



Rapport

GROUPE : 1

Introduction 🧐

L'objectif

Notre solution à la problématique

Composition de l'équipe

L'architecture 🏗️

MVC

Les entités

UML

Les bases de données

Conception 🧱

Structure du modèle

La facade

Les fabriques

Singleton

Interroger la base Mongo

Interroger la base MySQL

La technique 🛠️

Choix algorithmiques

Relation entre les bases de données

POJO Carte

Les DTO

pom.xml

Docker 🐳

Le docker-compose

Scripts d'initialisation

La vue 💻

Navigation

Un exemple d'utilisation

Introduction 🙄

L'objectif

L'objectif de ce projet était de développer un système de gestion d'abonnements et de trajets pour les utilisateurs de transports en commun.

Avec cette application un client doit pouvoir valider son titre de transport à chaque changement de bus/tram, en sachant qu'un titre doit rester valide pendant une heure.

Il doit aussi pouvoir s'inscrire à la plateforme de manière simple, se désinscrire, se connecter à son espace personnel, se déconnecter, souscrire à un abonnement mensuel ou annuel, commander des titres de transport de 1 à 10 trajets et enfin pouvoir valider sa carte ou son titre de transport lorsqu'il rentre dans les transports en commun.

Notre solution à la problématique

Pour répondre à l'objectif, nous avons créé une application Java avec une base de données MySQL et Mongo.

Composition de l'équipe

- Nicolas Deguyenne (2177545)
- Theau Paty (2172879)
- Nurullah Sakar (2170246)
- Maxime Rousset (2177658)

L'architecture

MVC

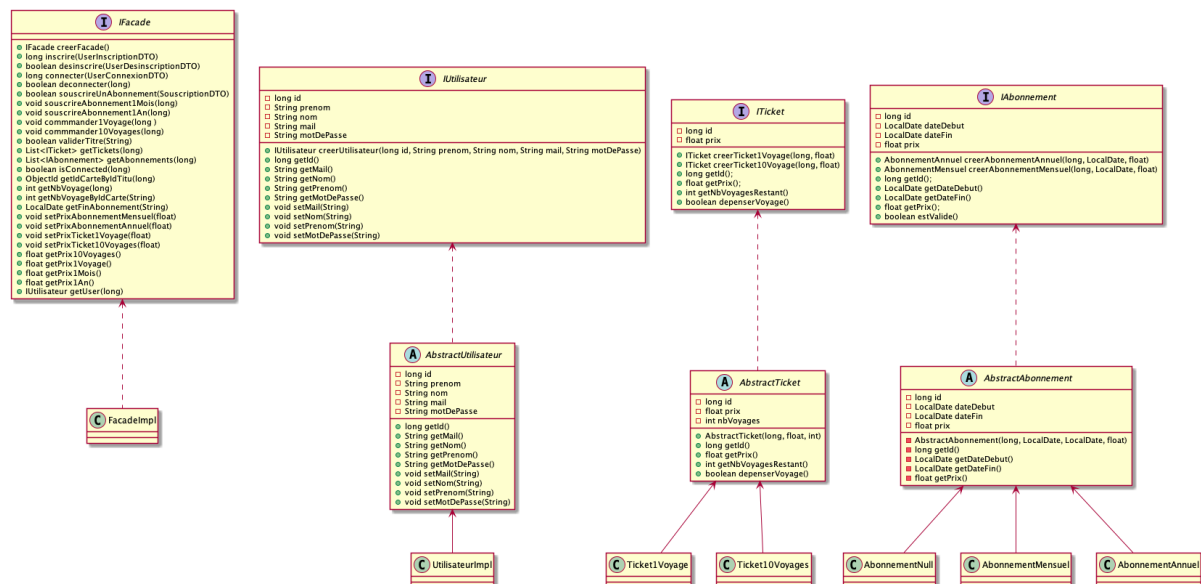
Pour ce projet, nous avons utilisé l'architecture MVC, mais nous nous sommes concentrés en priorité sur le modèle. Nous avons donc préparé le modèle ainsi qu'un contrôleur et une vue.

Les entités

Notre application regroupe 4 types d'entités :

- Les utilisateurs
- Les cartes des utilisateurs
- Les abonnements
- Les tickets

UML



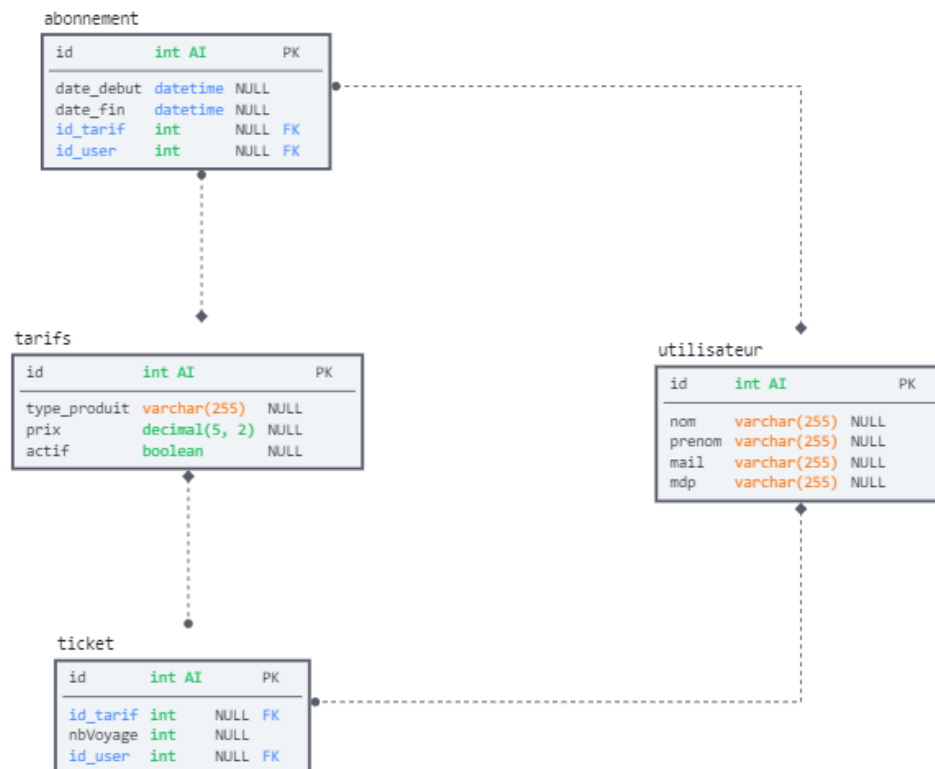
Les bases de données

Pour les bases de données nous avons l'obligation d'utiliser deux bases de données différentes:

- **Une base de données en MySQL**

La base de données en MySQL de notre application est utilisée pour la partie de gestion des utilisateurs et de leurs achats. Elle gère la partie inscription et connexion des utilisateurs et permet de commander des tickets de voyage et des abonnements mensuels ou annuels.

Elle est modélisée de cette façon :



- **Une base de données en MongoDB**

La base de données en MongoDB de notre application est utilisée pour stocker les cartes de transport des utilisateurs. Elle gère toute la partie de validation des abonnements et des titres de transport lorsque l'utilisateur rentre dans un bus/tram.

Pour ce faire, nous avons créé une collection "Cartes" correspondant à la carte de transport de chaque utilisateur. Sur cette carte on peut retrouver les informations nécessaires aux validations des abonnements ou titres de transport, telles que l'id de la carte, l'id du titulaire de la carte, la date de fin de l'abonnement, le nombre de voyages restants grâce aux tickets et enfin, la dernière date de validation d'un ticket pour savoir s'il est encore valide lors d'un changement de ligne par exemple.

On a choisi de ne pas rajouter plus de champs dans la base Mongo pour ne pas avoir de champs inutiles qui surchargerait la base pour rien.

Exemple d'un document dans la collection `cartes` :

```
{
  "_id" : "5ff9cb8c657e7479835ac3f6",
  "id_titulaire" : 1234,
  "nb_voyage" : 2,
  "date_fin_abonnement" : "2022-03-12T00:00:00.000Z",
  "date_derniere_validation" : "2021-01-05T00:00:00.000Z"
}
```

Conception

Structure du modèle

Dans ce projet nous avons fait le choix, de créer une structure similaire pour chaque entité.




Chaque entité du modèle possède :




- un **package** (ex : application.models.produits.abonnement)
- une **interface** (ex : IAbonnement), qui représente l'entité à représenter et qui fait également office de fabrique
- une **classe abstraite** (ex : AbstractAbonnement), qui regroupe les similarités dans les implémentations
- des **implémentations** (ex : AbonnementMensuel, AbonnementAnnuel), qui représente concrètement nos entités.

La facade

La facade nous permet de centraliser tous les cas d'utilisation de notre application.

Principales méthodes

 Nom	 Return	 Description
<u>Inscrire</u>	L'identifiant du nouvel utilisateur	Inscription d'un utilisateur et création de sa carte
<u>Desinscrire</u>	True si l'utilisateur est bien supprimer de la bdd ou false	Désinscrit l'utilisateur et supprime sa carte
<u>Connecter</u>	l'identifiant de l'utilisateur connecté	Connecte un utilisateur et l'ajoute dans la hashmap des utilisateurs connectés

 Nom	 Return	 Description
<u>Deconnecter</u>	True si l'utilisateur est bien déconnecté	Déconnecte l'utilisateur et le retire de la hashmap
<u>SouscrireUnAbonnement</u>	True si l'abonnement a bien été souscrit sinon false	Permet à l'utilisateur de souscrire à un abonnement mensuel ou annuel et ajoute l'abonnement correspondant sur sa carte
<u>SouscrireAbonnement1Mois</u>	Void	Permet à l'utilisateur de souscrire à un abonnement mensuel et ajoute l'abonnement correspondant sur sa carte
<u>SouscrireAbonnement1An</u>	Void	Permet à l'utilisateur de souscrire à un abonnement annuel et ajoute l'abonnement correspondant sur sa carte
<u>Commander1Voyage</u>	Void	Permet à l'utilisateur de commander 1 voyage et ajoute les voyages sur sa carte
<u>Commander10Voyages</u>	Void	Permet à l'utilisateur de commander 1 voyage et ajoute les voyages sur sa carte
<u>ValiderTitre</u>	True si la carte est bien alimentée en voyage sinon false	Valide ou non la carte de l'utilisateur sous réserve d'un nombre suffisant de voyage ou d'un abonnement valide
<u>GetTickets</u>	Retourne la liste des tickets posséder par l'utilisateur	Récupère la liste des tickets posséder par un utilisateur
<u>GetAbonnements</u>	Retourne la liste des abonnements de l'utilisateur	Récupère la liste des abonnements possédés par un utilisateur
<u>IsConnected</u>	True si l'utilisateur est connecté	Check si l'utilisateur est inclus dans la hashmap
<u>GetIdCarteByTitu</u>	Retourne l'id de la carte de l'utilisateur	Récupère l'id de la carte d'un utilisateur via son identifiant

Les fabriques

Chaque élément du modèle est décrit par une interface dans laquelle il y a des **fabriques**.

Par exemple, pour créer un abonnement d'un mois il y a la méthode fabrique :

```
ITicket.creerAbonnementMensuel(...)
```

Singleton

Pour garantir une cohérence dans l'application, nous utilisons le pattern **singleton** pour la façade. Ainsi une seule instance de la façade sera créée.

Interroger la base Mongo

Pour interroger notre base mongo, nous avons regroupé toutes les méthodes qui interagissent avec mongo, dans la classe `MongoDbConnection`. Celles-ci sont ensuite appelées dans la façade.

Interroger la base MySQL

De même, la base MySQL interagit avec une seule classe dédiée :


```
MySQLBddConnection
```

Les configurations de ces bases sont placées dans les ressources du projet :

```
config.ressources.
```


La technique

Choix algorithmiques

<u>Aa</u> Cas	 Traitement
<u>Validation d'une carte</u>	Si la carte a un abonnement valide en cours (antérieur à la date de l'appel) : alors la carte est valide. Sinon : Si la date et l'heure de dernière validation d'un ticket est inférieur d'une heure par rapport à la date de l'appel : alors la carte est valide. Sinon : Si le nombre de voyages restants est supérieur à 0 : alors la carte est valide, on supprime un voyage de la carte et on met à jour la dernière date de validation d'un ticket (si la carte en a déjà une, sinon on la crée) avec la date et l'heure de la nouvelle validation. Sinon : la carte n'est pas valide.
<u>Souscription à 1 mois (resp. an).</u> <u>d'abonnement</u>	Si la personne n'a pas encore d'abonnement ou que son abonnement est périmé : alors on lui ajoute un nouvel abonnement (NB: Dans MySQL, on ajoute une nouvelle ligne dans la table. Dans MongoDB, on met à jour la date de fin si elle est périmée, sinon on ajoute le document) Sinon : on lui ajoute, 1 mois (resp. 1 an), à la date de fin. (NB : Dans MySQL, on ajoute un nouvel abonnement avec comme date de début, la date de fin la plus lointaine des abonnements présents. Dans MongoDB, on met simplement à jour la date de fin) Exemple : Si j'ai déjà un abonnement de 1 mois (04/01/21 → 03/02/21) et que je prends un abonnement de 1 an alors l'abonnement commence le 04/02/21 et prend fin le 03/02/22. Sur la carte on passe date_fin_abonnement de 03/02/21 à 03/02/22.
<u>Gestion des connexions et déconnexions</u>	Pour gérer les connexions et les déconnexions, nous utilisons une Map dans la façade. À sa connexion, l'utilisateur est ajouté dans cette Map. Pour vérifier, s'il est connecté, on regarde dans cette Map avec son id. Un utilisateur ne peut pas faire d'achat si il est pas connecté. Pour déconnecter, on le retire simplement de cette Map

Relation entre les bases de données

Pour chaque utilisateur créé dans l'application il existe une interaction entre MySql et Mongo, dès lors que l'utilisateur est ajouté dans MySql avec ses différents attributs, une carte reliée à son identifiant est automatiquement créée dans Mongo.

La base MySql gère les achats des utilisateurs et leurs informations personnelles. Elle peut servir pour créer l'espace personnel de l'utilisateur avec ses informations, ses factures...

Le base Mongo gère la validation des cartes, elle stocke uniquement les informations essentiels.

POJO Carte

Les documents Mongo sont récupérés par notre application au travers du POJO Carte. Cela facilite la manipulation des documents retournés.

Les DTO

Nous avons décidé de mettre en place des objets de transfert pour utiliser notre façade lorsqu'il y a plus d'un argument. Cela nous permet de contrôler la structure des données en entrée.

pom.xml

Version de java : 11 / Les dépendances pour se connecter aux bases :

```
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <version>8.0.22</version>
</dependency>
<dependency>
  <groupId>org.mongodb</groupId>
  <artifactId>mongodb-driver-sync</artifactId>
  <version>4.1.1</version>
</dependency>
<dependency>
  <groupId>org.mongodb</groupId>
  <artifactId>mongodb-driver-reactivestreams</artifactId>
  <version>1.9.2</version>
</dependency>
```

Docker

Le docker-compose

Le fichier docker-compose monte automatiquement les conteneurs. Il démarre ainsi un conteneur mysql et mongo.

Pour l'exécuter, il faut lancer depuis la racine du projet :

```
cd ../docker/transport-databases/  
docker-compose up
```

Scripts d'initialisation

À la création du conteneur, le docker-compose s'exécute à partir d'un script SQL pour créer la table des utilisateurs dans MySQL et rempli celle-ci par un jeu de test (que l'on a généré sur internet).

Pour compléter ce jeu de test, il faut lancer le `main` la classe `Fixture` du package `programme`. Cela va créer des cartes pour les utilisateurs précédemment inscrits. On leur assignera aléatoirement leurs tickets et leurs abonnements.

La vue

Pour rendre notre application plus concrète, nous avons créé une IHM dans le terminal.

Afin de garder une architecture MVC, nous avons créé un contrôleur. Celui-ci lance l'application à son instantiation.

```
new Controleur()
```

Cette vue regroupe tous les principaux cas d'utilisation que pourrait avoir l'application finale et essaye de se rapprocher le plus possible de la réalité.

Navigation

Pour naviguer dans l'application, vous devez soit choisir une des options présentées ou alors renseigner l'information demandée. Puis appuyer sur <Entrer>.

Deux "espaces" sont présents :

- l'espace client : qui simule le site web ou l'application sur laquelle l'utilisateur gère son compte.
- un espace de validation, qui simule la borne à l'entrée du bus ou du tram. Pour valider sa carte, il faut renseigner son numéro de carte. (disponible dans l'espace client)

Un exemple d'utilisation

On va d'abord s'inscrire

```
##### BIENVENUE #####  
Faites un choix  
-----  
1 - Valider son titre  
2 - Accéder à son espace personnel  
3 - Inscrivez-vous  
4 - Quitter  
-----  
3|
```

On remplit ensuite le formulaire d'inscription

```
##### INSCRIPTION #####  
Nom :  
Musk  
Prenom :  
Elon  
Mail :  
elon.musk@tesla.com  
Mot de passe :  
Mars42  
  > Bienvenue Elon, vous êtes inscrit !  
  > Voici votre nouvelle carte : 6001c669071e2676b2166fd6  
  > Vous pouvez dès maintenant acheter un ticket ou souscrire à un abonnement  
Appuyer sur <Entrer> pour continuer...
```

A ce moment on est enregistré dans la base et on a une carte qui est créée.
On garde ce numéro pour valider sa carte plus tard. Vous pouvez retrouver ce
numéro plus tard dans "espace client > 1 - voir ma carte"

On appuie sur entrée, et on arrive sur l'espace client

```
##### ESPACE CLIENT : MUSK ELON #####  
Faites un choix :  
-----  
1 - Voir ma carte  
2 - Acheter un ticket  
3 - Souscrire un abonnement  
4 - Se désinscrire  
5 - Se déconnecter  
-----
```

Ici on veut acheter un ticket, donc on choisit 2

```
##### ESPACE CLIENT : MUSK ELON #####  
Faites un choix :  
-----  
1 - Voir ma carte  
2 - Acheter un ticket  
3 - Souscrire un abonnement  
4 - Se désinscrire  
5 - Se déconnecter  
-----  
2
```

Pour acheter 10 tickets on choisit 2

```
##### ACHETER DES TICKETS #####  
Faites un choix :  
-----  
1 - Acheter ticket 1 voyage à 3.0€  
2 - Acheter ticket 10 voyages à 20.99€  
3 - Retour  
-----  
2  
  > Vous avez maintenant: 10 voyages  
Appuyer sur <Entrer> pour continuer...  
|
```

On passe alors de 0 à 10 voyages.

Allons maintenant faire un voyage.

Pour cela, il faut retourner la page d'accueil de l'application en se déconnectant 5 et choisir 1.

```
##### ESPACE CLIENT : MUSK ELON #####
Faites un choix :
-----
1 - Voir ma carte
2 - Acheter un ticket
3 - Souscrire un abonnement
4 - Se désinscrire
5 - Se déconnecter
-----
5|
```

```
##### BIENVENUE #####
Faites un choix
-----
1 - Valider son titre
2 - Accéder à son espace personnel
3 - Inscrivez-vous
4 - Quitter
-----
1|
```

On renseigne son numéro de carte (comme à l'entrée du bus quand on présente sa carte)

```
##### VALIDER VOTRE CARTE #####  
Votre numero de carte ( 0 = retour) :  
6001c669071e2676b2166fd6
```

On valide et le titre est valable 1 heure.

```
##### VALIDER VOTRE CARTE #####  
Votre numero de carte ( 0 = retour) :  
6001c669071e2676b2166fd6  
> Votre titre est valide vous pouvez passer [17:59]  
>          Voyages restants : 9  
Appuyer sur <Entrer> pour continuer...
```

A vous d'essayer !