HMIN 301 - TP3

Métamodèle UML

DALIE Basil
NGUYEN Huu Khang
NGUYEN Tran Tuan Nam
STEFANOVSKI Stefan
TRAN Thi Tra My

Remarques et modifications lors d'échange des modèles

Use Case:

Instances:

- "Extend" est une classe et pas une relation. Donc nous avons ajouté une instance de la classe "Extend" avec les compositions correspondant (extend, extension, extendedCase).
- Les inclusions sont eux-même une classe aussi.
- La relation entre le "Classifieur" et les "UseCase" est une composition.
- *The ExtensionPoints referenced by the Extendrelationship must belong to the UseCase that is being extended* → Nous avons remarqué que l'extension point doit être référencé au cas d'utilisation qui contient aussi une extension. Cette extension a une référence vers cette extension point de type "Reference Case".

Packages:

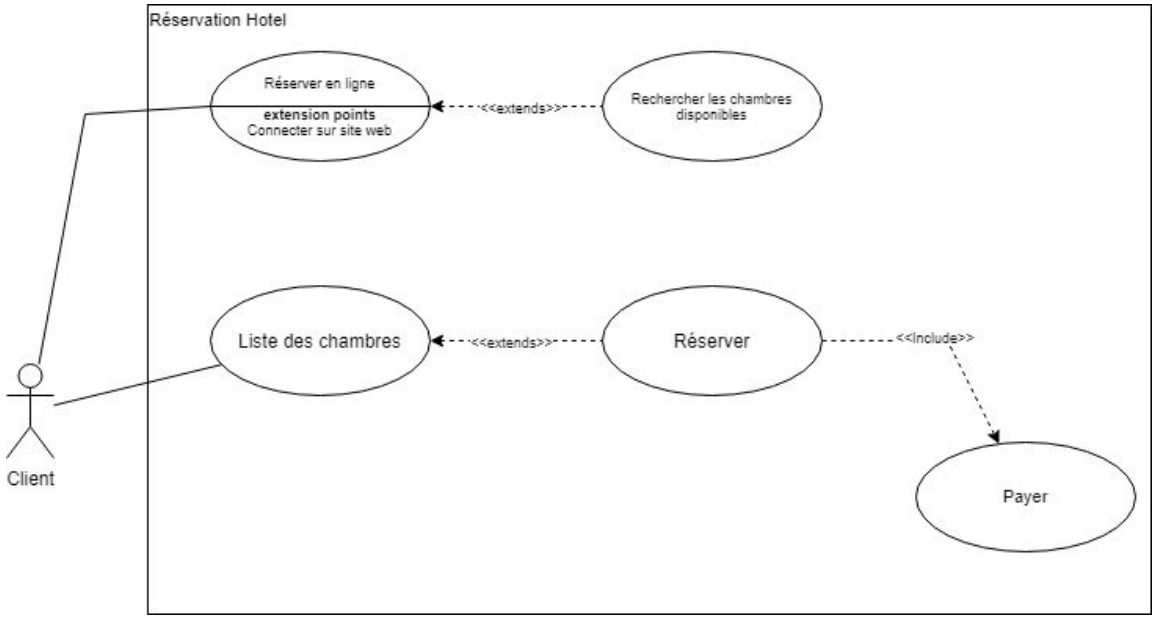
Instances:

- Dans le package "Scheduling" il existe trois classes avec le même nom "GroupFilter". Cela est remplacé par les classes correspondant: "GroupFilter", "Storage", et "SchedulingMenager".
- Lors du merge de trois packages il existe une seule instance de la classe "packageMerge". Nous avons résolu ce problème en rajoutant une autre instance donc une pour le merge entre les packages "Scheduling" et "Student" et une pour le merge entre les packages "Scheduling" et "Teacher"

UseCase : Réservation hôtel

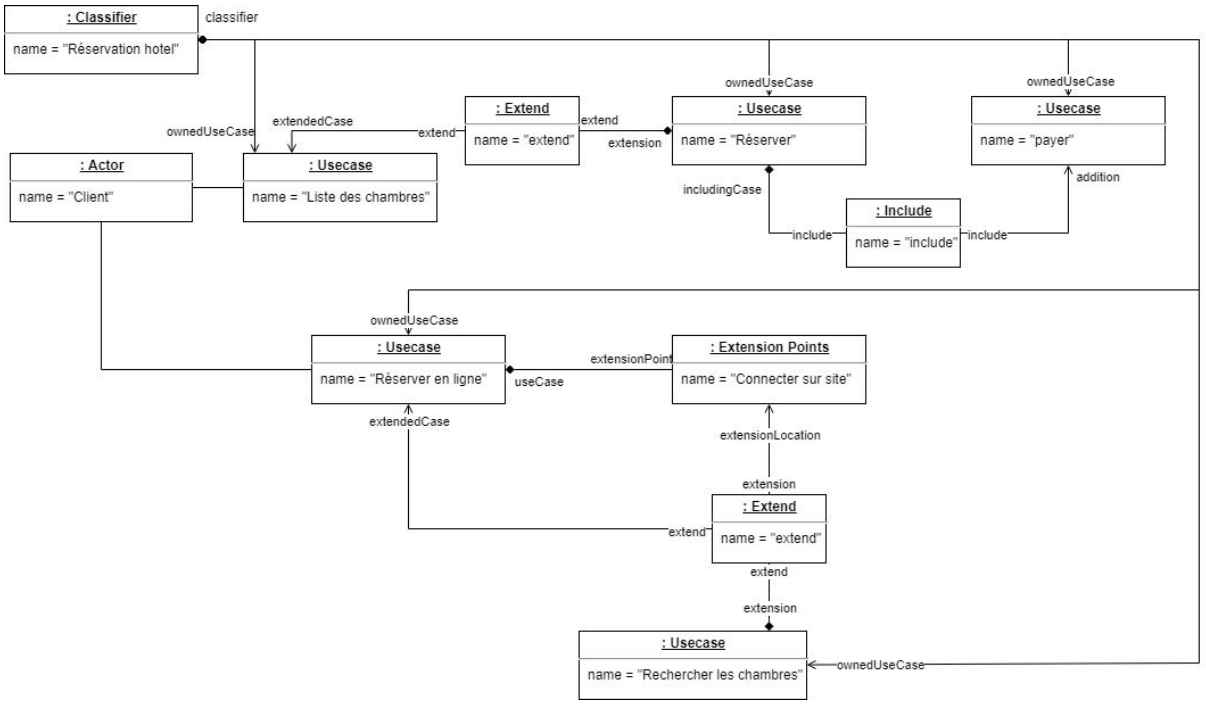
NGUYEN Huu Khang - TRAN Thi Tra My

Modèle UML - Diagramme Use Case



Nous avons le scénario d'une réservation d'une chambre dans l'hôtel via le diagramme cas d'utilisation. Les comportements chez un client seront décrits par des bulles du diagramme cas d'utilisation ci-dessus. Un client peut réserver une chambre directement à l'accueil en passant par une liste des chambres disponibles. Il peut aussi réserver en ligne en connectant sur le site internet de l'hôtel.

Diagramme d'instance du modèle UML use-case :



Tous les cas d'utilisation peuvent appartenir de 0 à 1 instance de la classe classifier. Dans notre diagramme cas d'utilisation, tous les cas appartiennent à l'instance "Réservation Hôtel".

L'instance de la classe "Actor", qui représente le client, a pour l'objectif d'interagir avec les cas d'utilisation proposés comme "réserver en ligne" ou "consulter la liste des chambres disponibles sur place".

L'instance de la classe "Use case" nommée "liste des chambres" contient une extension de la classe "Use case" nommée "réserver" et celle-ci est un cas étendu par le cas "liste des chambres". Cette relation est représentée par une instance de la classe "extend" entre 2 cas d'utilisation.

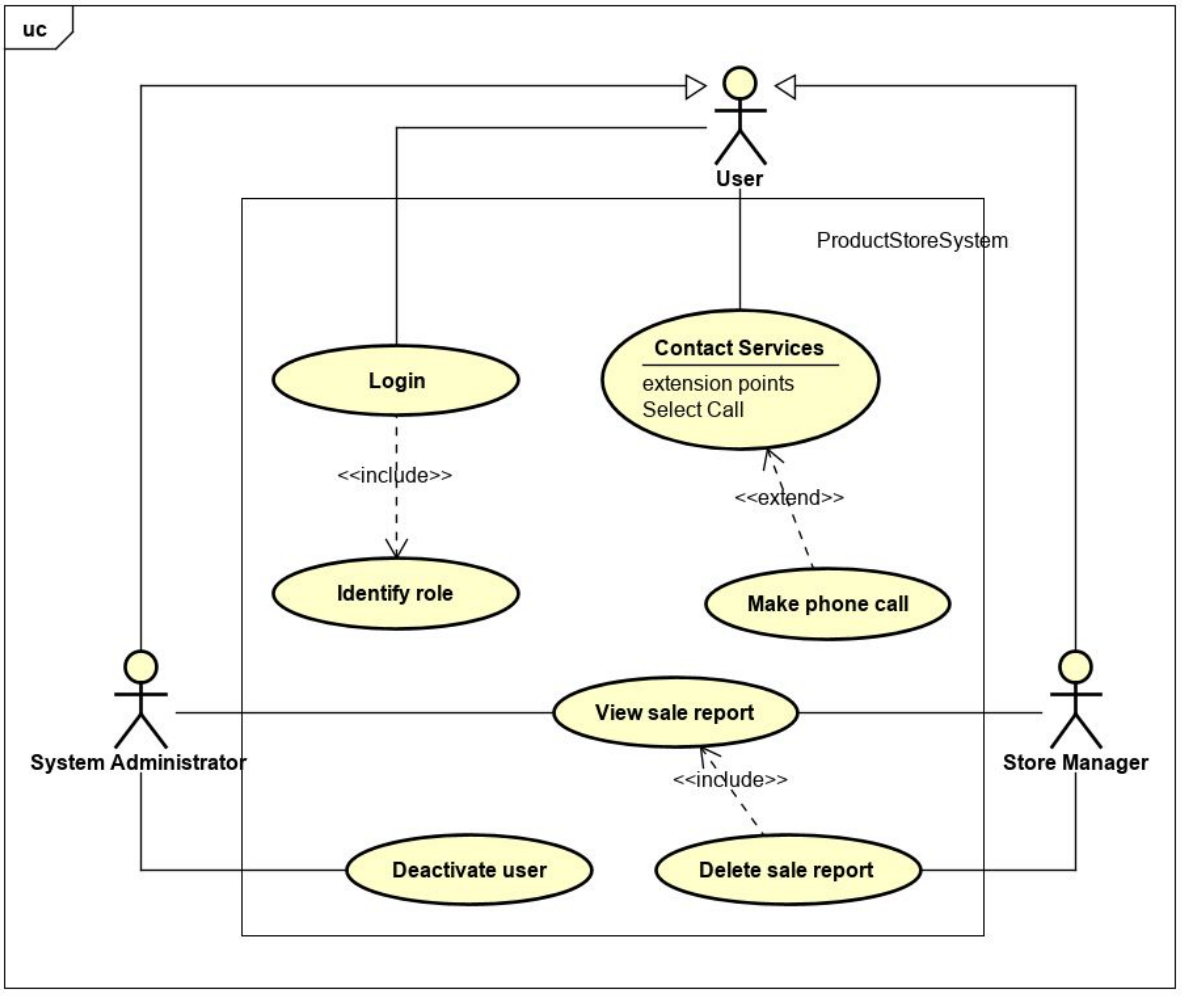
Ensuite, Il y a une relation "include" entre "réserver" et "payer" dans notre diagramme cas d'utilisation. La représentation sur le diagramme d'instance s'est fait entre l'instance "Use case" nommée "payer" et l'instance "Use case" nommée "réserver" d'où l'instance "payer" est une addition de l'instance "réserver" via une instance de la classe "include".

Concernant le cas "réserver en ligne", il s'agit une instance d'un point d'extension d'un cas d'utilisation. Cette instance est nommée "connecter sur site" qui permet à l'utilisateur d'accéder à l'instance Use case "Rechercher les chambres" en remplissant la condition de ce point d'extension.

UseCase : Product Store System

NGUYEN Tran Tuan Nam

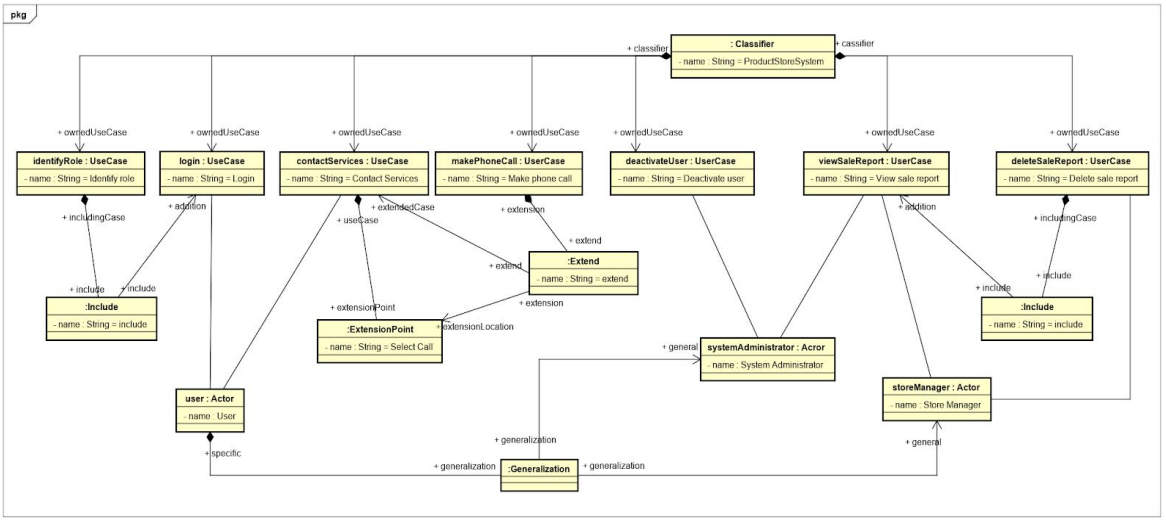
Modèle UML - Diagramme Use Case



Ce diagramme Use Case représente un système simple d'un Store (Alimentation par exemple) avec les fonctionnalités de base (Login, Contact Services), les gestions d'utilisateur et de produit et ce que l'utilisateur (identifiant par leur rôle) peut faire.

Il y a trois actors que ces use case appliquent: User, System Administrator et Store Manager qui sont également un User et qui ont les comportements d'un User (généralisation). User peut faire les Use Case Login (exigeant UseCase Identify Role) et Contact Services (étendant UseCase Make phone call). System Administrator et Store Manager ont un Use Case View sale report en commun et il est obligatoire pour réaliser Use Case Delete sale report de Store Manager.

Diagramme d'Instance du modèle UML Use Case



(*)

(*) *Dans le diagramme de classe de Astah qui a été utilisé pour construire ce diagramme d'instance, les visibilités des relations doivent être public (+) ou private (-) ou protected (#) ou package (~). Un diagramme d'instance de Astah doit être créé à partir d'un diagramme de classe déjà existant.*

Dans le contexte de ce modèle, les instances, sauf celle de la classe Actor (Utilisateur du système), ont relations composites avec l'instance de la classe Classifier nommée ProductStoreSystem et elles sont toutes liées, selon la structure du modèle Use Case ci-dessus.

Les instances de la classe Actor ont une relation de généralisation, représentée par une instance de Généralisation.

Il existe trois types de Use Case spéciaux: Extend, ce qui a Extension Point et Include. Ils sont montrés par les instances des classes correspondantes. En effet, dans le cas de Extend, l'instance de la classe Contact Services, par celle de Extension Point - Select Call et celle de Extend, est étendue par celle de Make phone call. Ensuite, celle de Login, par celle de Include, inclus celle de Identify Role; pareillement avec celle de Delete sale report.

Packages : Gestion d'université

DALIE Basil - STEFANOVSKI Stefan

Le diagramme de package réalisé représente une partie de l'organisation en packages d'une application de gestion d'une université tel que l'ENT de l'université de Montpellier. Les éléments de l'application concernés par ce diagramme sont la gestion des inscriptions étudiantes, la gestion des notes, le planning des étudiants et des professeurs, les emprunts (de livres, de films, etc.) à la bibliothèque, ainsi que l'interface graphique de l'application.
Chacun de ces éléments est associé à un package de premier niveaux.
Au deuxième niveau :

- Les "packages imbriqués" de "inscriptions" contiennent : un package associés aux pièces justificatives à fournir lors d'une inscription et un autre associé à la gestion des frais d'inscription
- Les "packages imbriqués" de "grades" contiennent : un package associé à l'enregistrement des notes des étudiants, et un autre associé au calcul de rang au sein d'un groupe d'étudiant
- Les "packages imbriqués" de "planning" contiennent trois sous-package : un est associé à la gestion du planning des étudiants, un autre à la gestion du planning des professeurs, et un dernier (scheduling) contient les classes nécessaires pour calculer l'ordonnancement idéal des cours en fonction des planning des étudiants et des professeurs.
- Le package library contient deux packages liés d'une part aux emprunts de livres, et d'autre part aux emprunts de films.

Le diagramme fait apparaître une fusion ("merge") de planning.student et planning.teachers vers planning.scheduling. L'intérêt de ce merge est de pouvoir combiner les deux classes Storage définies toutes deux dans planning.student et planning.teachers en une seule classe Storage de planning.schedulecomputer qui contient en même temps les informations des planning des étudiants et des professeurs, celle-ci a également accès à la classe GroupFilter de planning.student qui permet de filtrer les cours plannifiés des étudiants en fonction des groupes d'appartenance (groupe d'anglais, options choisies, etc.). Par ailleurs, on peut toujours importer les packages "planning.student" et "planning.teachers" séparément.

Modèle UML

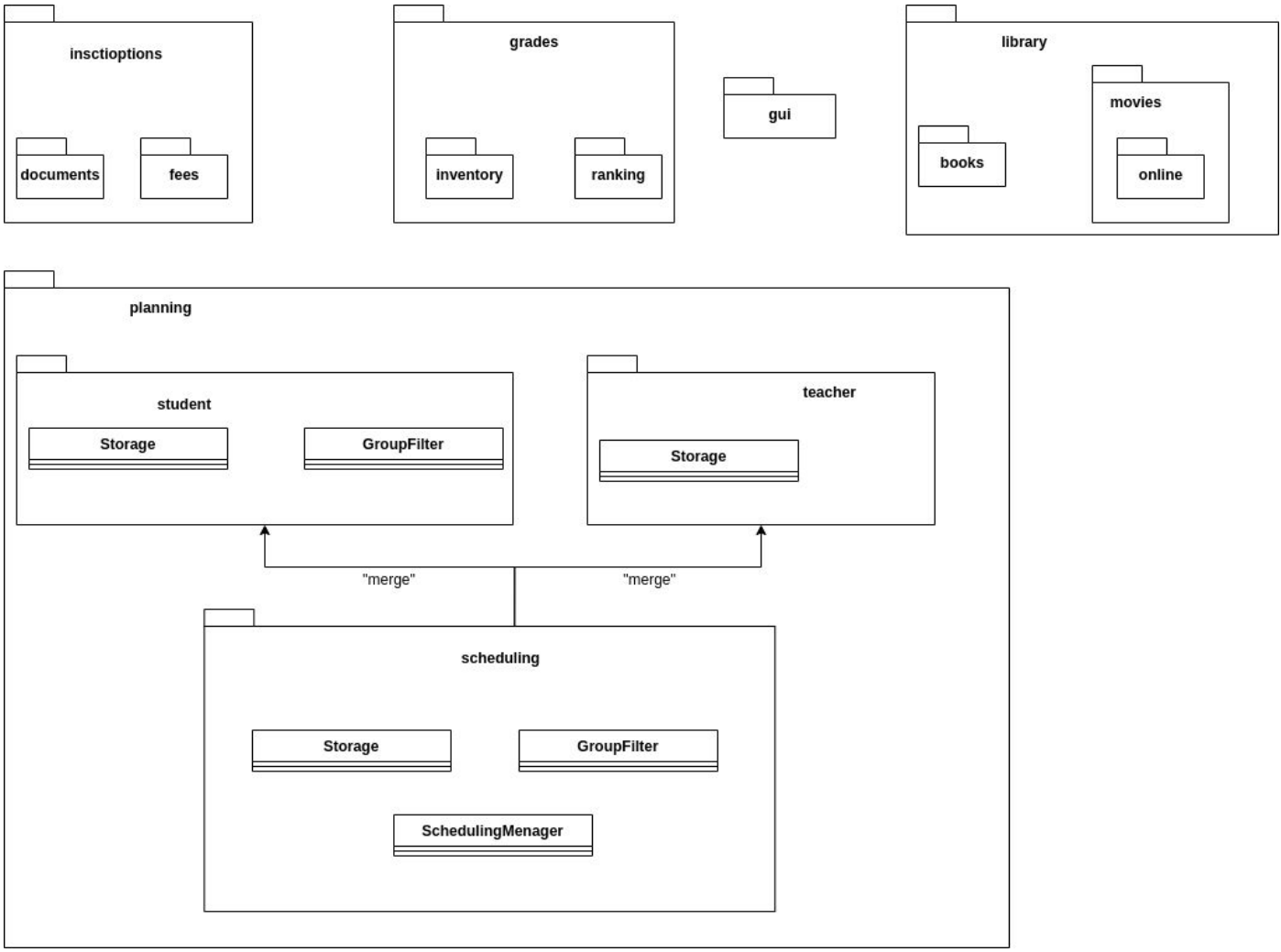

insctioptions
documents
fees
grades
inventory
ranking
gui
library
books
movies
online
planning
student
Storage
GroupFilter
teacher
Storage
"merge"
"merge"
scheduling
Storage
GroupFilter
SchedulingMenager

Diagramme d’instance du modèle UML - Packages