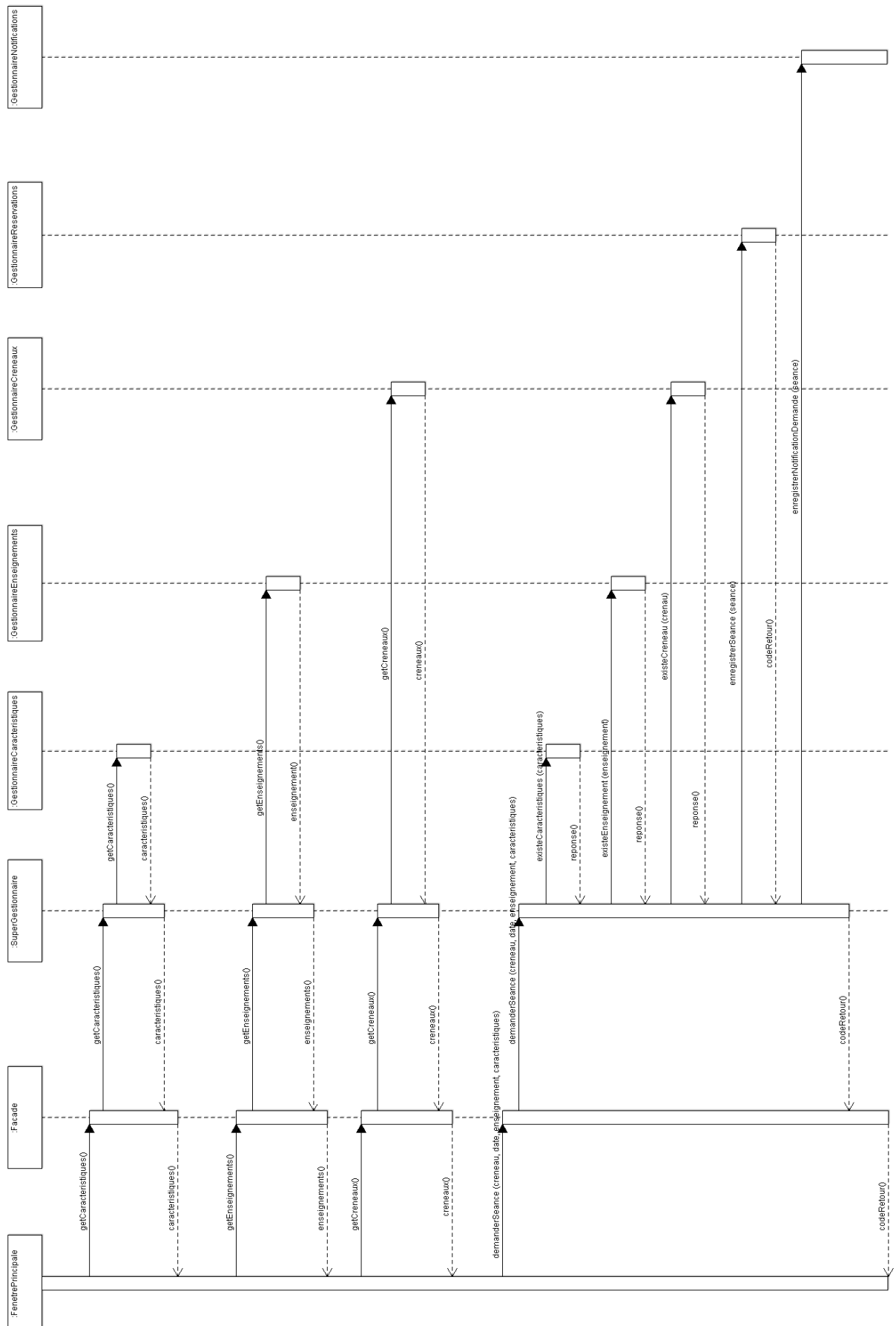
Visualisation du planning d'un groupe :



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.

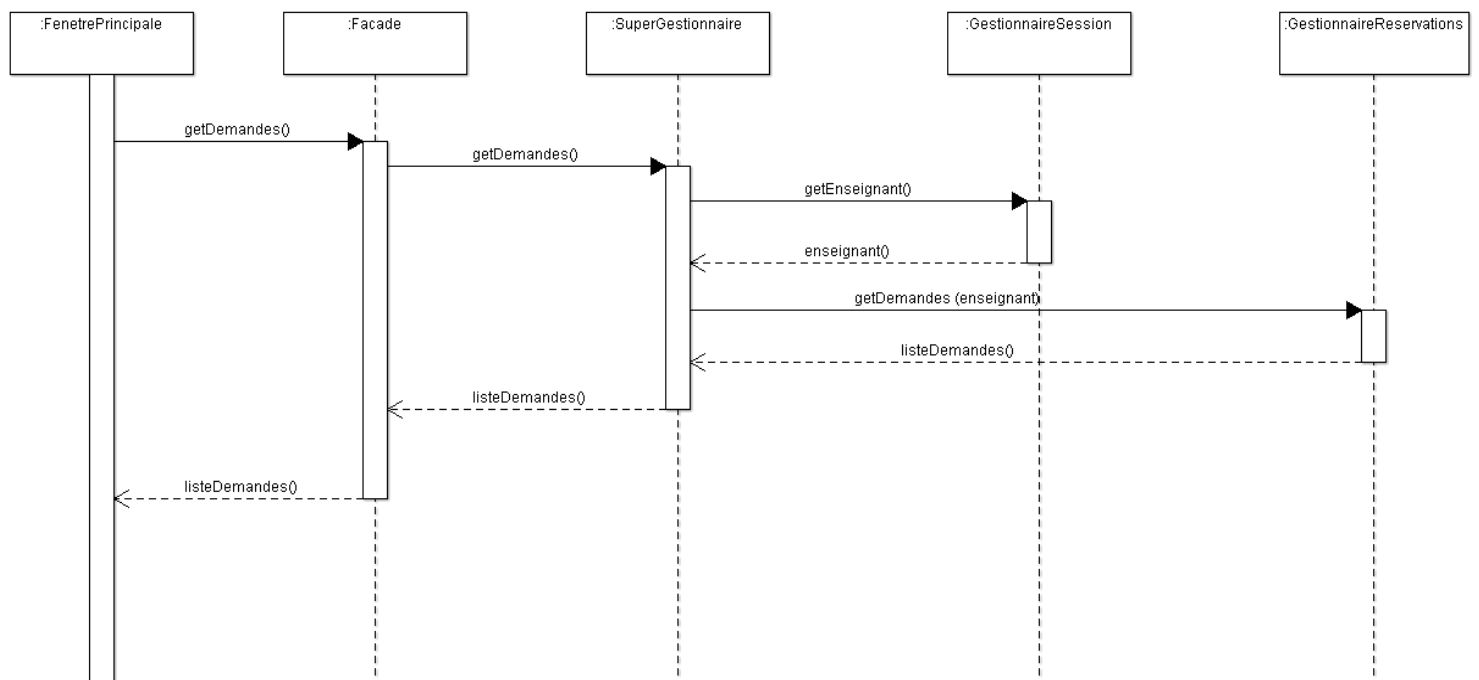
Comme pour la visualisation du planning personnel l'interface utilisateur pourra afficher le planning de ce groupe grâce à la liste des réservations correspondantes renvoyée par la méthode.

Faire une nouvelle demande :



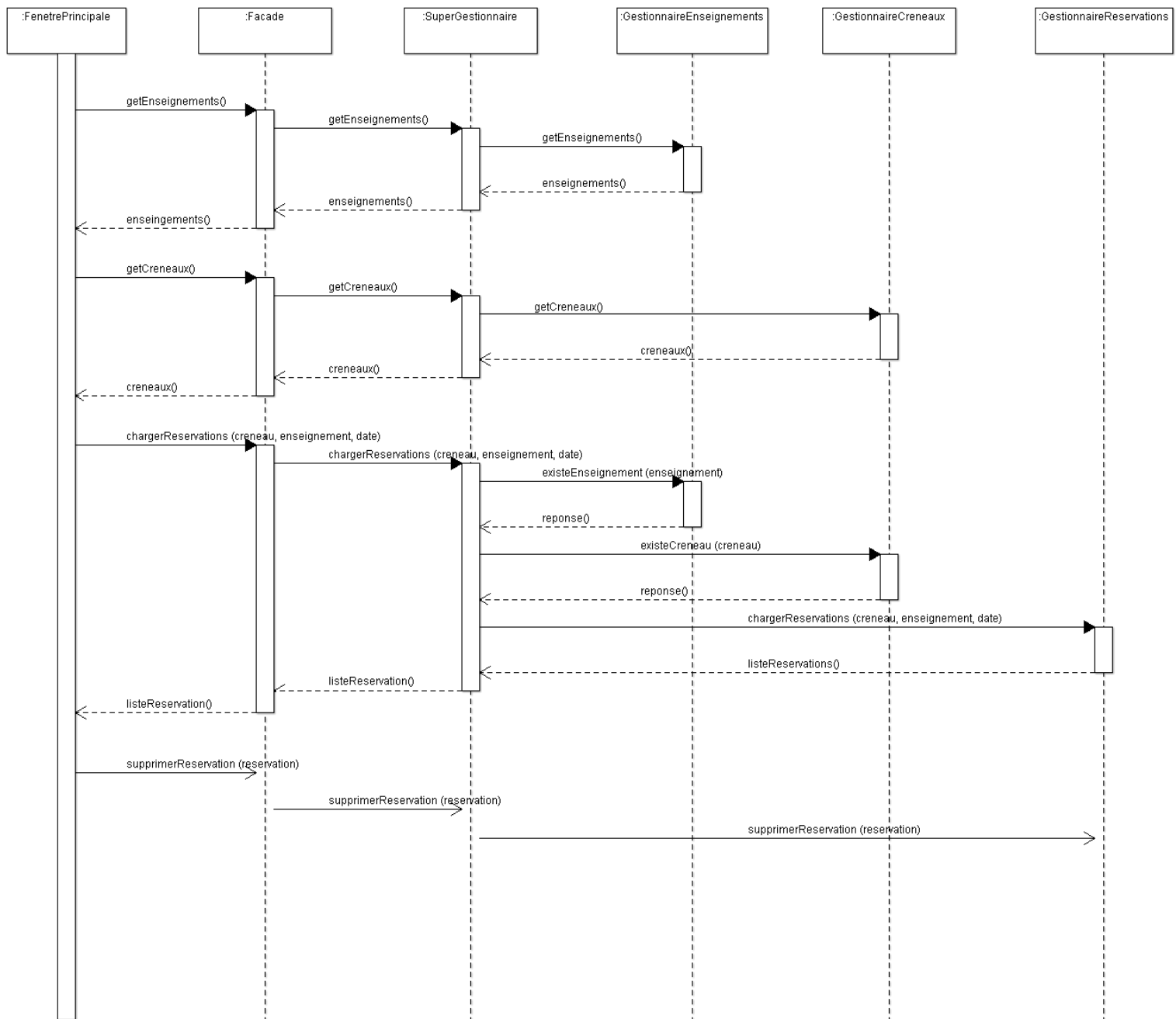
Ce diagramme représente la demande d'une séance par un enseignant. Pour effectuer une demande l'interface utilisateur doit fournir à la business logic une séance, un créneau, une date et une liste de caractéristiques. L'interface peut donc charger depuis les classes métiers les informations disponibles pour l'enseignant connecté. L'interface peut ensuite appeler la méthode demanderSeance avec l'enseignement, le créneau, la date et les caractéristiques souhaitées. La façade délègue le travail au super gestionnaire qui, par mesure de sécurité vérifie que les informations fournis existent bien dans les données persistantes via les différents gestionnaires appropriés et que la date donnée est cohérente, en effet il est incohérent de demander une réservation pour une date antérieure à la date du jour ou pour une dimanche. Si les informations sont correctes le super gestionnaire demande au gestionnaire de réservations d'enregistrer la demande dans les données persistantes. Une fois les données enregistrées le gestionnaire de réservations renvoie un code de retour au super gestionnaire indiquant si l'enregistrement a bien eu lieu et l'information remonte jusqu'à l'interface utilisateur.

Visualisation de ses demandes personnelles :



Un enseignant peut visualiser ses demandes en cours. Pour cela la façade fournit une méthode getDemandes. Cette méthode demande au super gestionnaire les demandes de l'enseignant qui demande donc au gestionnaire de session l'instance de l'enseignant connecté et demande au gestionnaire de réservations de chercher dans les données persistantes les demandes de cet enseignant. La liste des demandes remonte donc ensuite vers l'interface utilisateur qui pourra les afficher à l'utilisateur.

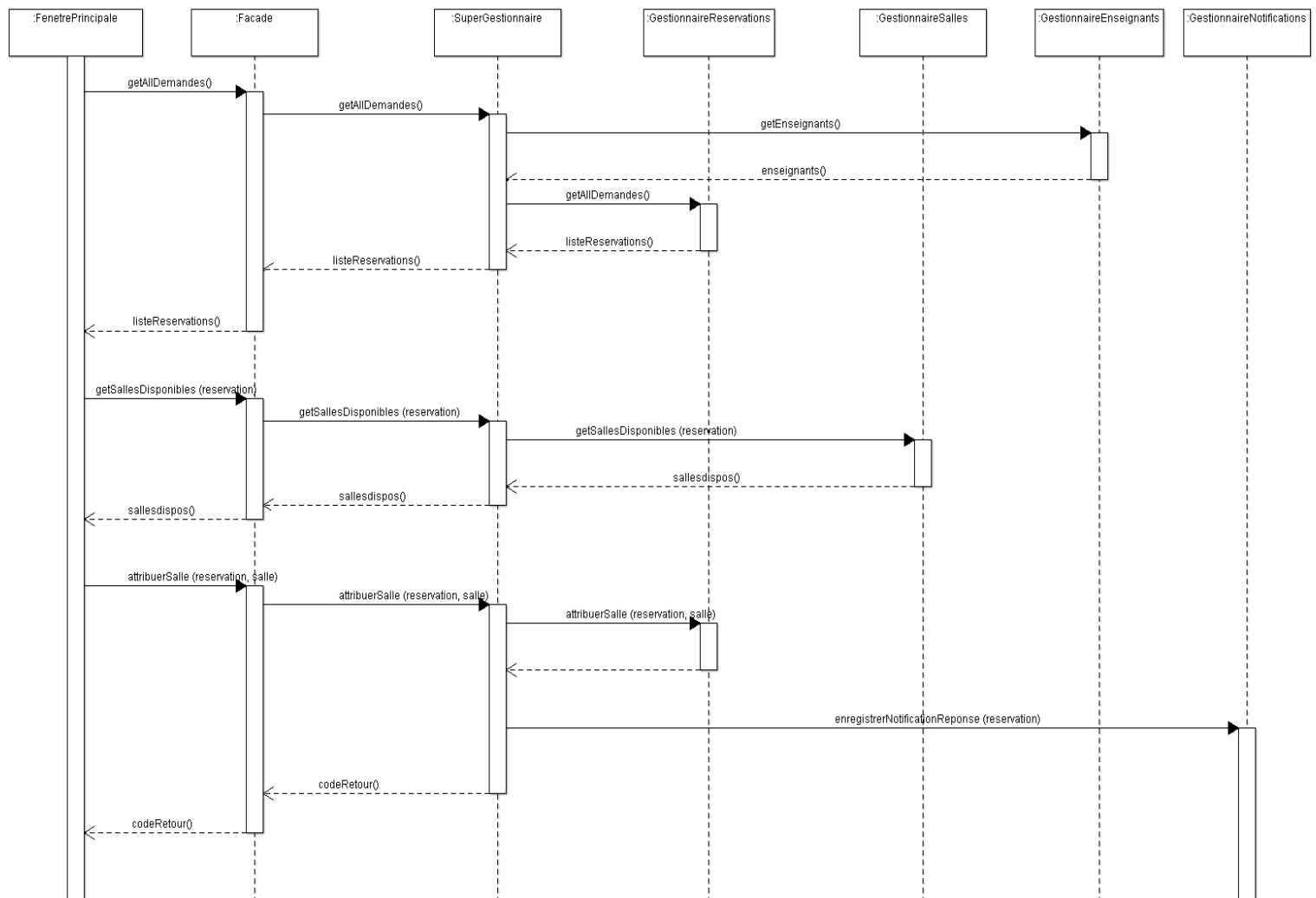
Suppression d'une séance :



Ce diagramme montre le cas d'utilisation de suppression d'une séance par un enseignant. Pour cela la façade fournit la méthode supprimerSeance qui attend une séance en paramètre. La façade fournit aussi une méthode permettant de charger les réservations pour un enseignement, un créneau et une date donnée, ceci facilitera le travail de l'utilisateur qui pourra effectuer une recherche et ne pas à avoir à parcourir la liste de ses réservations pour en supprimer une. La méthode chargerReservations dans la façade demande donc au super gestionnaire de charger les réservations correspondant à un enseignement, une date et un créneau. Le super gestionnaire vérifie que l'enseignement et le créneau existe bien dans les données persistantes et demande au gestionnaire de réservations les séances correspondantes qui remontent jusqu'à l'interface utilisateur.

L'interface peut donc ensuite demander la suppression d'une séance grâce à la méthode supprimer séance.

Attribution d'une salle :



Ce diagramme montre la principale fonctionnalité liée à un responsable à savoir répondre à une demande.

La façade fournit à un responsable la méthode `getAllDemandes` qui donne l'ensemble des demandes en cours. Pour cela la façade demande au super gestionnaire ces demandes et celui-ci récupère l'ensemble des enseignants préalablement chargés dans le gestionnaire d'enseignants. Le super gestionnaire demande ensuite au gestionnaire de réservations l'ensemble des demandes de cette liste d'enseignants.

A partir de ces demandes l'interface pourra faire appel à la méthode `getSallesDisponibles` qui donnera pour une demande donnée la liste des salles qui sont disponibles.

Enfin la façade fournit la méthode `attribuerSalle` qui permet, comme son nom l'indique, d'attribuer une salle à la réservation donnée. Pour cela la façade appelle la méthode `attribuerSalle` chez le super gestionnaire qui demande au gestionnaire de réservation d'enregistrer cette demande dans les données persistantes puis délègue au gestionnaire de notifications la tâche d'enregistrer une notification de la réponse.

c. Persistance des données

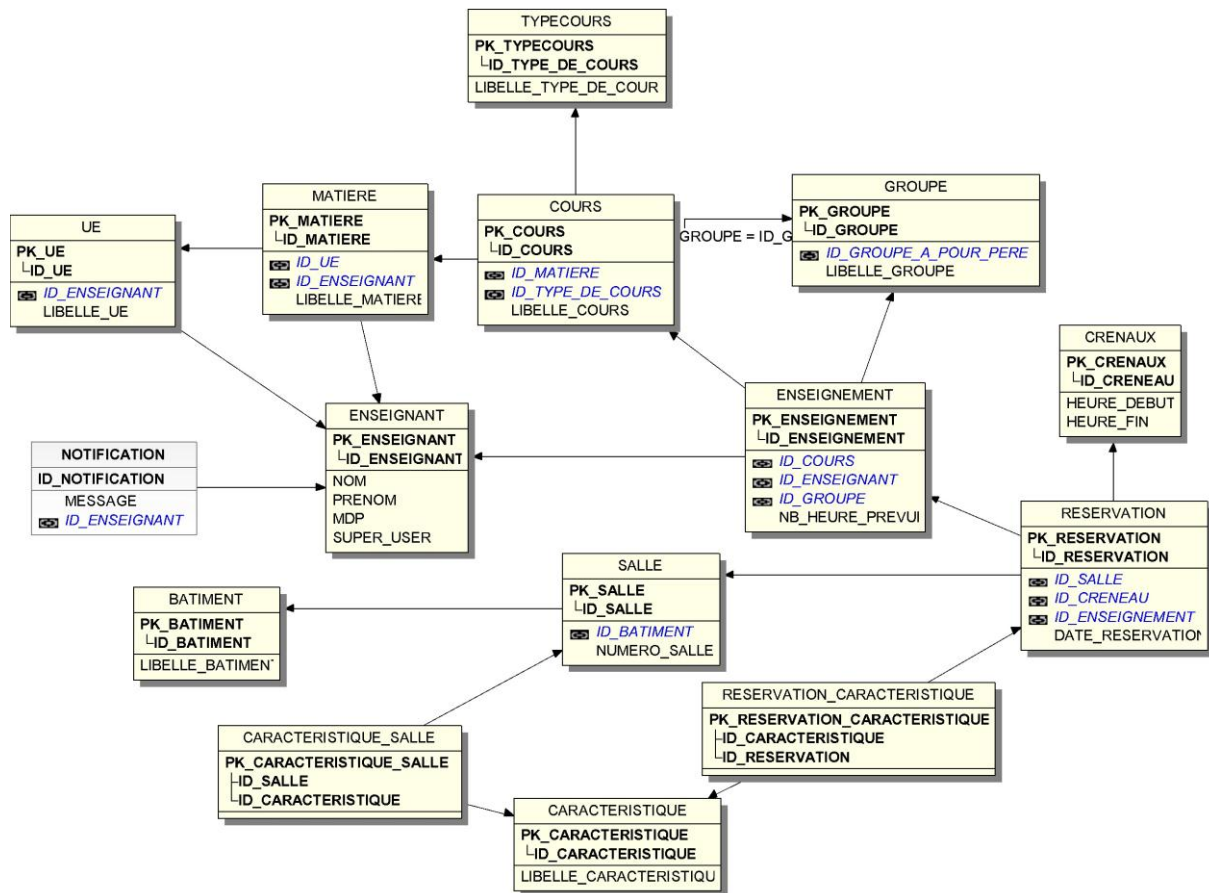
Comme nous l'avons présenté précédemment nous avons tenu au cours de notre conception à ce que notre business logic soit le plus indépendante possible de l'interface utilisateur et de la représentation des données qu'il utilise.

Nous pensons que notre modèle de gestionnaires permet une indépendance totale peu importe le type de persistance des données, en effet les gestionnaires sont tous abstraits et décrivent les fonctionnalités attendus. Pour utiliser un certain type de persistance il suffit donc d'implémenter ces méthodes abstraites de manière à pouvoir utiliser ce type de persistance. De plus la classe `AbstractFactory` permet la création automatique du type de gestionnaire dont l'application a besoin.

Il nous a donc été très simple d'implémenter la fonctionnalité de login avec une persistance sous forme de fichiers. Nous avons simplement créé une nouvelle classe `GestionnaireSessionFile` qui authentifie un enseignant à partir d'un fichier xml.

III. Conception de la Base de données

a. Schéma de la base de données



Le schéma précédent présente le modèle logique de notre base de données. Nous avons décidé d'ajouter la table notification qui contient un message et référence un enseignant. Cette table sert à donc à stocker tout nouvel événement dont doit être informé un enseignant comme la réponse à l'un de ses demandes.

Nous avons dans un premier temps considéré qu'une notification devait avoir une référence vers une réservation cependant nous avons décidé de la remplacer par une référence vers un enseignant car avec notre première idée nous n'aurions pas pu représenter les notifications de suppression de réservation.

b. Vues et types

De manière à utiliser notre base de données plus simplement nous avons décidé de créer des types objets représentant les tables de la base de données ainsi que des vues sur ces types.

La vue principalement utilisée dans notre projet est la vue VEnseignant qui représente la table Enseignant sous forme de vue objet. Cette vue nous permet d'avoir simplement pour un enseignant la liste de ses enseignements et de ses réservations. Celle-ci nous a grandement facilité un certain nombre de requêtes notamment pour connaître le planning d'un enseignant. Nous avons donc ainsi évité d'effectuer des jointures compliqués.

Nous avons créé une vue VSalle ainsi qu'une vue réservation qui nous permettent d'avoir pour une salle et pour une réservation la liste de leurs caractéristiques.

De plus l'application étant développée en java et donc dans un langage objet, l'utilisation des vues était cohérente et simplifiée le lien entre l'application et la base de données.

Ces types sont disponibles dans le fichier types.sql. (cf annexe).

Représentation de la vue Enseignant :

ID_enseignant	Nom	Prenom	Enseignement 1	Réservation 1
			Enseignement 2	Réservation 2
			Enseignement 3	Réservation 3
			Enseignement 4	Réservation 4

Représentation de la vue VReservation :

ID_Reservation	Salle	Créneau	Enseignement	Date	Caractéristique1
					Caractéristique2
					Caractéristique3

Représentation de la vue VSalle :

ID_Salle	Numero_Salle	Caractéristique1
		Caractéristique2
		Caractéristique3

IV. Conclusion

Notre conception répond aux besoins immédiats mais laisse également la place à d'envisageables évolutions. Nous avons une bonne transparence au niveau persistance des données ainsi qu'un couplage faible entre la couche graphique et la couche logique.

V. Annexes

Types.sql :

```
drop type ty_Enseignant;
```

```
drop type nt_reservation;  
drop type ty_ref_reservation;  
drop type ty_Reservation ;
```

```
drop type ty_Salle ;
```

```
drop type ty_creneau ;
```

```
drop type nt_caracteristique ;  
drop type ty_ref_caracteristique ;  
drop type ty_caracteristique ;
```

```
drop type nt_enseignement ;  
drop type ty_ref_ENseignement;  
drop type ty_Enseignement ;
```

```
drop type ty_cours;
```

```
drop type ty_type_cours ;
```

```
drop type ty_matiere;
```

```
drop type ty_groupe;
```

```
drop view VEnseignant;
```

```
drop view VReservation;
```

```
drop view VCreneau;
```

```
drop view VGroupe;
```

```
drop view VCours;
```

```
drop view VType_cours;
```

```
drop view VMatiere;
```

```
drop view VSalle;  
drop view VCaracteristique;
```

```
drop view VEnseignement;
```

-----TYPES OBJETS-----

```
create or replace type ty_caracteristique as object (ID_Caracteristique number (10),  
libelle_caracteristique char (255))
```

```
/
```

```
create or replace type ty_ref_caracteristique as object (RefCa ref ty_caracteristique)
```

```
/
```

```
create or replace type nt_caracteristique as table of ty_ref_caracteristique
```

```
/
```

```
create or replace type ty_salle as object (ID_SALLE NUMBER(10), numero_salle CHAR(255),  
carac nt_caracteristique)
```

```
/
```

```
create or replace type ty_groupe as object (ID_GROUPE number(10), libelle_groupe char(255),  
ID_GROUPE_PERE number(10))
```

```
/
```

```
create or replace type ty_creneau as object (ID_CRENEAU NUMBER(10),heure_debut char(255),  
heure_fin CHAR(255))
```

```
/
```

```
create or replace type ty_matiere as object (ID_matiere NUMBER(10), libelle_matiere char(255))
```

```
/
```

```
create or replace type ty_type_cours as object (ID_type_cours NUMBER(10), libelle_type_cours  
char (255))
```

```
/
```

```
create or replace type ty_cours as object(ID_COURS number(10), matiere ref ty_matiere,  
type_cours ref ty_type_cours)
```

```
/
```

```
create or replace type ty_enseignement as object(ID_ENSEIGNEMENT NUMBER(10), cours ref  
ty_cours, groupe ref ty_groupe, nb_heure_prevue NUMBER(4))
```

```
/
```

```
create or replace type ty_ref_enseignement as object (RefE ref ty_enseignement)
```

```
/
```

```
create or replace type nt_enseignement as table of ty_ref_enseignement
```

```
/
```

```
create or replace type ty_Reservation as object (ID_RESERVATION NUMBER(10), salle ref  
ty_salle, creneau ref ty_creneau,  
enseignement ref ty_enseignement, dateRes date, caracteristiques nt_caracteristique)
```

```
/
```

```
create or replace type ty_ref_reservation as object (RefR ref ty_reservation)
```

```
/
```

```
create or replace type nt_reservation as table of ty_ref_reservation
```

```
/
```

```

create or replace type ty_enseignant as object (ID_ENSEIGNANT NUMBER(10), nom CHAR(255),
prenom CHAR(255),
liste_Enseignement nt_enseignement, liste_Reservations nt_reservation)
/

```

-----VUES-----

```

create or replace view VCreneau of ty_creneau
with object identifier(ID_CRENEAU)
as select C.ID_CRENEAU,C.HEURE_DEBUT, C.HEURE_FIN from crenaux C;

```

```

create or replace view VGroupe of ty_groupe
with object identifier(ID_GROUPE)
as select G.ID_GROUPE, G.libelle_groupe, G.ID_GROUPE_A_POUR_PERE from groupe G;

```

```

create or replace view VMatiere of ty_matiere
with object identifier(ID_MATIERE)
as select M.ID_MATIERE, M.LIBELLE_MATIERE from matiere M;

```

```

create or replace view VType_cours of ty_type_cours
with object identifier(id_type_cours)
as select T.id_type_de_cours, T.libelle_type_de_cours from typecours T;

```

```

create or replace view VCours of ty_cours
with object identifier(id_cours)
as select C.id_cours, make_ref(VMatiere, C.id_matiere), make_ref(VType_cours,
C.id_type_de_cours) from cours C;

```

```

create or replace view VCaracteristique of ty_caracteristique
with object identifier(ID_CHARACTERISTIQUE)
as select C.ID_CHARACTERISTIQUE, C.LIBELLE_CHARACTERISTIQUE from caracteristique C;

```

```

create or replace view VSalle of ty_salle
with object identifier(ID_SALLE)
as select S.ID_SALLE, S.NUMERO_SALLE,
cast(
    multiset(
        select make_ref(VCaracteristique, C.ID_CHARACTERISTIQUE) from
caracteristique_salle C where S.ID_SALLE = C.ID_SALLE) as nt_caracteristique) from salle S;

```

```

create or replace view VEnseignement of ty_enseignement
with object identifier(id_enseignement)
as select E.id_enseignement, make_ref(VCours, E.id_cours), make_ref(VGroupe, E.id_groupe),
E.nb_heure_prevue from enseignement E;

```

```

create or replace view VReservation of ty_reservation
with object identifier(id_reservation)
as select R.id_reservation, make_ref(VSalle, R.id_salle), make_ref(VCreneau, R.id_creneau),
make_ref(VEnseignement, R.id_enseignement), R.date_reservation,
cast (
    multiset(
        select make_ref(VCaracteristique, RC.ID_CHARACTERISTIQUE) from
reservation_caracteristique RC where RC.ID_RESERVATION = R.ID_RESERVATION) as
nt_caracteristique)

```

```
from reservation R;
```

```
create or replace view VEnseignant of ty_enseignant
with object identifier(ID_ENSEIGNANT)
as select ens.ID_ENSEIGNANT, ens.NOM, ens.PRENOM,
cast(
    multiset(
        select make_ref(VEnseignement, E.id_enseignement) from enseignement E where
ens.id_enseignant = E.id_enseignement) as nt_enseignement),
cast(
    multiset(
        select make_ref(VReservation, R.id_reservation) from reservation R where
ens.id_enseignant = (select Ebis.id_enseignant from enseignement Ebis where
R.id_enseignement = Ebis.id_enseignement)) as nt_reservation) from enseignant ens;
```

```
-----REQUETES-----
```

```
-- Enseignements d'un professeur donné
```

```
select Ens.RefE.nb_heure_prevue, deref(deref(Ens.RefE.cours).type_cours).libelle_type_cours,
deref(deref(Ens.RefE.cours).matiere).libelle_matiere, deref(Ens.RefE.groupe).libelle_groupe
from table( select E.liste_enseignement from VEnseignant E where nom = 'Stratulat') Ens;
```

```
-- Demandes en cours d'un enseignant
```

```
select Res.RefR.id_reservation, deref(Res.RefR.creneau).heure_debut,
deref(Res.RefR.creneau).heure_fin, deref(Res.RefR.enseignement).nb_heure_prevue,
deref(deref(deref(Res.RefR.enseignement).cours).matiere).libelle_matiere,
deref(deref(deref(Res.RefR.enseignement).cours).type_cours).libelle_type_cours,
deref(deref(Res.RefR.enseignement).groupe).libelle_groupe,
Res.RefR.dateRes from table(select E.liste_Reservations from VEnseignant E where nom =
'Stratulat') Res where Res.RefR.salle is null;
```

```
select Res.RefR.id_reservation, deref(Res.RefR.creneau).heure_debut,
deref(Res.RefR.creneau).heure_fin, deref(Res.RefR.enseignement).nb_heure_prevue,
deref(deref(deref(Res.RefR.enseignement).cours).matiere).libelle_matiere, deref(deref(deref(Re
s.RefR.enseignement).cours).type_cours).libelle_type_cours,
deref(deref(Res.RefR.enseignement).groupe).libelle_groupe, Res.RefR.dateRes from table(select
E.liste_Reservations from VEnseignant E where nom = 'Stratulat') Res where Res.RefR.salle is
null;
```

```
-- savoir si un groupe est disponible pour un creneau donné
-- vérifier si groupe pere disponible
```

```
select id_reservation from reservation r where r.id_creneau = (select id_creneau from crenaux
where heure_debut = '8h00' and heure_fin = '9h45') and (select id_groupe from enseignement
where id_enseignement = r.id_enseignement) = (select id_groupe from groupe where
libelle_groupe = 'IG4') and id_salle is not null;
```

```
-- Salles qui ont toutes les caracteristiques souhaitées
select id_salle from caracteristique_salle cars where (select count(*) from
(select id_caracteristique from caracteristique c where c.libelle_caracteristique =
'veideoprojecteur' or c.libelle_caracteristique = 'grande'
```



```
MINUS
select id_caracteristique from caracteristique_salle cs where cs.id_salle = cars.id_salle)) = 0;
```

```
--select s.id_salle from salle s where (select count(*) from
--(
--(select id_caracteristique from caracteristique c where c.libelle_caracteristique =
'videoprojecteur' or c.libelle_caracteristique = 'grande')
-- MINUS
--(select id_caracteristique from caracteristique_salle cs where --cs.id_salle = s.id_salle)
--)) = 0;
```

```
Select s.ID_SALLE
from salle s, reservation r
where r.ID_RESERVATION = 5 and
s.ID_SALLE in (select ID_SALLE from salle sa where
(select count(ID_CHARACTERISTIQUE) from caracteristique where
ID_CHARACTERISTIQUE in ((
SELECT ID_CHARACTERISTIQUE from reservation_caracteristique where ID_RESERVATION =
r.ID_RESERVATION) MINUS(
SELECT ID_CHARACTERISTIQUE from CHARACTERISTIQUE_SALLE where ID_SALLE =
sa.ID_SALLE))) = 0)
and s.ID_SALLE not in (select sa.ID_SALLE from salle sa, reservation re
WHERE sa.ID_SALLE = re.ID_SALLE and re.DATE_RESERVATION = r.DATE_RESERVATION
and re.ID_CRENEAU = r.ID_CRENEAU);
```

```
-- groupes d'un enseignant
```

```
select distinct(libelle_groupe) from groupe g, enseignement e where e.id_groupe = g.id_groupe
and e.id_enseignant = 1;
```

```
-- toutes les demandes
```

```
select Res.RefR.id_reservation, deref(Res.RefR.creneau).heure_debut,
deref(Res.RefR.creneau).heure_fin, deref(Res.RefR.enseignement).nb_heure_prevue,
deref(deref(deref(Res.RefR.enseignement).cours).matiere).libelle_matiere,deref(deref(deref(Re
s.RefR.enseignement).cours).type_cours).libelle_type_cours,
deref(deref(Res.RefR.enseignement).groupe).libelle_groupe,Res.RefR.dateRes from table(select
E.liste_Reservations from VEnseignant E ) Res where Res.RefR.salle is NULL;
```