

**Camille Raymond – Christophe Gire
Damien Sendner – Thibaut Rouquette**

Diagrammes de séquence du projet URM

University Resources Management

Résumé des diagrammes de séquence du logiciel de gestion de réservation des salles et de la consultation de planning : URM.

05/03/2012

Contenu

Diagramme de séquence – Login et Menu.....	2
Diagramme de séquence – Consultation du planning	3
Diagramme de séquence – Demande de réservation	4
Diagramme de séquence – Traitement des demandes de réservation	6

Diagramme de séquence – Login et Menu

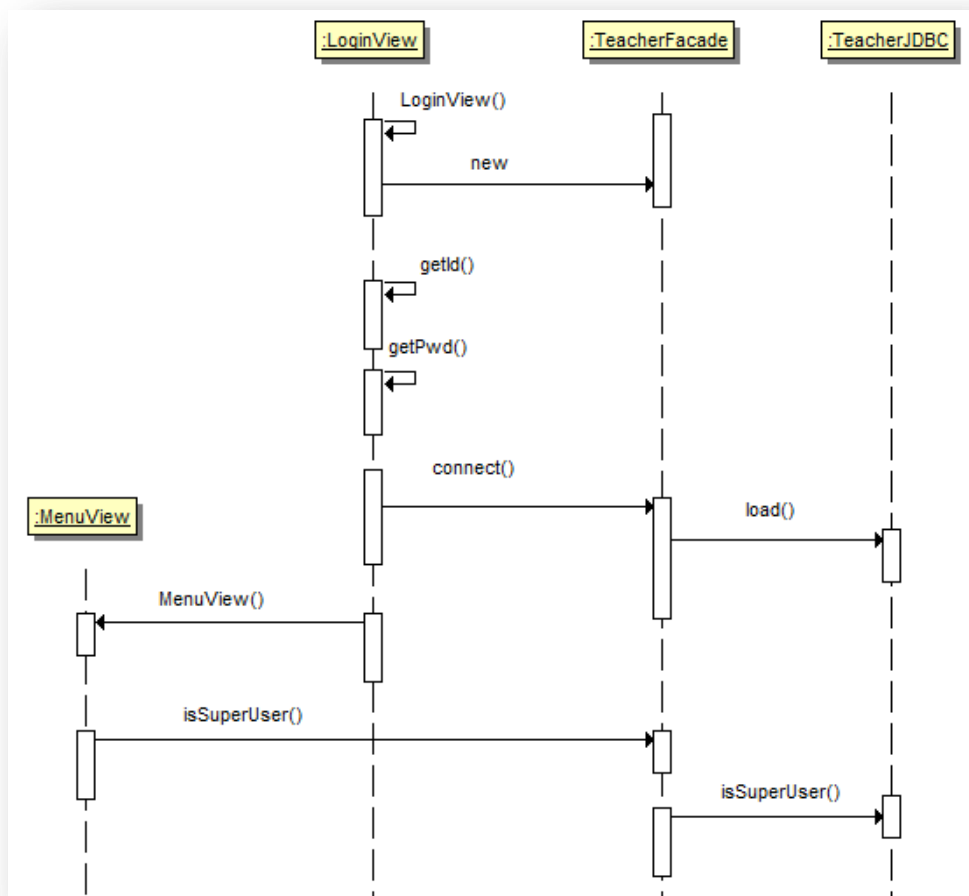


Figure 1 - Diagramme de séquence : Login et Menu

Commentaires

Lors de la création de l'objet **LoginView** représentant la fenêtre graphique pour l'identification d'un enseignant, celui-ci crée l'objet **TeacherFacade** qui a pour but de manipuler les objets métiers de manière à répondre aux demandes d'informations de la couche d'interface utilisateur.

Pour s'identifier, l'utilisateur appuie sur le bouton connecter du LoginView après avoir rentré son identifiant et son mot de passe. Le système récupère ces informations et la façade va charger l'enseignant correspondant à l'utilisateur authentifié.

Ainsi, le LoginView initialise le MenuView qui affiche le menu correspondant à l'utilisateur. Ce menu va être différent si l'utilisateur est responsable ou non.

Scénarios alternatifs

- cancel() : l'utilisateur appuie sur le bouton 'Cancel' de la fenêtre login : le cas d'utilisation se finit.
- quit() : l'utilisateur appuie sur le bouton 'Quit' du menu : le cas d'utilisation se finit.
- Le champ *username* ou le champ *password* n'est pas rempli : un message d'erreur est alors affiché et le cas d'utilisation recommence.
- La combinaison *username* / *password* est invalide : un message d'erreur est alors affiché et le cas d'utilisation redémarre.

Diagramme de séquence – Consultation du planning

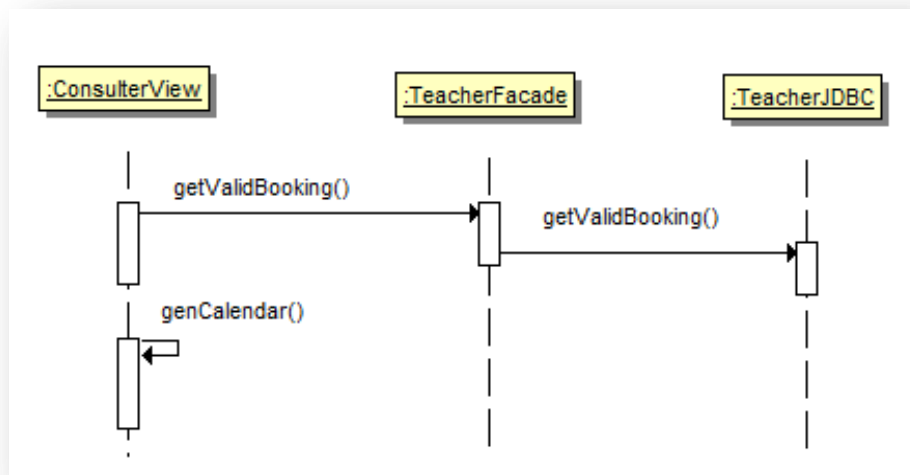


Figure 2 - Diagramme de séquence : Consultation

Commentaires

La fenêtre **ConsulterView** représente la fenêtre graphique pour la consultation des enseignements d'un enseignant.

ConsulterView demande à la façade **TeacherFacade** toutes les réservations valides de l'enseignant.

La façade, qui ne fait que déléguer : elle demande à **TeacherJDBC** de lui fournir les réservations valides de l'enseignant.

Une fois les informations récupérées, ConsulterView va pouvoir ainsi générer le calendrier en affichant le planning de l'enseignant avec ses enseignements correspondant aux demandes de réservations qui ont été validées par le responsable.

Au chargement, le système va chercher dans la base les données de l'emploi du temps de la semaine sélectionnée pour les afficher.

Lors d'un changement de semaine, l'affichage est mis à jour et les données rechargées.

Scénarios alternatifs :

- L'utilisateur appuie sur la flèche '>' du calendrier : on passe à la semaine suivante et le cas d'utilisation continue.
- L'utilisateur appuie sur la flèche '<' du calendrier : on passe à la semaine précédente et le cas d'utilisation continue.
- L'utilisateur appuie sur le bouton 'Quit' de la fenêtre: le cas d'utilisation se termine.

Diagramme de séquence – Demande de réservation

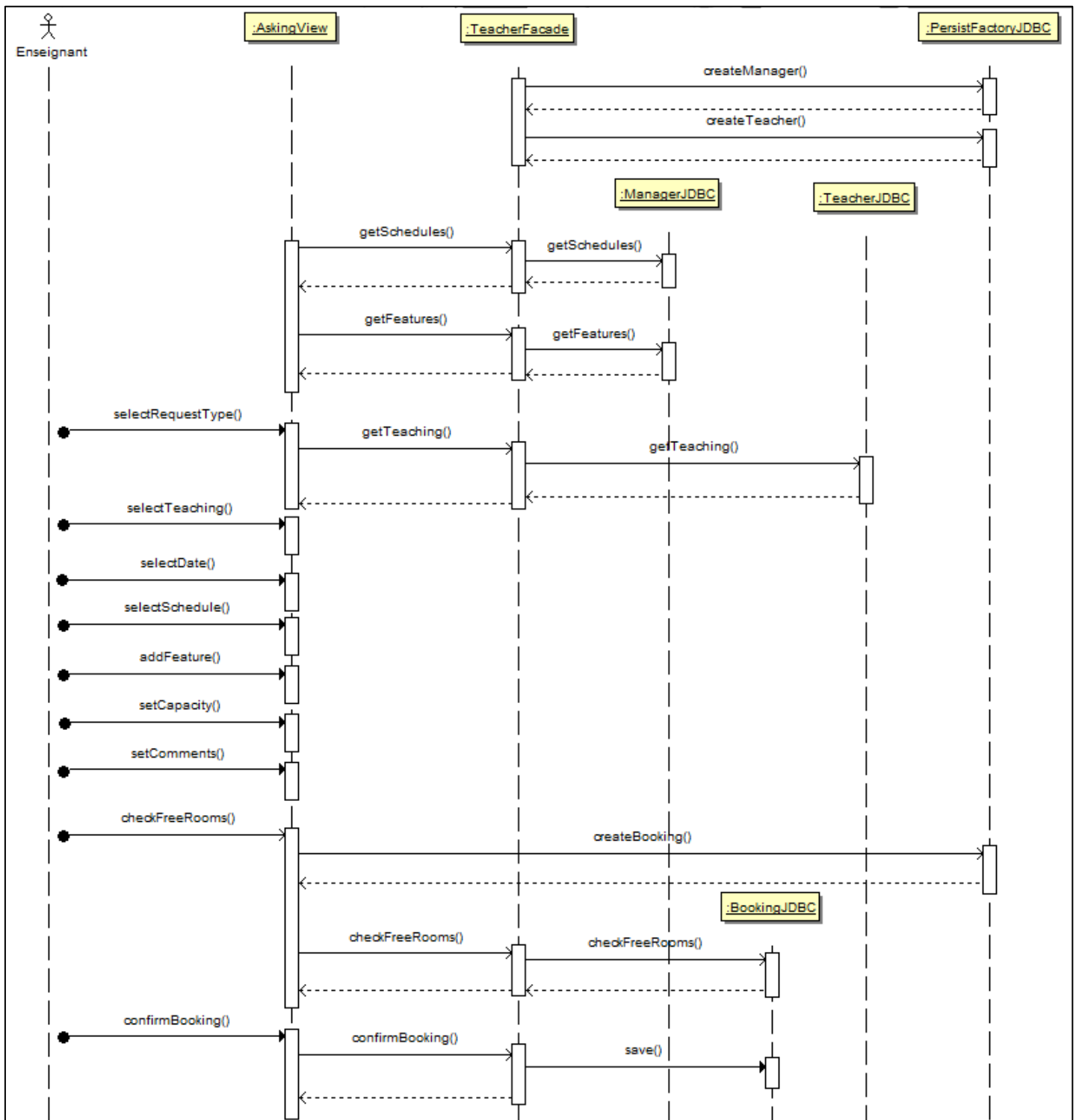


Figure 3 - Diagramme de séquence : demande de réservation

Préconditions

L'utilisateur doit être logué.

Commentaires

Lors de la création de l'objet **AskingView** représentant la fenêtre graphique pour la demande de réservation de salle, l'objet **TeacherFacade** crée le gestionnaire qui a pour but de répartir les **ressources** et l'**enseignant** qui correspond à l'utilisateur authentifié.

Lors de l'initialisation de la fenêtre **AskingView**, cette dernière demande à la **façade** les **caractéristiques** et les **créneaux horaires** existants. La façade, qui ne fait que déléguer, demande à **ManagerJDBC** de lui fournir les **créneaux** et les **caractéristiques** stockés dans la base de données.

Dans ce cas, l'utilisateur fait le choix de demander une réservation pour un enseignement. **AskingView** va donc faire une demande à la **façade** pour récupérer les **enseignements** de l'enseignant connecté. La façade va chercher l'information demandée dans **TeacherJDBC**.

Une fois les enseignements récupérés et affichés dans une liste déroulante, l'utilisateur réalise diverses actions sur la fenêtre, sans qu'il y ait d'interaction avec le système :

- Sélection de l'enseignement choisi
- Sélection de la date
- Sélection du créneau horaire
- Ajout ou suppression de caractéristiques
- Renseignement de la capacité maximale de la salle souhaitée
- Renseignement de commentaires à destination du responsable

Une fois ces champs renseignés, l'utilisateur vérifie le nombre de salles disponibles en fonction de ces choix. Pour ceci, en cliquant sur un bouton « vérifier », **AskingView** crée une réservation. **AskingView** va par la suite demander à la **façade** si cette réservation que l'on vient de créer existe déjà dans le système. Ainsi, la façade **TeacherFacade** vérifie l'existence d'une telle réservation en interrogeant **BookingJDBC** grâce à la méthode `checkFreeRoom()` qui retourne un entier supérieur ou égal à zéro.

L'utilisateur valide enfin sa demande de réservation. **AskingView** informe la **façade** via la méthode `confirmBooking()`. La **façade** va enfin sauvegarder la demande de réservation grâce à la méthode `save()` qui s'effectue sur **BookingJDBC**.

Scénarios alternatifs :

- L'utilisateur appuie sur le bouton 'Quit' de la fenêtre: le cas d'utilisation se termine.
- L'utilisateur appuie sur le bouton 'Undo' de la fenêtre: la validation est annulée, le cas d'utilisation se termine.

Diagramme de séquence – Traitement des demandes de réservation

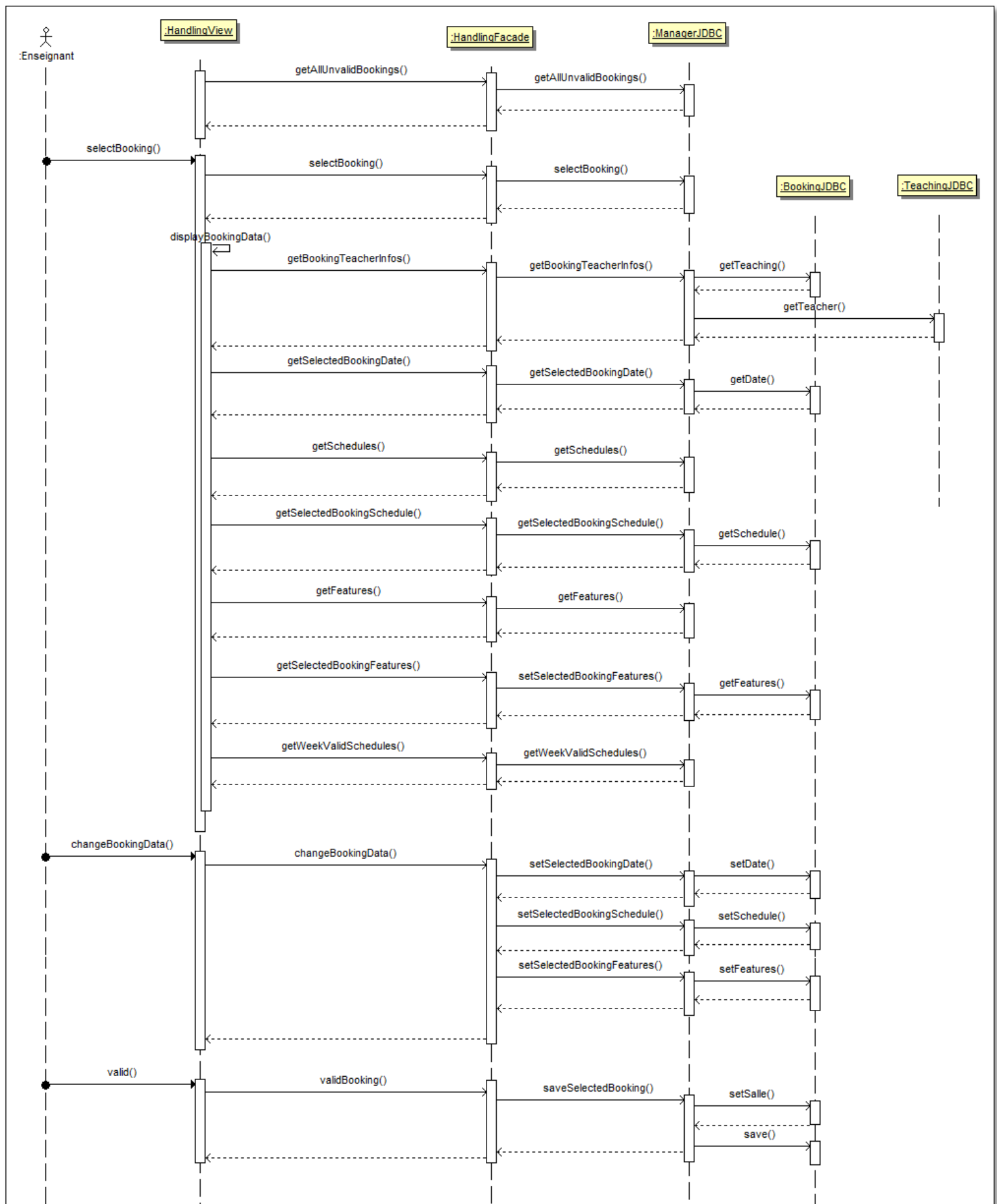


Figure 4 - Diagramme de séquence : traitement des demandes de réservation

Préconditions

L'utilisateur doit être logué.

Commentaires

La fenêtre **HandlingView** représente la fenêtre graphique pour le traitement des demandes de réservations. Celle-ci ne peut être accessible uniquement via le menu des responsables via le bouton "Traitement des demandes réservations".

Pour traiter ces demandes, **HandlingView** appelle la méthode "getAllInvalidBooking()" de la façade **HandlingFacade**. Celle-ci ne fait que déléguer : elle demande au gestionnaire **ManagerJDBC** de lui donner toutes les demandes de réservation en cours de validation, en allant les rechercher dans la base de données. Il va en même temps instancier chaque réservation (de la classe **Booking**).

Une fois les demandes affichées, le responsable en sélectionne une. **HandlingView** va donc faire une demande à la **façade** pour récupérer les informations concernant cette demande (**enseignement, enseignant, date, créneau** et **caractéristiques**).

La façade va chercher l'information demandée dans **TeachingJDBC** et **BookingJDBC** par le biais du gestionnaire **Manager**.

Une fois les informations récupérés et affichés en dessous du tableau contenant toutes les demandes, le responsable réalise diverses actions sur la fenêtre :

- Ajout ou suppression de caractéristiques
- Changement du créneau
- Changement de la date
- Accepter une demande (bouton 'Accept')
- Accepter une demande (bouton 'Refuse')

Une fois ces champs renseignés, le responsable doit alors accepter la demande de réservation.

Pour faire cela, **HandlingView** va demander à la façade d'attribuer une salle à la réservation concernée. La **façade** délègue ce travail au **Manager** qui va donc attribuer la salle choisit à l'objet **BookingJDBC** correspondant.

Après attribution de la salle, il va donc sauvegarder la demande de réservation grâce à la méthode **save()** qui s'effectue sur **BookingJDBC**.

Ainsi, la demande de réservation devient validée et n'apparaît plus dans le tableau des demandes de réservations non validées de la fenêtre.

Scénarios alternatifs

- L'utilisateur appuie sur le bouton 'Refuse' : la demande de réservation reste invalidée et le cas d'utilisation recommence.
- L'utilisateur appuie sur le bouton 'Quit' de la fenêtre: le cas d'utilisation se termine.
- L'utilisateur appuie sur le bouton 'Undo' de la fenêtre: la validation est annulée, le cas d'utilisation se termine.