Rapport de Stage

Concepteur Développeur Informatique

Session 2017

Houssam EL ATMANI

Sommaire

Introduction

Dans le cadre de la formation Concepteur Développeur Informatique au sein du centre AFPA, un stage doit être effectuer.

Il a pour but d'expérimenter et prendre conscience du métier, concepteur développeur informatique.
L'intéret du stage permettera au stagiaire de prendre du recul et participer à un projet qui va le permettre de pousser ses connaissances plus loin.

Ce rapport de stage permet par écrit, la description des étapes effectuées.

Presentation

Je m'appelle Houssam, j'ai 30 ans, depuis quelques années, je suis très attiré par le domaine de la programmation.

En effet, les années passées en tant que technicien de maintenance, j'ai vraiment pris conscience du codage, notamment en collaboration avec les collègues informatiens de G7 et en parallèle les tutoriels suivis sur le net, notamment sur le site du zéro, nommé aujourd'hui OpenClassrooms.

J'ai décidé de faire une rupture conventionnelle avec mon ancienne entreprise afin de concrétiser mon ambition de devenir un développeur professionnel.

En faisant cette formation, j'ai pu réussir à cerner ce domaine.
De plus étant trés attiré par la possibilité d'être à mon compte, ce métier me permettera par le fait d'être freelancer, d'apporter des solutions et faire un métier que j'aime.

Environement de stage

J'ai fait mon stage dans une startup EASYCAB.
Le choix d'une startup m'a attiré pour deux principales raisons,

- Savoir si je peux répondre un besoin en tant que seul développeur et éxperimenter le developpement à la manière d'un freelancer.

Le sujet lié au stage est le développement d'un site de réservation de chauffeur privé.
La principale activité de cette entreprise est le domaine de transport privé.
Avoir un site de reservation est vitale pour la suite de sa croissance.

En effet aujourd'hui, face à l'évolution du numérique, et le besoin des clients d'un service de qualité, les applications web et applications de smartphones apportent une réponse à ce besoin.

1.Etude du domaine fonctionnel

Afin d'effectuer une bonne démarche dans le cadre de développement de l'application de réservation de vtc (véhicule de transport avec chauffeur).

Je dois récolter un maximum d'informations requises pour établir un cahier des charges, et un plan de travail, ce qui va me permettre de me baser et répondre au besoin du client

Pour cela j'ai trouvé plusieurs mot, verbes revenant souvent dans le cadre de cette mission:

- Chauffeur, client, adresse, destination, carte bancaire, choix du service, mettre un commentaire, noter un chauffeur, commander une course,
établir une facture, consulter panier, virement bancaire au compte chauffeur, enregistrer un prix, enregistrer une nouvelle voiture.

2.Conception du Projet en UML

2.1 Les Cas d'Utilisations

Aprés avoir récolter les infos nécessaires, j'ai schématiser le fonctionnement de la future application web.

Pour cela, j'ai écrit les cas d'utilisations qui vont permettre une facilité de communication.

Durant la conception, je me suis rendu que le diagramme des cas d'utilisation est devenu complexe à lire.

J'ai choisi de l'épurer en répartisant les Uses Cases en plusieurs packages distincts.

- package client,
- package chauffeur,
- package administrateur,
- package fragments.

Diagramme des cas utilisations "client"

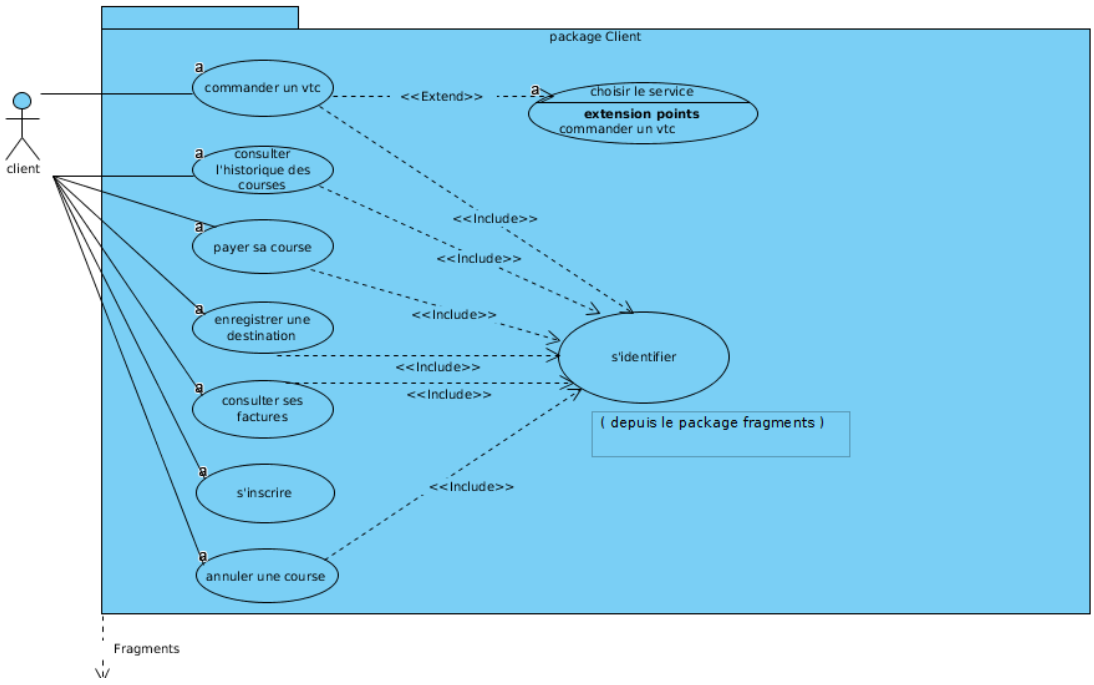


Diagramme des cas utilisations "chauffeur"

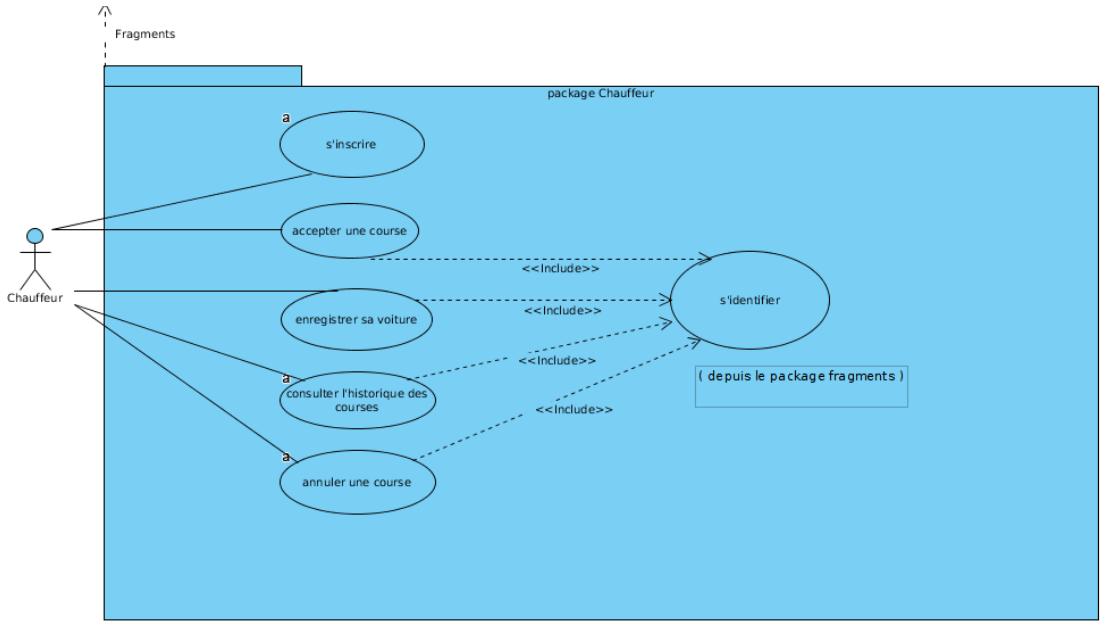


Diagramme des cas utilisations "Administrateur"

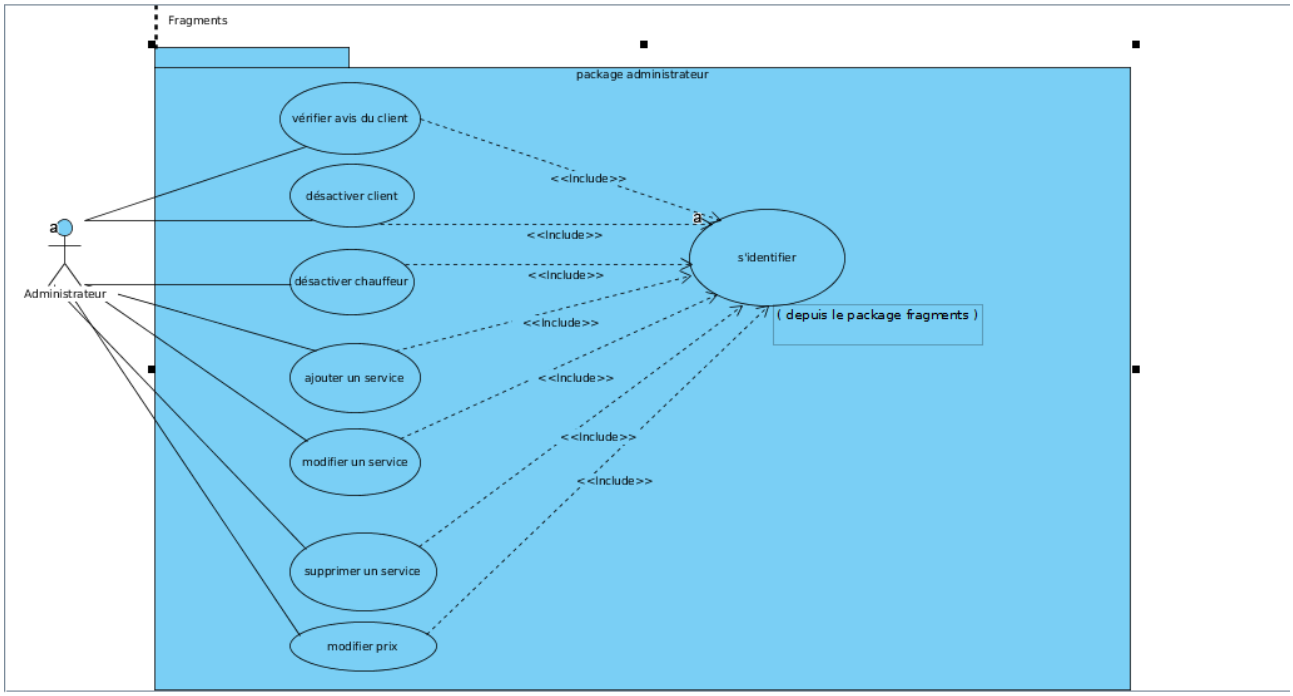
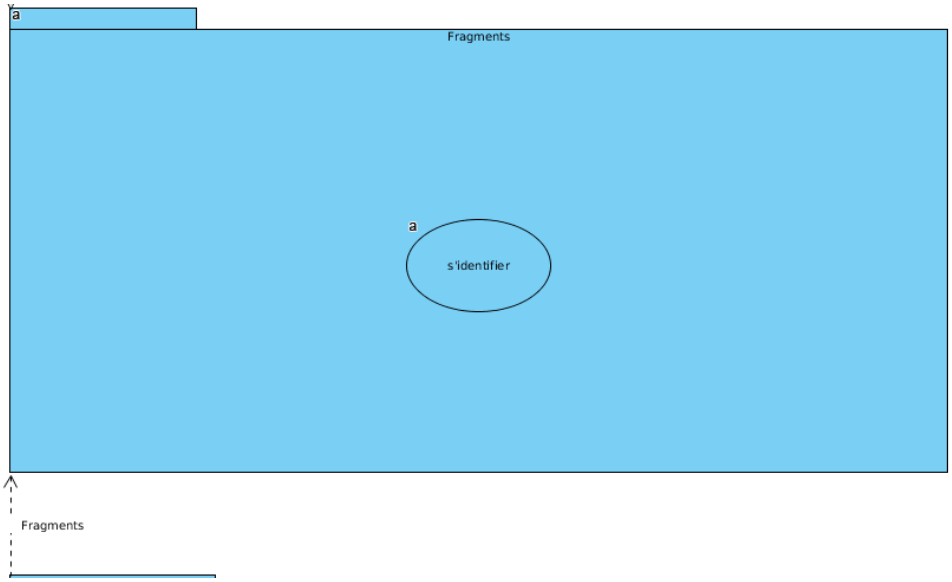


Diagramme des cas utilisations "fragments"



Ces cas d'utilisation permettent de généraliser un fonctionnement,
il est possible de prévoir d'une évolution du système attendu.

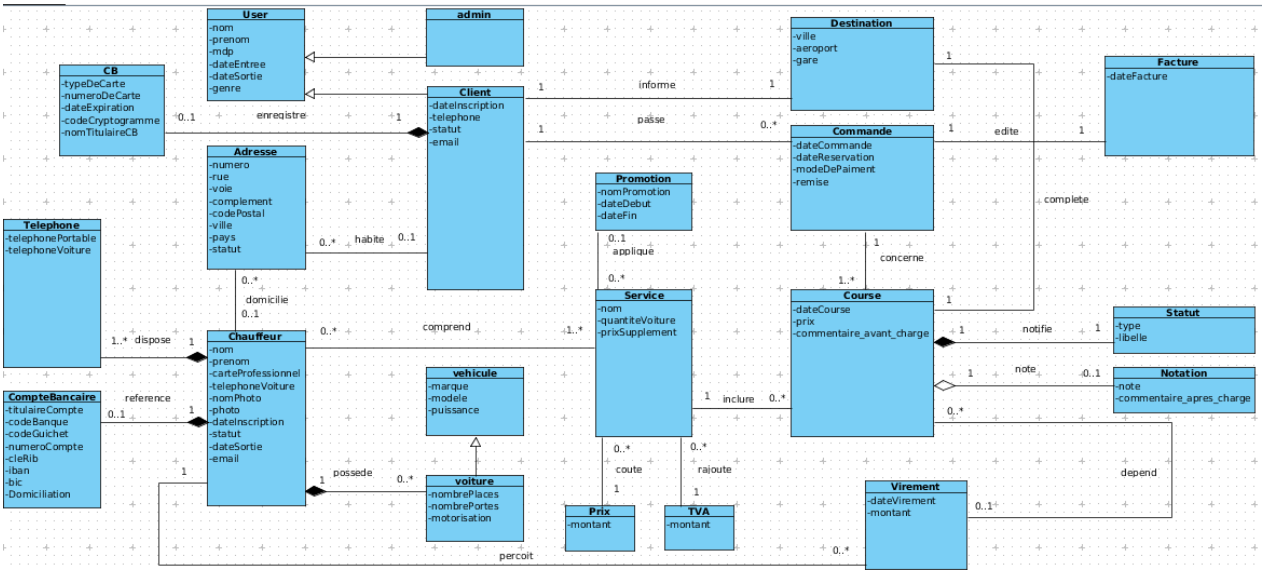
2.2 le Diagramme de classes

Après la récolte, le tri des informations et la mise en place des cas d'utilisations.

Je dois établir un diagramme de classes qui va me permettre de générer
un modèle logique de données et enfin un modèle physique de données.

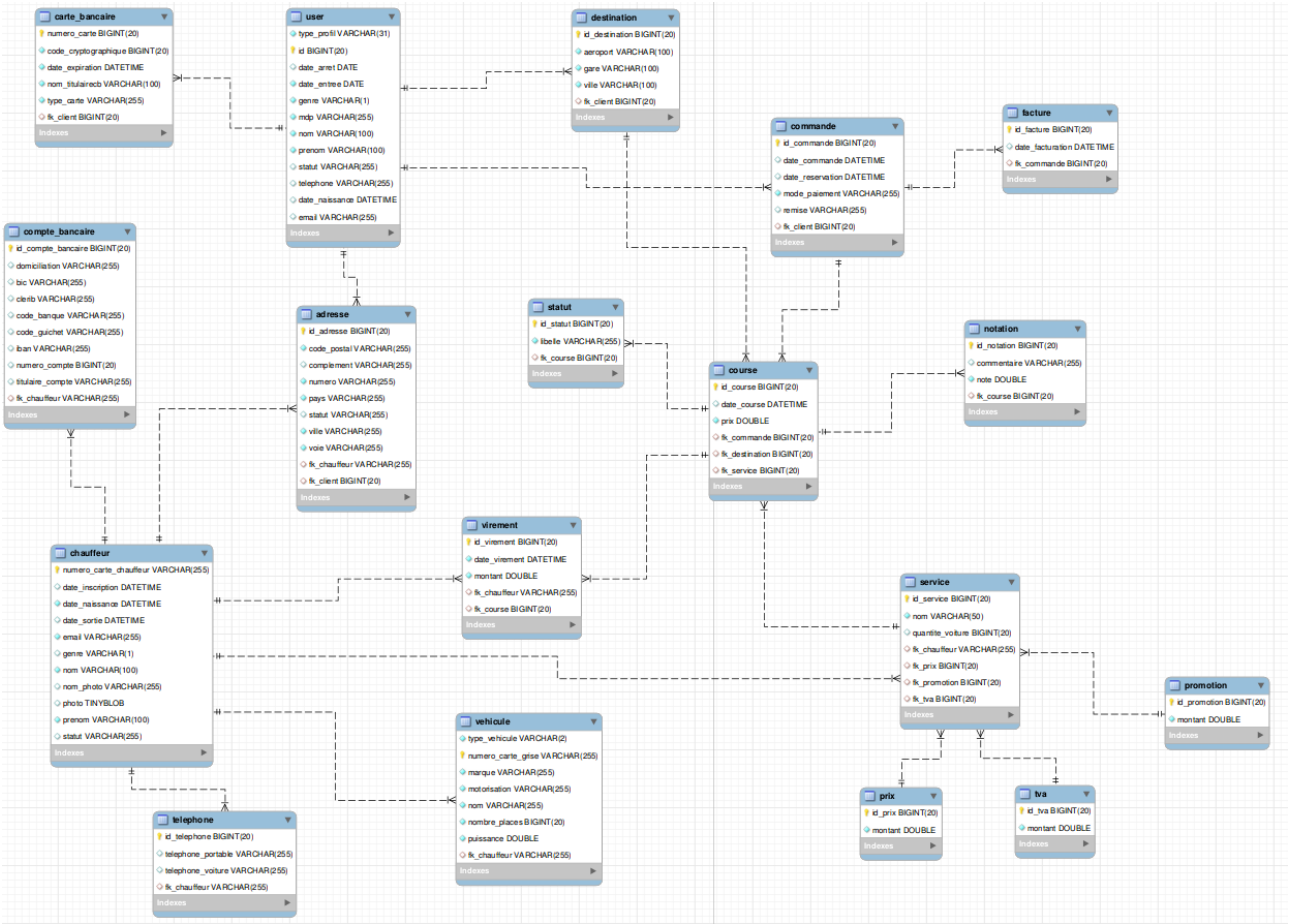
Pour cela, voila le diagramme de classe que j'ai effectué:

Diagramme de classes de l'application web



2.3 Modèle logique de données

Par la suite diagramme de classes, on obtient un modèle logique de données



3.Environement de Développement

3.1 Outils utilisés

Le dévoloppement du système s'est basé sur plusieurs outils et technologies:

- Visual Paradigm : Pour la conception en utilisant UML.
- MySql: Comme système de gestion de base de données relationnelles
- MySql Workbench: pour la visualisation des tables crées.
- NetBeans: un environement de développement intégré.
- Hibernate: pour gérer la persistence des objets dans la base de donnée.
- SpringFramework : un framework qui va permettre l'inversion de contrôle et faciliter le développement, il comprend les modules: Spring Web, Spring Data et Spring Boot.

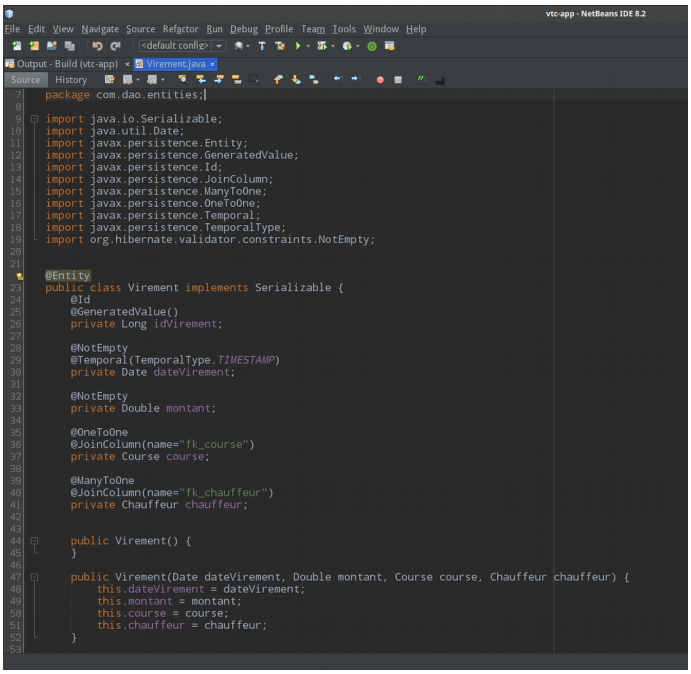
- Thymeleaf: un moteur de template.
- HTML5: pour l'affichage et contenu des pages web.
- CSS3 et Bootstrap pour le design.
- jQuery: librairie javascript pour le comportement.
- Apache Tomcat: comme conteneur web.
- Git: pour la gestion de version de l'application.
- Maven: pour la gestion du projet.

4.Persistence des données

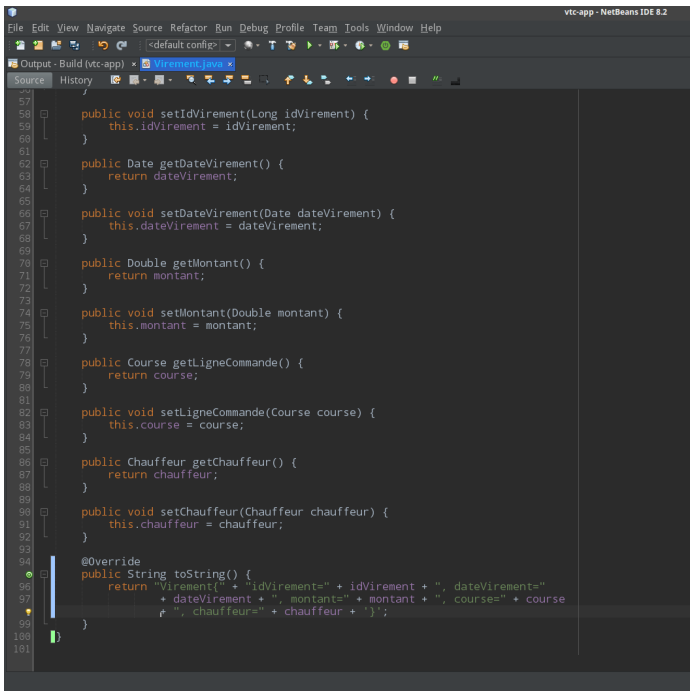
4.1 exemple d'une classe à persister

Pour persister les classes, j'ai utilisé l'api JPA, son implementation Hibernate et Spring Data.

classe Virement qui sera persister en tant que table virement



```java
package com.dao.entities;

import java.io.Serializable;
import java.util.Date;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.OneToOne;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;
import org.hibernate.validator.constraints.NotEmpty;


@Entity
public class Virement implements Serializable {
    @Id
    @GeneratedValue()
    private Long idVirement;

    @NotEmpty
    @Temporal(TemporalType.TIMESTAMP)
    private Date dateVirement;

    @NotEmpty
    private Double montant;

    @OneToOne
    @JoinColumn(name="fk_course")
    private Course course;

    @ManyToOne
    @JoinColumn(name="fk_chauffeur")
    private Chauffeur chauffeur;


    public Virement() {
    }

    public Virement(Date dateVirement, Double montant, Course course, Chauffeur chauffeur) {
        this.dateVirement = dateVirement;
        this.montant = montant;
        this.course = course;
        this.chauffeur = chauffeur;
    }
```

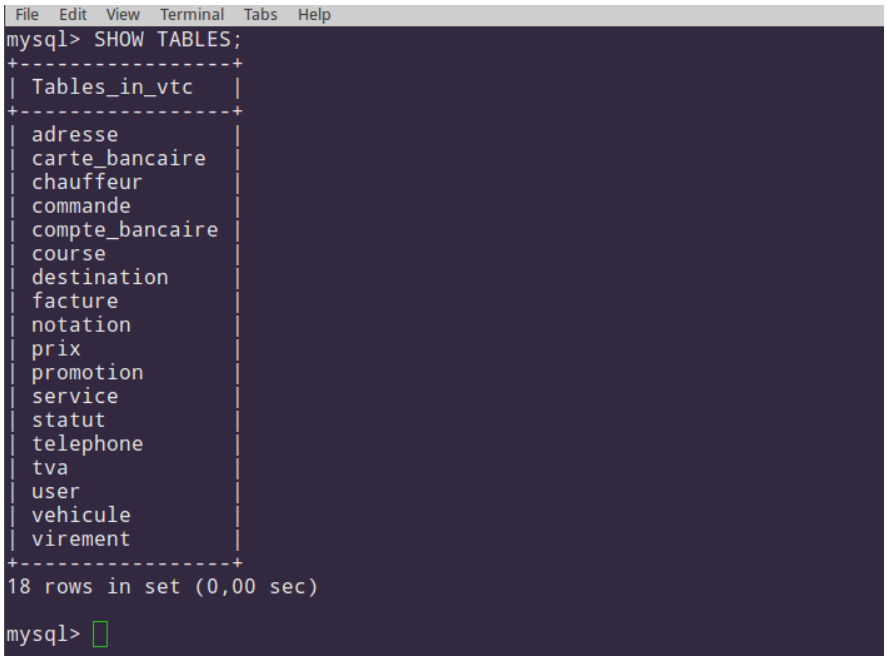


```java
    public void setIdVirement(Long idVirement) {
        this.idVirement = idVirement;
    }

    public Date getDateVirement() {
        return dateVirement;
    }

    public void setDateVirement(Date dateVirement) {
        this.dateVirement = dateVirement;
    }

    public Double getMontant() {
        return montant;
    }

    public void setMontant(Double montant) {
        this.montant = montant;
    }

    public Course getLigneCommande() {
        return course;
    }

    public void setLigneCommande(Course course) {
        this.course = course;
    }

    public Chauffeur getChauffeur() {
        return chauffeur;
    }

    public void setChauffeur(Chauffeur chauffeur) {
        this.chauffeur = chauffeur;
    }

    @Override
    public String toString() {
        return "Virement{" + "idVirement=" + idVirement + ", dateVirement="
                + dateVirement + ", montant=" + montant + ", course=" + course
                + ", chauffeur=" + chauffeur + '}';
    }
}
```

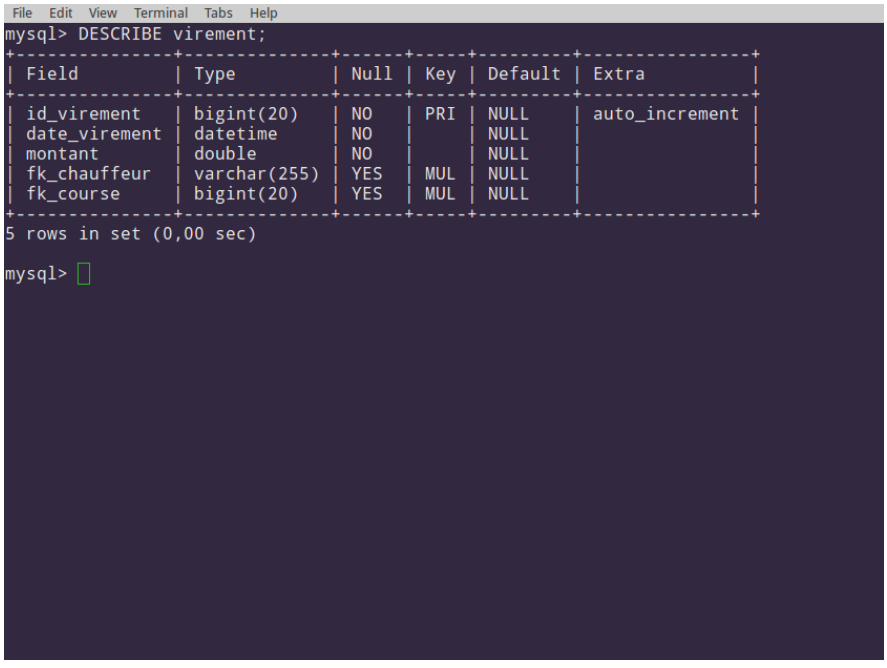
4.2 Extrait de jeu de données

Après l'écriture des classes, il faut vérifier la cohérence des tables dans la base de données.

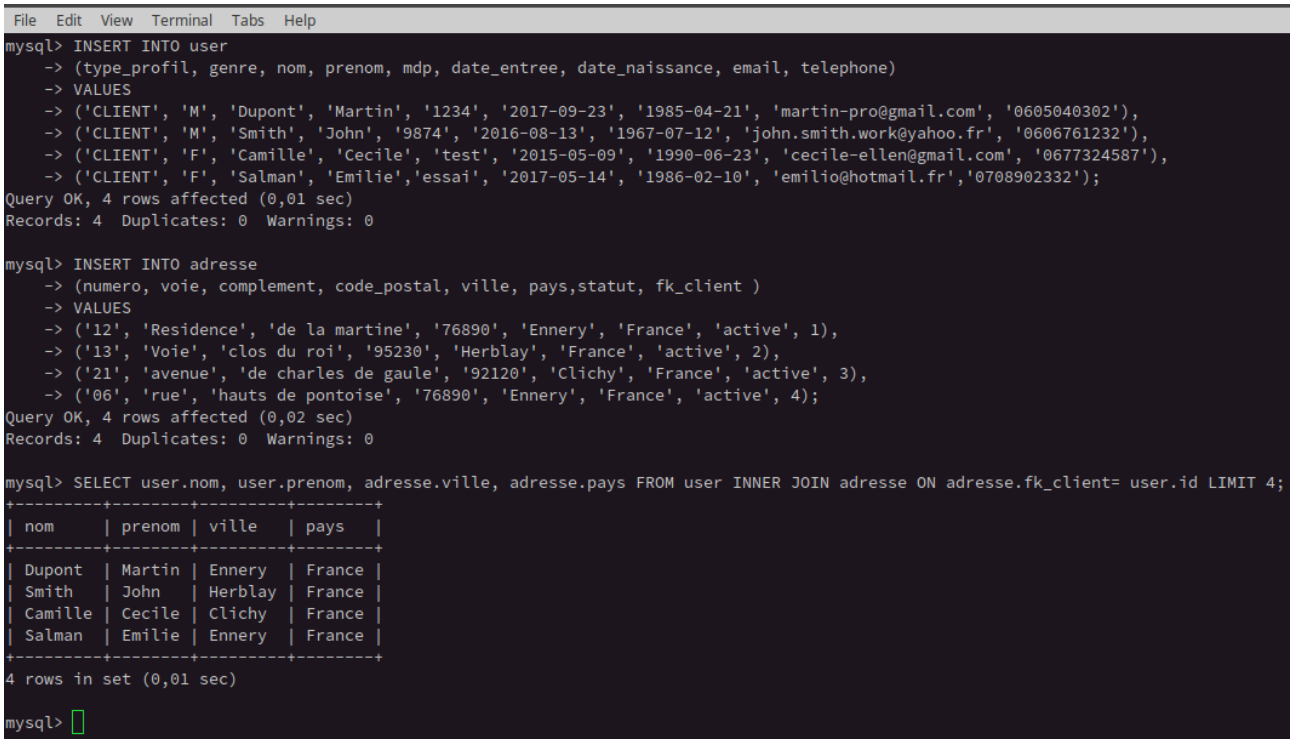
Affichage des tables crées dans la base de données nommée "vtc"



Affichage de la table virement



Insertion puis affichage des données des tables client et adresse



```
File  Edit  View  Terminal  Tabs  Help
mysql> INSERT INTO user
    -> (type_profil, genre, nom, prenom, mdp, date_entree, date_naissance, email, telephone)
    -> VALUES
    -> ('CLIENT', 'M', 'Dupont', 'Martin', '1234', '2017-09-23', '1985-04-21', 'martin-pro@gmail.com', '0605040302'),
    -> ('CLIENT', 'M', 'Smith', 'John', '9874', '2016-08-13', '1967-07-12', 'john.smith.work@yahoo.fr', '0606761232'),
    -> ('CLIENT', 'F', 'Camille', 'Cecile', 'test', '2015-05-09', '1990-06-23', 'cecile-ellen@gmail.com', '0677324587'),
    -> ('CLIENT', 'F', 'Salman', 'Emilie','essai', '2017-05-14', '1986-02-10', 'emilio@hotmail.fr','0708902332');
Query OK, 4 rows affected (0,01 sec)
Records: 4  Duplicates: 0  Warnings: 0

mysql> INSERT INTO adresse
    -> (numero, voie, complement, code_postal, ville, pays,statut, fk_client )
    -> VALUES
    -> ('12', 'Residence', 'de la martine', '76890', 'Ennery', 'France', 'active', 1),
    -> ('13', 'Voie', 'clos du roi', '95230', 'Herblay', 'France', 'active', 2),
    -> ('21', 'avenue', 'de charles de gaule', '92120', 'Clichy', 'France', 'active', 3),
    -> ('06', 'rue', 'hauts de pontoise', '76890', 'Ennery', 'France', 'active', 4);
Query OK, 4 rows affected (0,02 sec)
Records: 4  Duplicates: 0  Warnings: 0

mysql> SELECT user.nom, user.prenom, adresse.ville, adresse.pays FROM user INNER JOIN adresse ON adresse.fk_client= user.id LIMIT 4;
+---------+--------+---------+--------+
| nom     | prenom | ville   | pays   |
+---------+--------+---------+--------+
| Dupont  | Martin | Ennery  | France |
| Smith   | John   | Herblay | France |
| Camille | Cecile | Clichy  | France |
| Salman  | Emilie | Ennery  | France |
+---------+--------+---------+--------+
4 rows in set (0,01 sec)

mysql>
```

Partie Codage

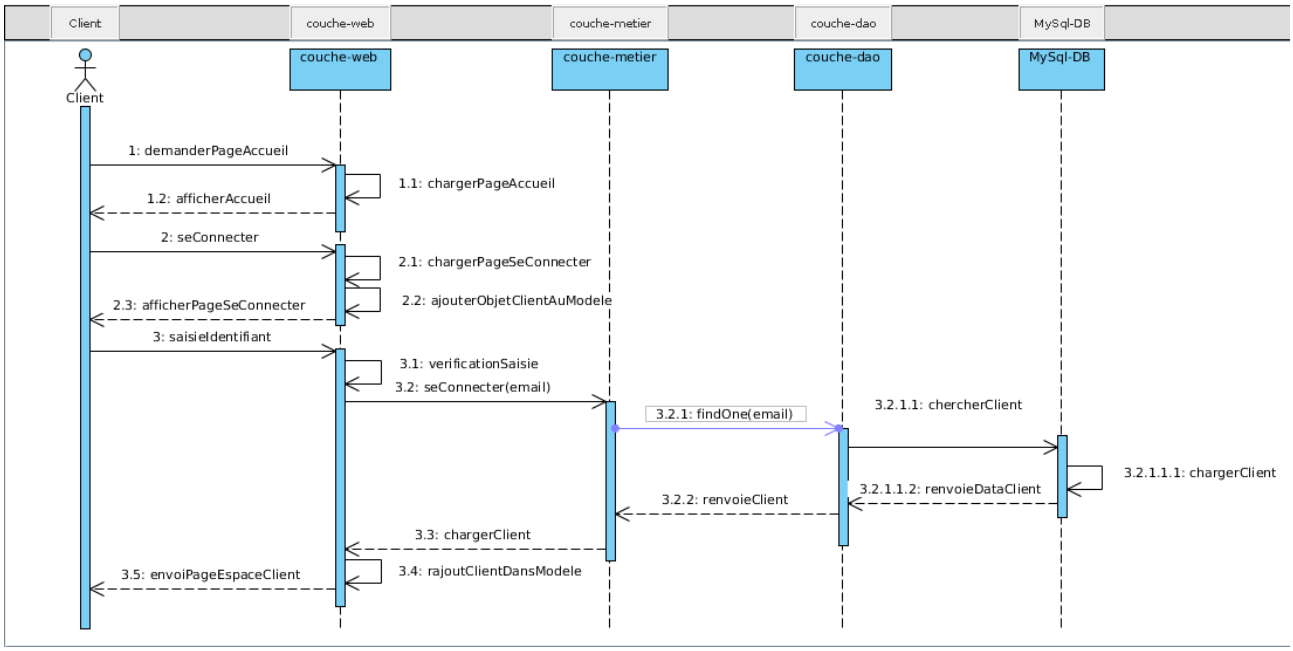
5.Développement de l'application en trois couches

Afin de coder une bonne application, qui facilitera la maintenance et les mises à jours. Une architecture en plusieurs couches doit être envisagée.

J'ai fractionné le développement en trois couches distinctes :

- la couche DAO
- la couche METIER
- la couche WEB

Diagramme de séquence schématisant la connection d'un client membre



5.1 La Couche Dao

J'ai créé un package que j'ai nommé Dao.
Cette couche va permettre la communication avec la base de données.

Dans ce package, on trouve deux autres packages:

- entities
- repository

Le package entities contient les classes persistables.
La package repository contient les interfaces qui héritent de l'interface jPaRepository de spring data.
Cela va faciliter l'échange avec la base.

Pour un client, par exemple, je vais faire de la manière suivante:

une interface nommée "IclientRepository" qui hérite de JpaRepository<T, ID>
Pour T il faut spécifier l'objet concerné et "ID" le type choisi.
l'interface va héritée de toutes les méthodes de JpaRepository, mais on peut ajouter les méthodes spécifiques. Par exemple dans le cadre de l'application je vais avoir besoin la recherche par email.
Pour cela, j'ai rajoutée une méthode personnalisée

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

package com.dao.repository;

import com.dao.entities.Client;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;



public interface IClientRepository extends JpaRepository<Client, Long> {

    @Query("select c from Client c WHERE c.email = :x")
    public Client findByEmail(@Param("x") String email);
}
```

5.2 La Couche Metier

Cette couche va contenir les classes correspondant aux fonctionalitées métier.
Elle permet de séparer la couche web de la couche dao.
J'ai utilisé des interfaces, contentant les signatures des méthodes métiers.
Ensuite j'ai créée les classes correspondantes qui implementeront les interfaces.
Cela va permettre de faciliter les mises à jours et changement lors d'une évolution du système.

Interface IClientMetier

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

package com.metier;

import com.dao.entities.CarteBancaire;
import com.dao.entities.Client;



public interface IClientMetier {
    public Client inscrireClient(Client client);
    public Client connecterClient(String email, String mdp);
    public void enregisterCB(Client client, CarteBancaire carteBancaire);

    /*
    public Course commander();
    public Course annulerCourse();
    public Course noterChauffeur(Course course);
    public List consulterListeCourses();
    public List consulterListeCommandes
    */
}
```

Classe ClientMetierImpl implémentant l'interface IClientMetier

```java
@Service
@Transactional
public class ClientMetierImpl implements IClientMetier {
    @Autowired
    IClientRepository clientRepository;
    @Autowired
    ICarteBancaireRepository carteBancaireRepository;

    @Override
    public Client inscrireClient(Client client) {
        if(client != null) {
            clientRepository.save(client);
            return client;
        }
        else {
           throw new RuntimeException("client invalide ");
        }
    }

    @Override
    public Client connecterClient(String email, String mdp) {
        if(email != null || !email.isEmpty()) {
            Client c = clientRepository.findByEmail(email);

            if(c.getMdp().equals(mdp)){
                return c;
            }
            else {
                throw new RuntimeException("mot de passe invalide");
            }
        }
        else {
            throw new RuntimeException("client introuvable ");
        }
    }

    @Override
    public void enregisterCB(Client client, CarteBancaire carteBancaire) {
        if(client != null) {
            if(carteBancaire != null) {
                carteBancaire.setClient(client);
                carteBancaireRepository.save(carteBancaire);
            }
        }
    }
```

5.3 La Couche Web

Avec les deux couches précedemment citées, il devient plus aisé d'isoler le code
Du coup cette couche aura pour seul objectif le traitement liée à 'lIHM.
De plus avec le design pattern MVC, il est encore plus agréable de se focaliser.

J'ai utilisé un moteur de template nommé Thymeleaf, avec celui-ci, je peux utiliser uniquement des pages Html et la combinaison avec SpringWeb rend le developpement trés riche en possibilitées.

Avec la configuration de SpringBoot, on trouve dans le defaut package,
un package resources, qui contient deux autres packages,
qui sont "static" et "templates".
Static: contiendera les ressources de type image et css.
templates: contiendera les pages html et les templates pour customiser les pages html.

Extrait du code connexion client:

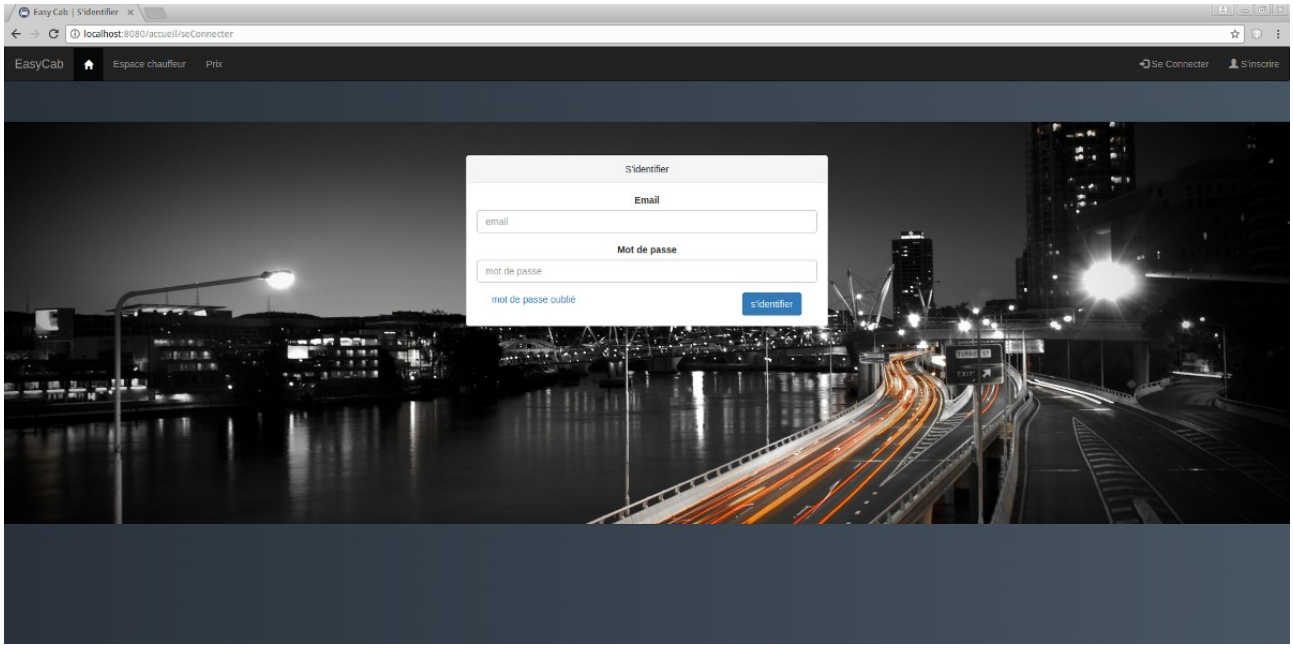
ClientController

```
@Controller
@RequestMapping("/accueil")
public class ClientController {
    @Autowired
    private Internaute internaute;

    @Autowired
    private ClientMetierImpl clientMetier;

    @RequestMapping("")
    public String index() {
        return "accueil";
    }

    @RequestMapping(value="/seConnecter", method=RequestMethod.GET)
    public String seConnecter(Model model) {
        model.addAttribute("internaute", new Internaute());
        return "connexionClient";
    }

    @RequestMapping(value="/espaceClient")
    public String espaceClient(Model model, @Valid Internaute internaute,
            BindingResult bindingResult) {
        if(bindingResult.hasErrors()) {
            return "connexionClient";
        }

        Client c = clientMetier.connecterClient(internaute.getEmail(), internaute.getMdp());

        if(c != null) {
            model.addAttribute("client", c);
            return "espaceClient";
        }
        else {
            return "connexionClient";
        }

    }

    @RequestMapping(value="/sinscrire", method=RequestMethod.GET)
    public String sinscrire(Model model) {
        model.addAttribute("client", new Client());
        return "formInscription";
    }
```

page html représentant la connexion client

```
<html
    xmlns="http://www.w3.org/1999/xhtml"
    xmlns:th="http://www.thymeleaf.org"
    xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"

    layout:decorator="template1">

    <head>
        <title>Easy Cab | S'identifier</title>
        <meta charset="UTF-8" />
        <link rel="stylesheet" type="text/css" th:href="@{/css/myStyle.css}" />

    </head>
    <body>
        <div layout:fragment="content" >
           <div id="scene_connexion">
            <div class="container">

                    <div class="col-md-6 col-sm-6 col-xs-12 col-md-offset-3 col-sm-offset-3 spacer">
                        <div class="panel panel-default ">
                            <div class="panel-heading">S'identifier</div>
                            <div class="panel-body">
                                <form th:action="@{espaceClient}" method="post" th:object="${internaute}">

                                    <div class="form-group">
                                        <label class="control-label">Email</label>
                                        <input  type="email" placeholder="email" th:field="*{email}" class="form-control" />
                                        <span class="text-danger" th:errors="*{email}"></span>
                                    </div>


                                    <div class="form-group">
                                        <label class="control-label">Mot de passe</label>
                                        <input  type="password" placeholder="mot de passe" th:field="*{mdp}" class="form-control" />
                                        <span class="text-danger" th:errors="*{mdp}"></span>
                                    </div>
                                    <div class="col-md-4">
                                        <a href="#">mot de passe oublié</a>
                                    </div>
                                    <div class="col-md-2 col-md-push-5">
                                        <button type="submit" class="btn btn-primary">s'identifier</button>
                                    </div>
                                </form>
                            </div>
                        </div>
                    </div>
                </div>
```

visual de la page html sur le navigateur



Conclusion

L'application est toujours en développement, la fin de stage ne signifie pas l'abandon du projet, au contraire, j'ai l'intention de le finaliser,
de plus je vais compléter ce projet par Angular4 pour le côté client.

Ma conclusion pour ce projet est qu'il faut savoir gérer son temps, et bien répartir ses tâches et mettre une stratégie d'approche pour avancer sereinement.