HomeSharingExpenses

Nom	Thiam Falilou	Numéro d'auditeur	0G5DROAVAU6
Version	1.01	Date de mise à jour	18/05/17

Table des matières

1. Objectif du document 3
2. Architecture 4
2.1. Contraintes techniques 4
2.2. Packages et dépendances 4
2.3. Déploiement 5
3. Technologies utilisées 6
3.1. Serveur web 6
3.2. Stockage des données 6
3.3. Couche de persistance 6
3.4. Couche métier 6
3.5. Couche service 6
3.6. Couche présentation 6
3.7. Environnement de développement 6
3.8. Test unitaire 6
4. Cas d'utilisation 7
4.1. Authentification 7
4.1.1. Se connecter 7
4.2. Mise en œuvre des groupes 8
4.2.1. Créer un groupe 8
4.2.2. Ajouter une liste de courses 9
4.3. Gestion d'une liste de courses 10
4.3.1. Visualiser une liste de courses 10
4.3.2. Valider un achat de la liste de courses 11
5. Regroupement des classes 12
5.1. Groupe domaine 12
5.2. Groupe domaine et cycle de vie 13
5.3. Groupe Service 13
5.4. Groupe interface utilisateur et système 14
5.5. Terminologie 15

1. Objectif du document

Ce document décrit la phase de conception de l'application HomeSharingExpenses. Nous nous servirons donc du document d'analyse pour traiter de l'architecture de l'application, exposer les différents choix de technologies.

2. Architecture

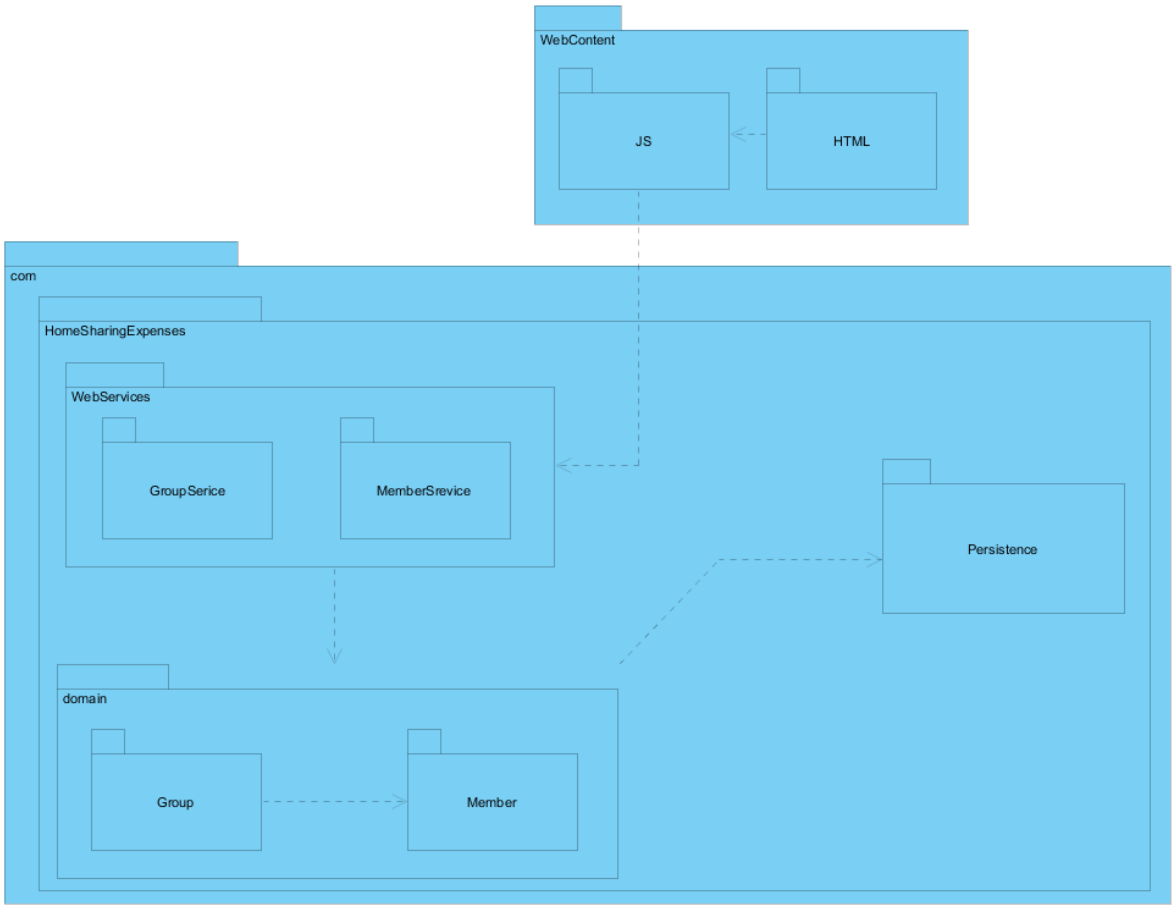
2.1. Contraintes techniques

La partie applicative de cette application sera basée sur Java. Nous devrons donc respecter la spécification JEE 8.
Cette application sera basée sur des web services Jax-ws.
Concernant sa partie persistance, nous mettrons en œuvre JPA.
Pour sa partie présentation, nous optons pour une application Web uniquement. Nous devrons donc respecter les normes HTLM en vigueur (celles du W3C).

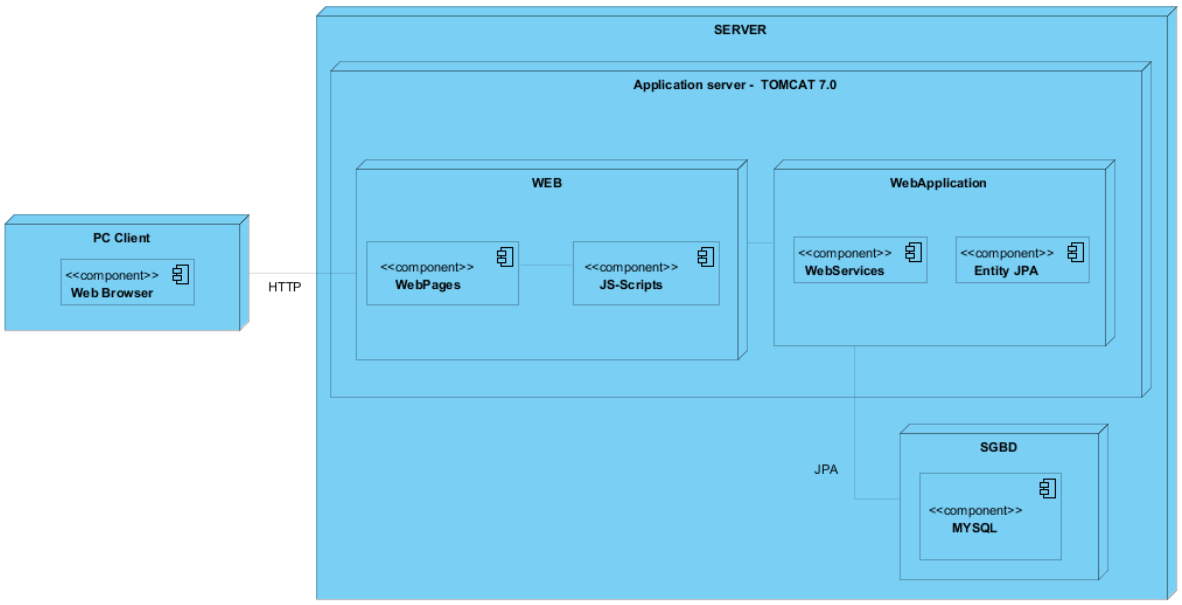
Point de vue déploiement, dans un premier temps, serveur applicatif et base de données se trouveront sur la même machine.

2.2. Packages et dépendances

Nous pouvons résumer les packages de notre application grâce au schéma suivant :



2.3. Déploiement



3. Technologies utilisées

3.1. Serveur web

Nous choisirons d'utiliser Tomcat v7.0 pour le serveur applicatif.

3.2. Stockage des données

Nous utiliserons MySQL en version 5.1. Le driver JDBC sera mysql-connector-java-5.1.21.

3.3. Couche de persistance

Concernant la partie stockage des données, nous utiliserons JPA. (javax persistence 2.0.3)

3.4. Couche métier

Notre couche métier sera simplement des POJOs, annotés avec des annotations JPA.

3.5. Couche service

Notre couche service sera implémentée grâce à des webServices, nous utiliserons l'API JAX-WS RT 2.2.5.

3.6. Couche présentation

Pour notre couche présentation, nous nous servirons d'Angular JS en 1.4.5 dans nos pages HTML, afin d'obtenir une présentation dynamique.
Pour finir nous nous servirons de CSS en version 4.

3.7. Environnement de développement

Pour notre environnement de développement nous utiliserons les technologies suivantes :

- Eclipse (Neon.3 Release (4.6.3))
- Java JDK 1.8.0_20
- Maven 3.0.3
- Visual Paradigm 14.1

3.8. Test unitaire

Nous utiliserons JUnit pour nos tests unitaires.

4. Cas d'utilisation

Nous nous servirons de Java Script, à travers Angular afin de mettre à jour les données de nos pages. C'est donc grâce à ces scripts que nous communiquerons avec les services présents sur notre serveur d'application. Nous effectuerons des requêtes AJAX sur nos serveurs REST. Nous utiliserons le format JSON pour échanger entre notre couche présentation et nos webservices.
Nous décidons de représenter ces scripts comme des contrôleurs dans nos diagrammes de séquence.
A noter aussi que nous avons choisi d'utiliser la technologie JPA pour satisfaire la persistance de nos objets en base de données. Ce sera ici le rôle de notre classe EntityManager que d'opérer l'ensemble des méthodes CRUD sur ces objets.

4.1. Authentification

4.1.1. Se connecter

4.1.1.1. Liste des objets candidats

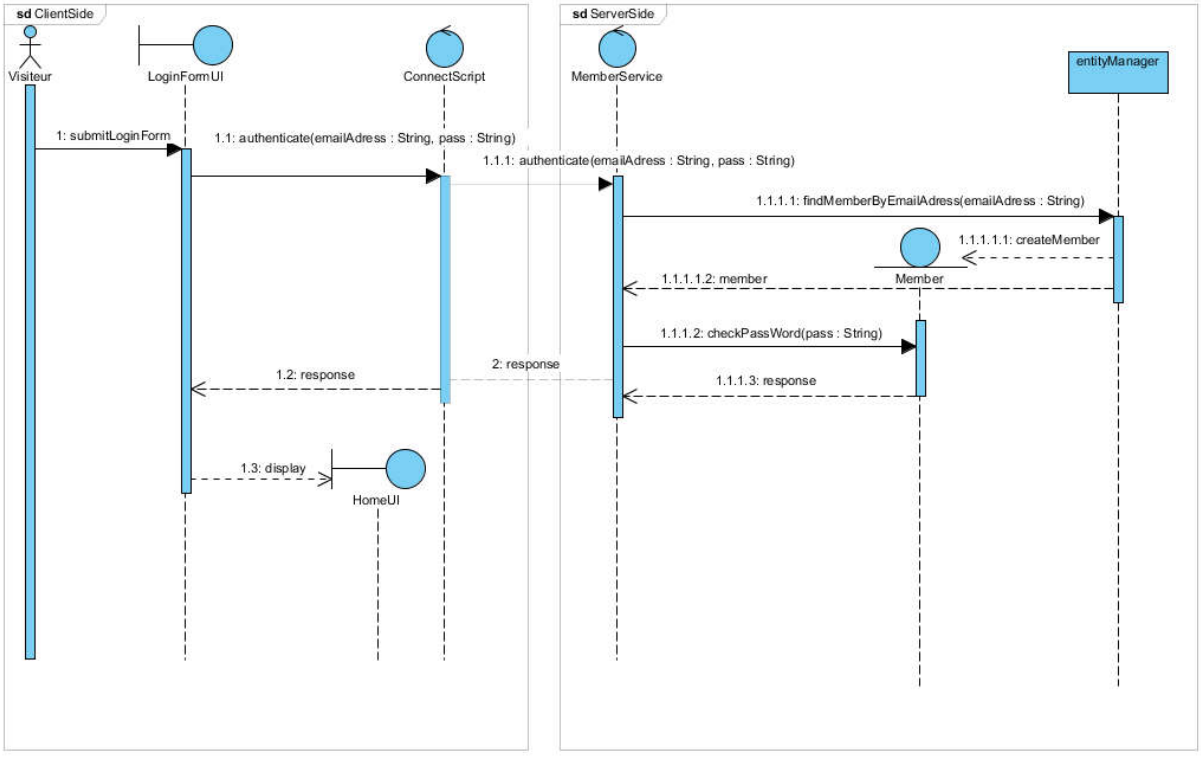
<<Entity>> : Member
<<lifecycle>> : EntityManager
<<control>> :

- MemberService
- ConnectScript

<<boundary>> :

- LoginFormUI
- HomeUI

4.1.1.2. Description des interactions entre objets



4.2. Mise en œuvre des groupes

4.2.1. Créer un groupe

4.2.1.1. Liste des objets candidats

<<Entity>> :

- Member : Objet métier représentant un membre
- Group : Objet métier représentant un groupe. Stocke les informations relatives à un groupe (liste d'achats, membres, budget etc…)

<<lifecycle>> : EntityManager

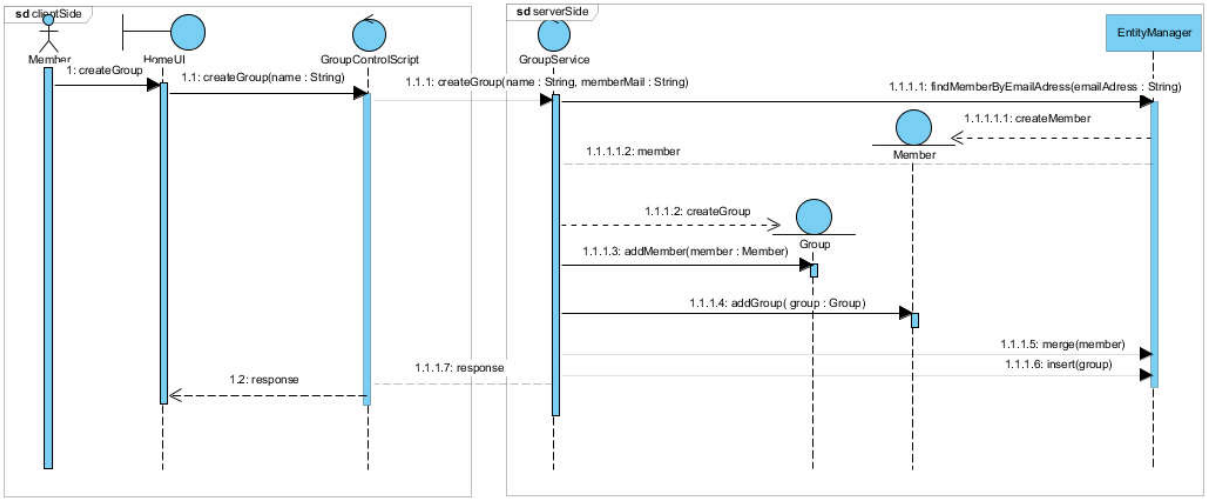
<<control>> :

- GroupService
- GroupControlScript

<<boundary>> :

- HomeUI

4.2.1.2. Description des interactions entre objets



4.2.2. Ajouter une liste de courses

4.2.2.1. Liste des objets candidats

<<Entity>> :

- Group : Objet métier représentant un groupe. Stocke les informations relatives à un groupe (liste d'achats, membres, budget etc…)
- ShoppingList : Objet métier qui correspond à une liste d'achats. Un shoppingList est principalement composé d'une liste d'Entity.

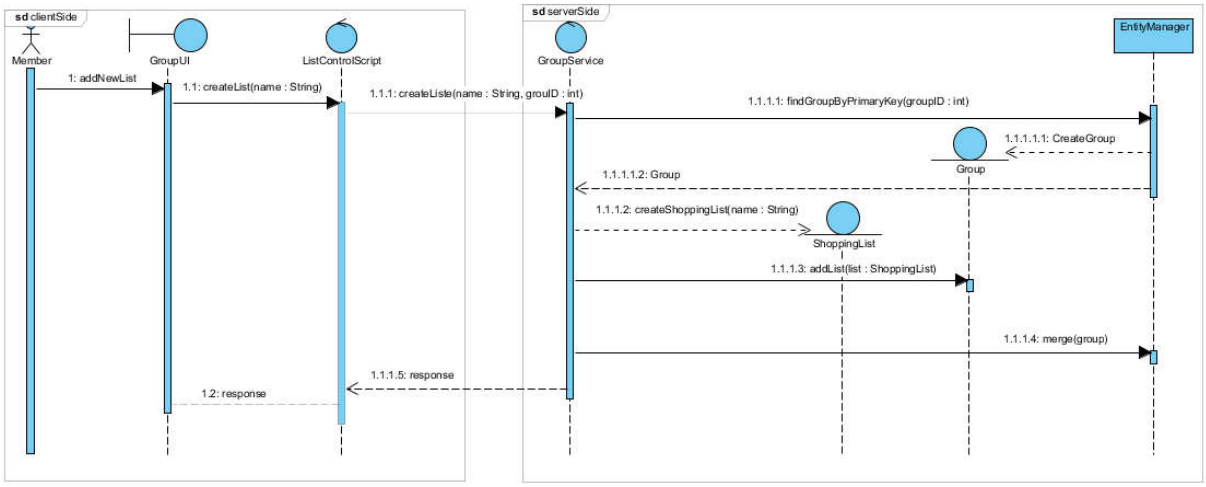
<<lifecycle>> : EntityManager

<<control>> :

- GroupService
- ListControlScript

<<boundary>> : GroupUI

4.2.2.2. Description des interactions entre objets



4.3. Gestion d'une liste de courses

4.3.1. Visualiser une liste de courses

4.3.1.1. List des objets candidats

<<Entity>> :

- Group : Objet métier représentant un groupe. Stocke les informations relatives à un groupe (liste d'achats, membres, budget etc…)

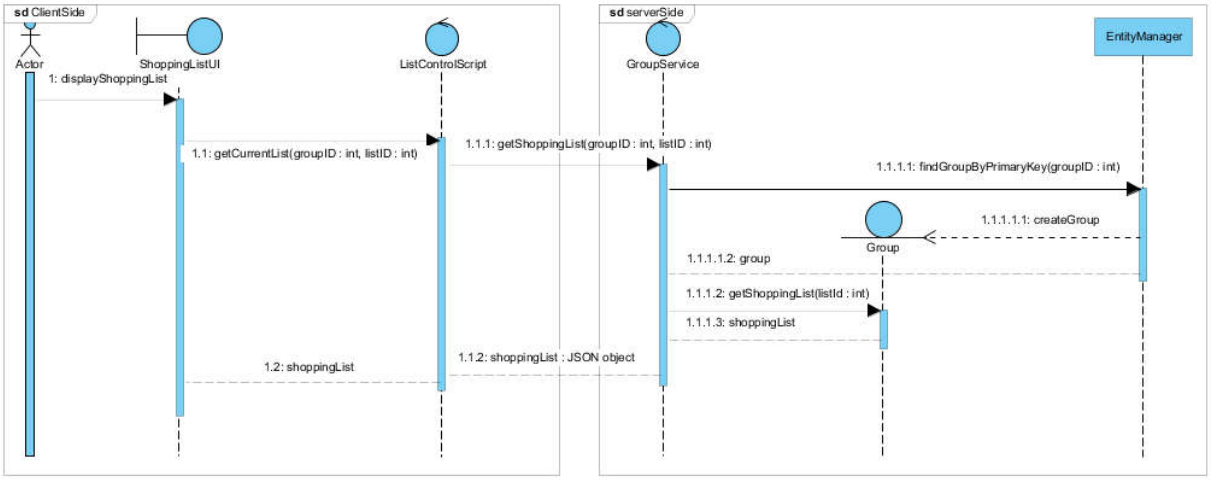
<<lifecycle>> : EntityManager

<<control>> :

- GroupService
- ListControlScript

<<boundary>> : ShoppingListUI

4.3.1.2. Description des interactions entre objets



4.3.2. Valider un achat de la liste de courses

4.3.2.1. List des objets candidats

<<Entity>> :

- Group : Objet métier représentant un groupe. Stocke les informations relatives à un groupe (liste d'achats, membres, budget etc…)
- Member : Objet métier représentant un membre
- ShoppingList : Objet metier qui correspond à une liste d'achats. Un shoppingList est principalement composé d'une liste d'Items.
- Item : Object metier correspondant à une entrée d'une liste de course. Il regroupe les informations relatives au produit, qui ce produit concerne son prix… Il est aussi composé d'un état (ToDo ou Done).

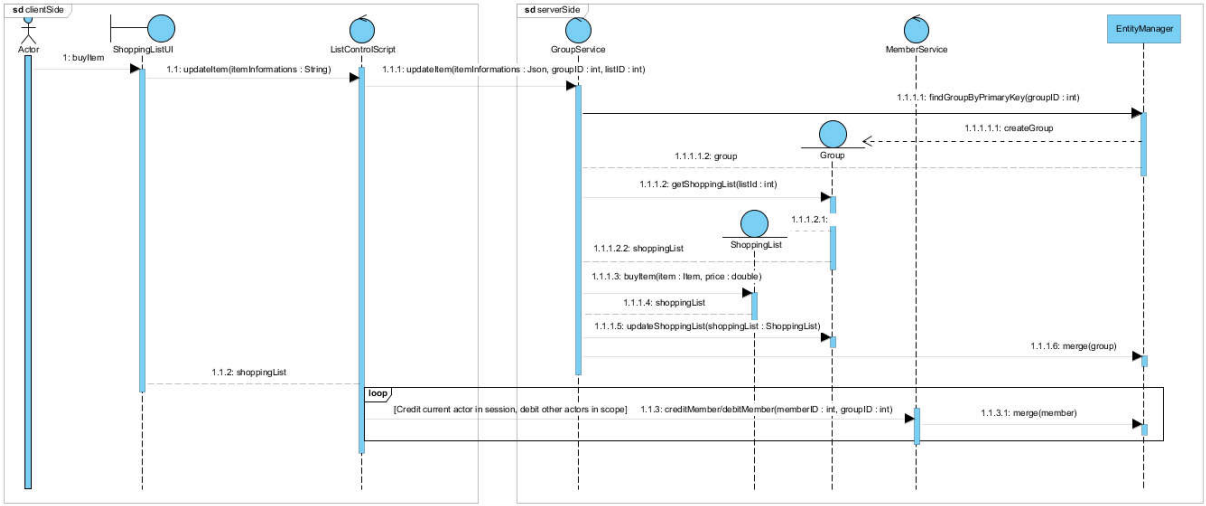
<<lifecycle>> : EntityManager

<<control>> :

- MemberService
- GroupService
- ListControlScript

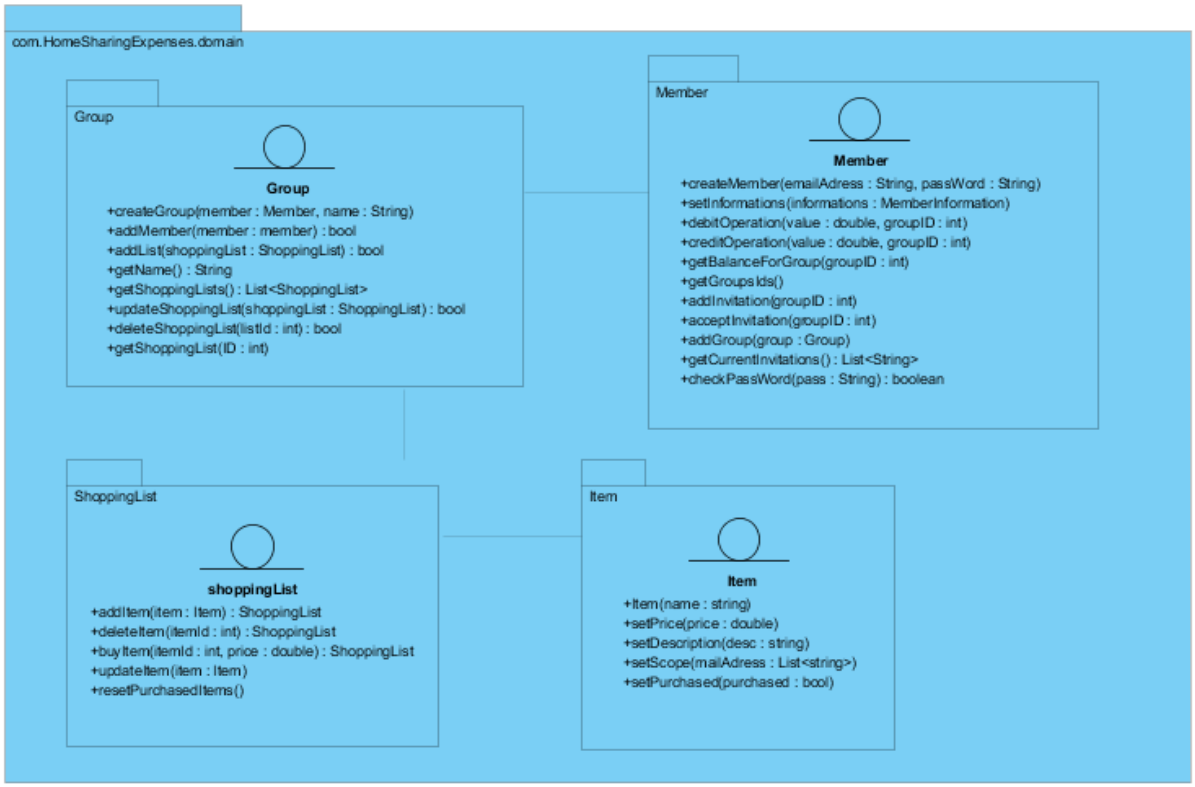
<<boundary>> : ShoppingListUI

4.3.2.1. Description des interactions entre objets

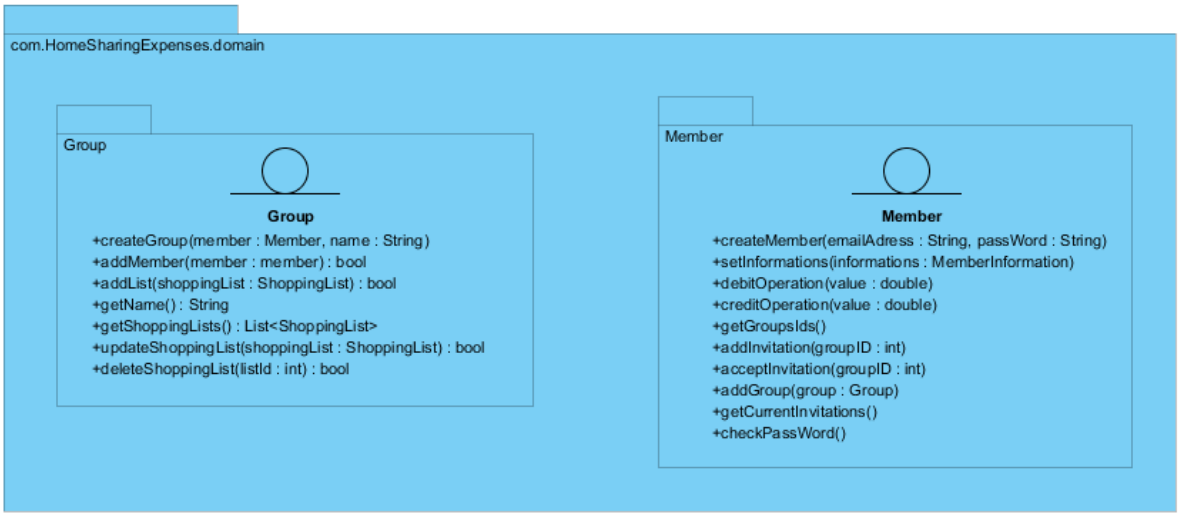


5. Regroupement des classes

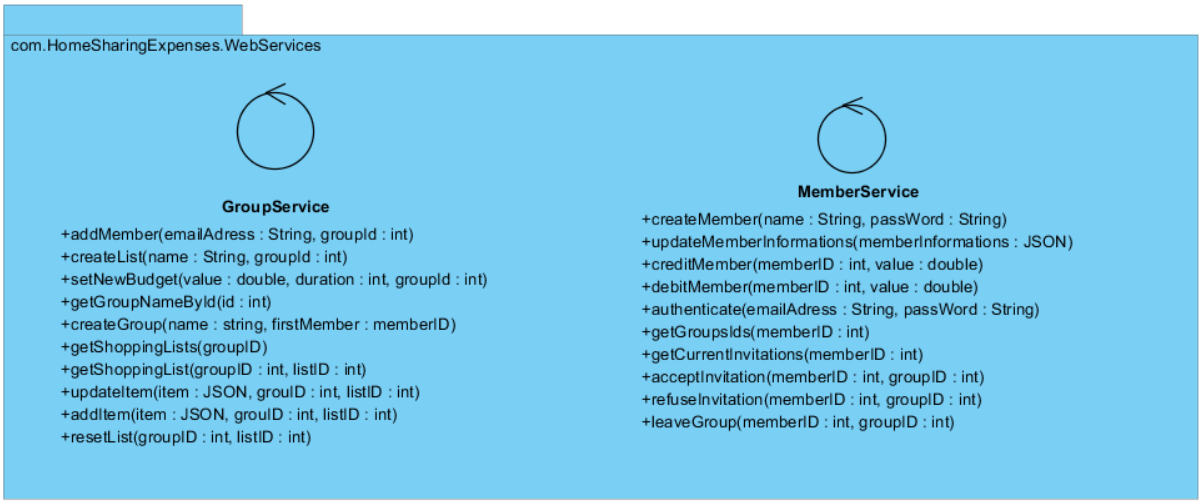
5.1. Groupe domaine



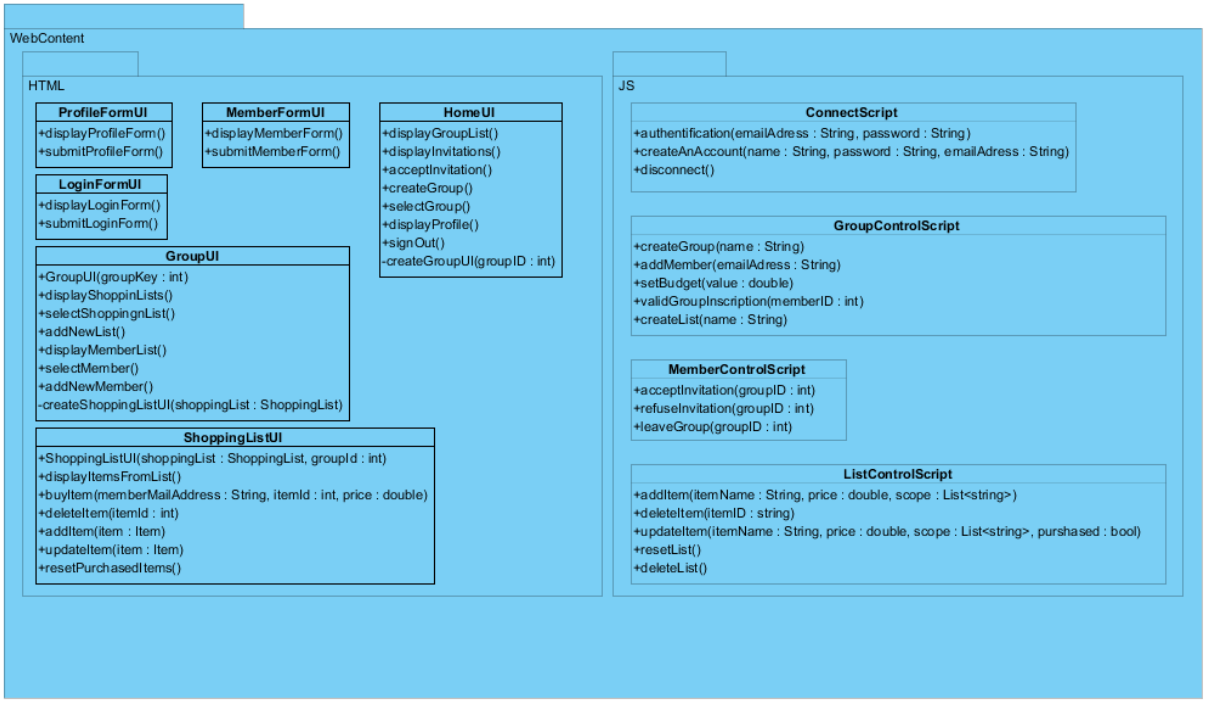
5.2. Groupe domaine et cycle de vie



5.3. Groupe Service



5.4. Groupe interface utilisateur et système



5.5. Terminologie

Entity	Designe les objets métier de notre application.
Control	Objet à l'intrerface entre les boundary et les entity/LifeCycle. Permet d'accéder aux objets métier en fonctions de requêtes formulées par les boundary.
Boundary	Interface entre utilisateur et système. Un cas classique serait un formulaire pour un utilisateur.
Lifecycle	Objet servant à d'une part persister les objets métier en base, et d'autre part à les récupérer au besoin. Il est donc responsable du cycle de vie des objets dits persistant de notre application.
Groupe	Le groupe correspond dans cette application à un foyer. C'est en faisant partie d'un même groupe que plusieurs utilisateurs pourront interagir sur des listes partagées. Il n'y a pas de notion d'administrateur. N'importe quel membre d'un groupe donné peut y inviter un autre membre, ajouter ou supprimer des listes. Un utilisateur peut faire partie de plusieurs groupes. Si un groupe n'a plus d'utilisateur, il sera automatiquement supprimé.
Liste de courses	Sous forme de prise de note, une liste de course va être composée de deux catégories. Une catégorie ToDo qui regroupera tous les achats à effectuer, qui n'ont pas étés réalisés. Une catégorie Done dans laquelle chaque item de provient de la catégorie ToDo après avoir fait l'objet d'un achat. Chaque item de cette liste sera composé de : • Nom du produit à ajouter • Indication sur le produit (Optionnel) • Scope : qui ce produit concerne : liste de choix d'identifiants présents dans le groupe. • Prix estimé (Optionnel) • Urgence de l'achat : faible, modéré, urgent (Optionnel)
Membre	Un membre est un visiteur qui s'est connecté sur l'application en utilisant son compte.
Visiteur	Est un utilisateur qui ne possède pas de compte sur l'application ou qui visite le site s'en s'y être connecté.