UNIVERSITE D'ORLEANS

RAPPORT PROJET M1MIAGE

Responsables : Yohan Boichut et Frédéric Moal

« BusTraM »

GROUPE 9

Réalisé par :

EL BARDI FATIMA EZZAHRAA

ALILECHE KAMELIA

1- INTRODUCTION :

Dans le cadre de nos études en première année de master MIAGE, nous avons réalisé une application qui nous a permis d'employer les connaissances et les techniques que nous avons apprises durant le premier semestre, nous avons développé un système de gestion de transports que nous l'avons nommé « **BusTram** »

BusTram est une application qui permet à un utilisateur « client » de se connecter/déconnecter à sa plateforme, s'inscrire à un abonnement mensuel ou annuelle, acheter des tickets consommable à partir d'une consommation et valider son ticket sur un terminal.

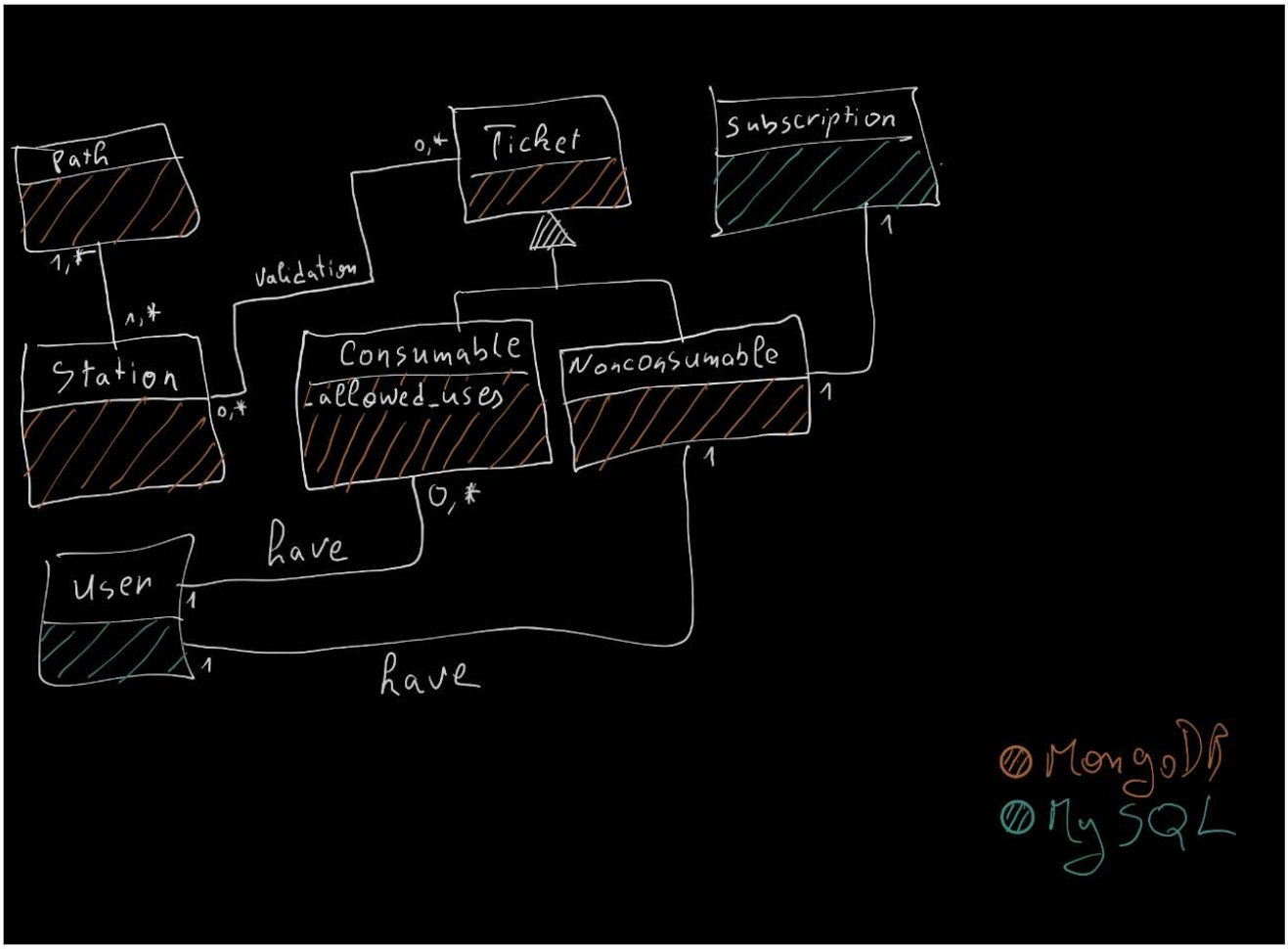
Cette application contient deux parties partie **Back-end** que nous avons développé pour le premier semestre et la partie **Front-end** à développer pour le deuxième semestre.

Nous avons employé nos compétences et nos connaissances pour aboutir une application qui contient les fonctionnalités dédié et en même temps flexible, « **BusTram** » est réalisé par ELBARDI fatima ezzahraa et ALILECH Kamelia sous le suivi de de Yohan BOICHUT & Frédéric MOAL.

2- Architecture Logicielle :

Diagramme de classe :

Ce diagramme explique l'architecture logiciel, le fonctionnement de notre application et les interaction entre les différente classes:



Ce diagramme généré explique la structure de l’application ainsi que la hiérarchie des classes .

3- Quelques choix techniques :

1- Docker Compose

un outil qui permet de décrire (dans un fichier YAML) et gérer (en ligne de commande) plusieurs conteneurs comme un ensemble de services inter-connectés. Cet outil vient simplifier la vie aux utilisateurs de docker.

2- Base de donnée Nosql :

Mongodb

Stocker et gérer : les Stations, les tickets et ses validations et les trajets.

3- Base de donnée relationnelle :

Mysql

Stocker et gérer : les clients (Users) et les abonnements (Subscriptions) .

4- Langage de programmation :

Java

Nous avons développé l'application avec le langage Java

5- IDEA

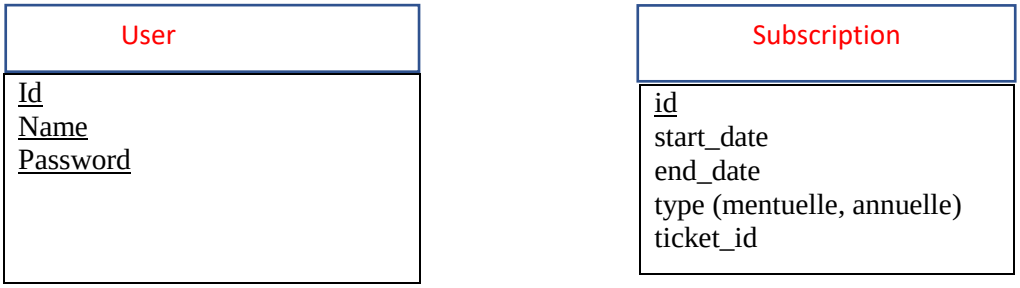
Intellij

IntelliJ IDEA également appelé « IntelliJ », « IDEA » ou « IDJ » est un environnement de développement intégré de technologie Java destiné au développement de logiciels informatiques

Dao(Data Access Object) contient simplement toutes les classes qui ont le rôle d'interagir avec la base de données, soit pour récupérer, insérer, modifier ou bien supprimer des données. Ces classes sont regroupées selon le modèle de données qui doit être manipulé. A titre d'exemple l'interface TicketsDao avec son implémentation TicketsDaoMongo est responsable de récupérer, insérer, modifier ou bien supprimer des Tickets dans la base de données Mongo.

4- Choix de conception :

1- MySQL (Abonnements & Clients) :



2- MongoDB (Validations & Trajets) :

Modélisation en Json :

- Tickets :

{ “id “:

” user_id ’’,

“status (active, expired...)”

“allowed_uses” (0 : expired | -1 : non-consumable| n : number of uses allowed),

“number_uses_allowed” (nombre d’utilisation)

“consumable “(True, False),

“price” :

“Ticket_validations”[

“ station_id :’’,

“validation_date :’’

….}]

}

- Stations :{

“id :’’

“nickname : ‘’,

“Station_validations “:[{

“ ticket_id :’’,

“ validation_date :’’

…}]

“paths”:[{

“id_path”

…}]

}

-Paths :{

“path_id ’’

“name”

“Stations “ [{

“ station_id ’’

…}]

}

Pourquoi le choix de 3 collections (station, tickets,paths) dans la base de données en mongoDb ?

Pour faciliter les traitements et la manipulation des données, il sera difficile de récupérer les données si nous avons utilisé des collection imbriqués, pour ça, nous avons utilisé la relation Many-Many (voir annexe).

Design pattern

Nous avons utilisé le design pattern Bridge dans le dataSource et tous les Dao et la façade (voir les diagramme de classe)

5- Etats des lieux et explication détaillé des fonctionnalités

Quand on lance le script **(Shema+Data)** les table **Users** , **Subscriptions** sont créés:

- A la première exécution du programme on lance la fonction "initSampleData()" qui remplit les collection stations et path dans la base de donnée mongodb

  (on met en commentaire la fonction initSampleData() et on décommente exp7())

1) initialiser la façade

   Qui contient toutes les fonctionnalités du projets

2) S'inscrire à la plate-forme

   On crée un User dans la base donnée Users via Dao;

   Un user qui est ajouté à la bdd mysql;

3) se désinscrire

   Facade.unregister(user,soft= true);

   si soft = true => l'utilisateur sera supprimé de la base de donnée

   si soft = false => l'utilisateur ainsi que ses tickets

   (ce qui supprime en retour toutes les validations de station de toutes les stations associées)

4) Se connecter à son espace personnel

   Le programme va vérifier est ce que l'utilisateur existe d'après

   son nom et son mot de posse et il va l'afficher

5) se déconnecter

   On doit implémenter la partie client pour visualiser cette fonctionnalité

   dans la deuxième partie du projet

6) Souscrire à un abonnement mensuel ou annuel

   Un ticket (non consommable -> "allowed_uses" : -1 et "consommable" = false)

   sera ajouté à la collection tickets dans la base de données mongo

7) Commander un certain nombre de titres (vendables à l'unité ou par 10)

La méthode **buy (Ticket ticket)** qui utilise la couche DAO

pour insérer nouveau ticket dans la base de données

8) Valider un titre de transport sur un terminal dédié

Une validation sera ajoutée pour le ticket validé dans la collection tickets

Si le ticket est consommable :

Si le ticket est à l'unité on le valide la premier fois un message

"ticket valide sera afficher" sinon un message "ticket non valide sera afficher"

S'il est pas à l'unité on peut le valider tant que "allowed_uses" ! = de 0"

Si le ticket n'est pas consommable

On peut valider tant qu'on veut jusqu'à la date de fin de ticket

6- Annexe

MongoDB Docker : https://dev.to/sonyarianto/how-to-spin-mongodb-server-with-docker-and-docker-compose-2lef

MySQL Docker Compose : https://medium.com/@chrischuck35/how-to-create-a-mysql-instance-with-docker-compose-1598f3cc1bee

Schéma Many-Many : http://learnmongodbthehardway.com/schema/schemabasics/