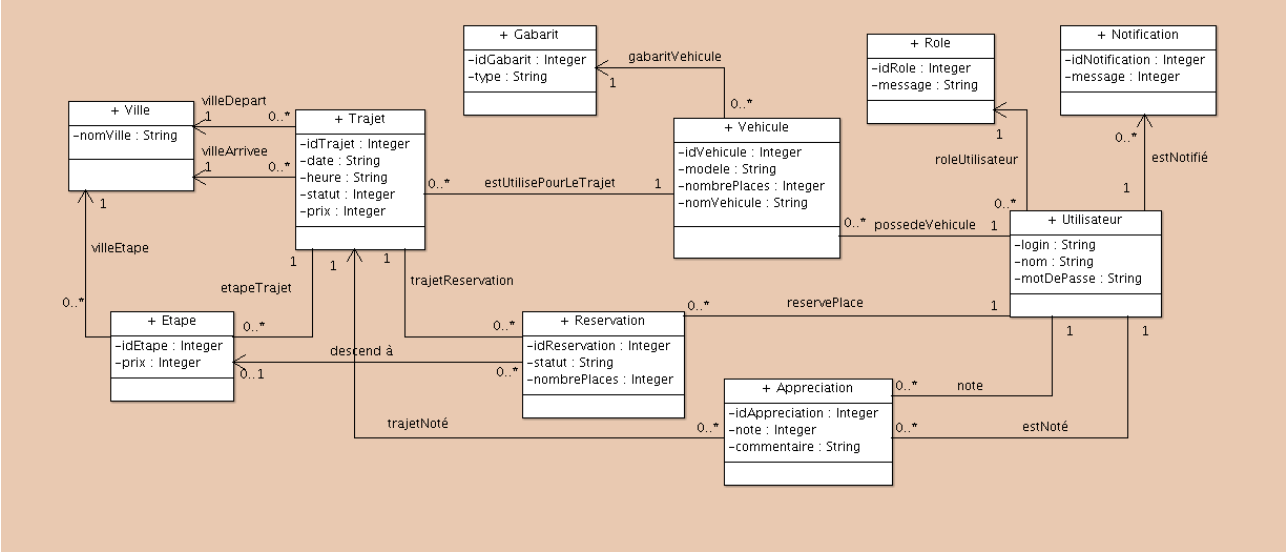
ARCHITECTURES APPLICATIVES RÉPARTIES

Gallet Benoît, Herrmann Emmanuel, Réty Martin

I. ARCHITECTURE

1. ARCHITECTURE RELATIONNELLE

Voici notre diagramme de classe, également disponible en annexe pour l'avoir de façon plus lisible.



Notre architecture est composée suivant deux pôles principaux : le Trajet et l'Utilisateur.

Plusieurs classes gravitent autour du Trajet : la classe Ville, reliée deux fois pour désigner la ville de départ et celle d'arrivée du Trajet, et la classe Étape, pour indiquer les éventuelles étapes du trajet et leurs prix. Pour connaître le prix du trajet total, c'est le prix contenu dans la classe Trajet. Plusieurs choix d'implémentations ont été faits ici :

1. La ville d'arrivée n'est pas une étape.
2. Le prix des étapes doit être inférieur à celui du trajet total.
3. Deux étapes ne peuvent pas concerner la même ville.

En ce qui concerne la classe Utilisateur, elle est associée à un rôle, qui peut être soit admin, soit utilisateur. Il y a aussi des notifications, qui alertent l'utilisateur d'une pléthore

d'actions différentes faites par lui ou d'autres utilisateurs (par exemple l'annulation du trajet par le conducteur, l'acceptation d'une réservation, …).

Ces deux grands pôles interagissent via trois classes : Véhicule, Reservation, Appreciation.

- Vehicule. Un véhicule est associé à un gabarit. Un utilisateur peut posséder plusieurs véhicules, et il est identifié comme conducteur d'un trajet si son véhicule est relié à un trajet.
- Reservation. Les réservations, quant à elles, désignent les passagers d'un trajet. Une réservation peut être faite par un conducteur sur son propre trajet (il réserve une place pour un ami ou autre).
- Appreciation. Une appréciation relie un trajet à deux utilisateur. L'un est noté, l'autre note. Une appréciation est composée d'une note entière entre 1 et 5, et d'un commentaire facultatif.

2. Architecture logicielle

a. Web

Au niveau de la couche web, nous avons trois Contrôleurs. Le premier, ControleurAnonyme, gère toutes les actions que peut faire un visiteur non authentifié. Le deuxième, ControleurAdmin, gère les actions que fait un utilisateur ayant le rôle admin. Le troisième, ControleurUtilisateur, est là pour exécuter toutes les taches commandées par un utilisateur normal, et authentifié.

b. Métier

Pour accéder aux entités et les modifier, nous avons trois façades principales, chacune correspondant comme les contrôleurs au rôle de l'utilisateur courant : MaFacadeAnonyme, MaFacadeAdministrateur, MaFacadeAnonyme. Comme à la fois l'anonyme et l'utilisateur peuvent faire des recherches, toutes les méthodes permettant la recherche sont synthétisées dans un EJB spécial. Enfin, le dernier EJB représente l'automate, effectuant des taches basiques ne dépendant pas de l'utilisateur, telles que hasher un mot de passe, créer une notification, tester une date, …

c. Sécurité

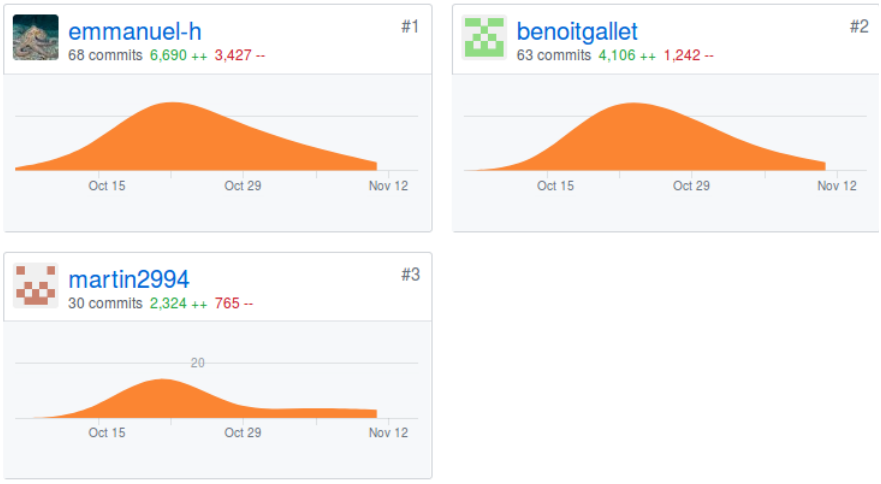
L'application est sécurisée à différents niveaux. Tout d'abord au niveaux des urls, où chaque personne n'a le droit d'afficher que celles que son rôle lui permet, et au niveau des fonctions grâce aux annotations @RolesAllowed. Quand l'utilisateur demande à accéder à une ressource demandant des droits alors qu'il n'est pas authentifié, il est redirigé vers une page de login, qui redirige à son tour vers un ControleurGeneral. Celui-ci redirige l'utilisateur en fonction de son rôle sur la page d'accueil appropriée. Les mots de passe dans l'application sont tous hashés en SHA-512.

II. Répartition des tâches

La base du projet a été faite en groupe : la création des entities, de leurs attributs et méthodes, des interfaces des façades et des méthodes des servlets. Ensuite, nous avons découpé l'application en fonctionnalités, et chacun a fait une fonctionnalité, en faisant donc le code dans la servlet, la façade et potentiellement dans les entités. Par exemple, Martin s'est notamment occupé de la partie anonyme et de la recherche, Benoît de la partie administrateur et l'historique, et Emmanuel les paramètres et la création des trajets. Pour les fonctionnalités qui se sont révélées plus compliquées que prévues, nous avons fonctionné en binôme pour avancer plus rapidement et ne pas bloquer trop longtemps dessus.

Pour le versionnage, nous avons utilisé un dépôt git, avec une branche pour chacun, afin de pouvoir avancer en parallèle.

Voici le total des commits pour chaque membre du groupe. Martin en a fait beaucoup moins que les deux autres suite à un problème sur son pc (ses commits étaient faits sur les ordinateurs des autres membres du groupe, et passait donc par leurs comptes git).



Le projet a été fait entièrement sur le git personnel, et a été push sur le commit stash à la fin.

III. UTILISATION

La première chose à faire pour pouvoir utiliser notre application est d'aller dans le dossier Modifications JBoss, de copier le fichier roles.properties dans le répertoire standalone de son JBoss, et copier ce qu'il y a dans le fichier modif_standalone.xml dans le standalone.xml de JBoss pour rajouter la balise security-domain.

Nous avons différents utilisateurs, dont le mot de passe est à chaque fois égal au login :

admin, alice, bob, charlie, diane, eric, fiona et gaston.

Le remplissage de la base sql se fait automatiquement via le script insert.sql. Ce remplissage est fait de manière à pouvoir tester le plus de cas possibles.

Les différentes URLs sont :

- http://localhost:8080/CavardageWeb/ pour l'accueil du site
- http://localhost:8080/CavardageWeb/ControleurAdmin pour la page d'accueil de l'administrateur
- http://localhost:8080/CavardageWeb/ControleurUtilisateur pour la page d'accueil d'un utilisateur authentifié
- http://localhost:8080/CavardageWeb/ControleurGeneral Si on veut forcer l'authentification

Si on est authentifié est qu'on essaie d'accéder à l'accueil (anonyme) du site, on est automatiquement redirigé vers la page d'accueil correspondant à son rôle.