Tristan Leduc
Victor Perez
Hugo Robert
Roxanne Xu

Rapport de projet
BDCO

Table des matières

I. Introduction

Voici le rapport de notre projet de BDCO. Son but fut de mettre au point une application se basant sur le principe similaire que l'application blablacar : des conducteurs proposent à des utilisateurs de les emmener à leur itinéraire à bord de leur véhicule, si ceux-ci doivent aller dans la même direction.

II. Mode d'emploi

Pour compiler le projet, il faudra lancer Maven avec les commandes suivantes :

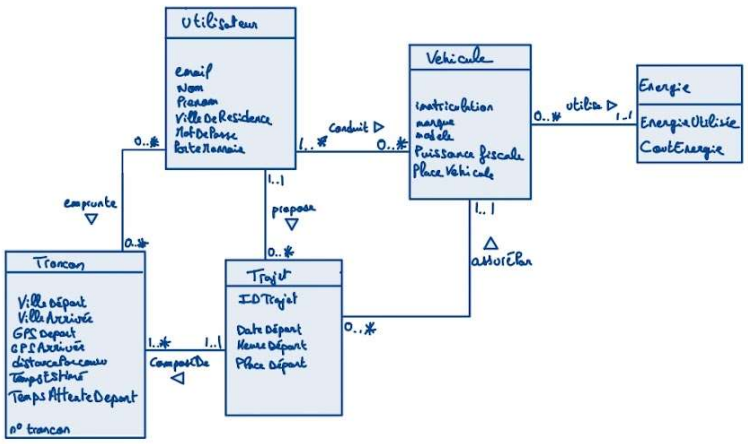
- mvn install pour installer les packages additionnels renseignés dans maven
- mvn compile pour compiler les fichiers du projet
- mvn exec:exec pour exécuter le projet

III. Description du projet

III.1)Analyse de la base de donnée

a) Les schémas entités associations

Diagramme entité association :



Contraintes de valeur

energie utilisée = {essence, diesel, electrique, hybride}

placeDépart > 0

placeDépart < PlacesVehicule

PlacesVéhicule > 0

Puissance fiscale > 0

distanceTroncon > 0

dureeTroncon > 0

TempsAttenteDépart ≥ 0

Temps Estimé > 0

Distance parcourue > 0

PorteMonnaie ≥ 0

n° tronçon ∈ N*

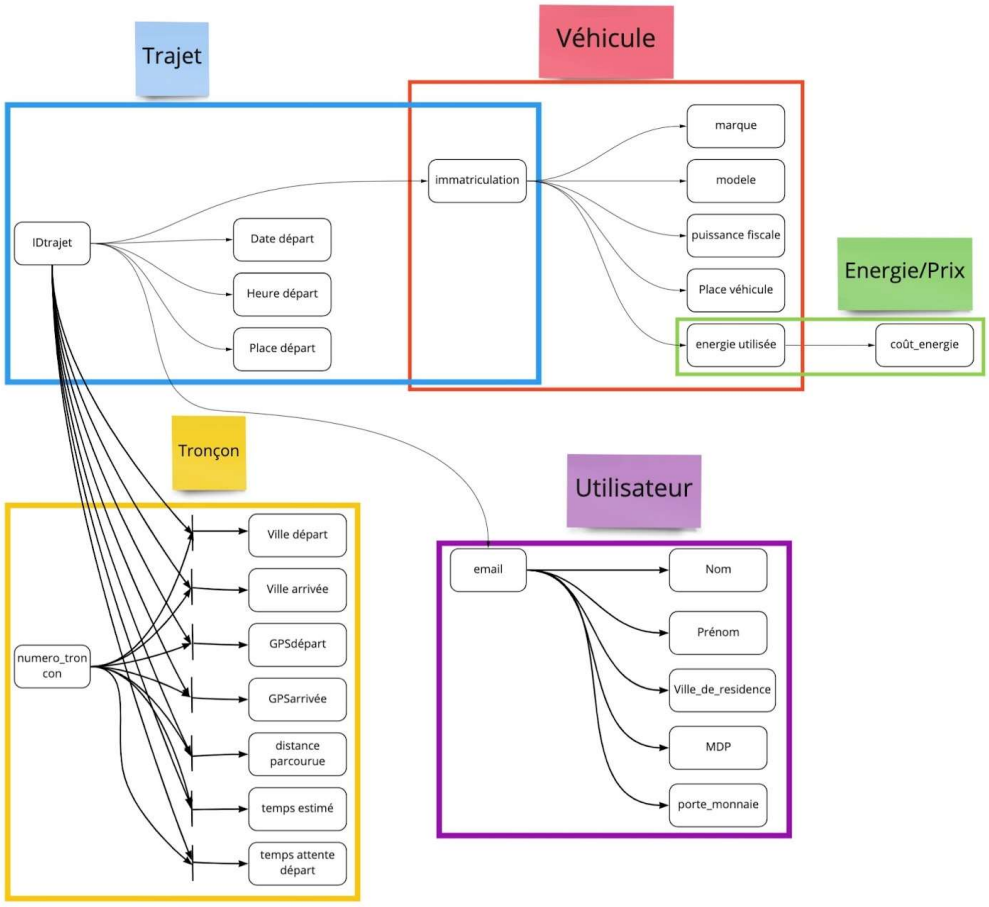
CoutEnergie > 0

Contraintes textuelles :

- Des passagers peuvent monter dans le véhicule à chaque étape
- 1 changement de véhicule maximum (si l'attente prévue à l'étape n'excède pas 1 heure,que la distance euclidienne entre les points GPS d'arrivée et de départ de l'étape ne dépasse pas 0,01)
- Pour proposer un trajet, un conducteur doit être associé à au moins une voiture

Le schéma entité association retranscrit les relations entre les entités du modèle. Il nous a fallu définir des contraintes de valeur pour certaines valeurs des tables. Nous avons aussi pu prendre conscience des attributs utiles, et il nous a fallu éviter les redondances dans les tables.

b) Du schéma E/S vers un modèle relationnel



c) La formalisation des différentes tables

La base de données de notre projet se divise en sept tables :

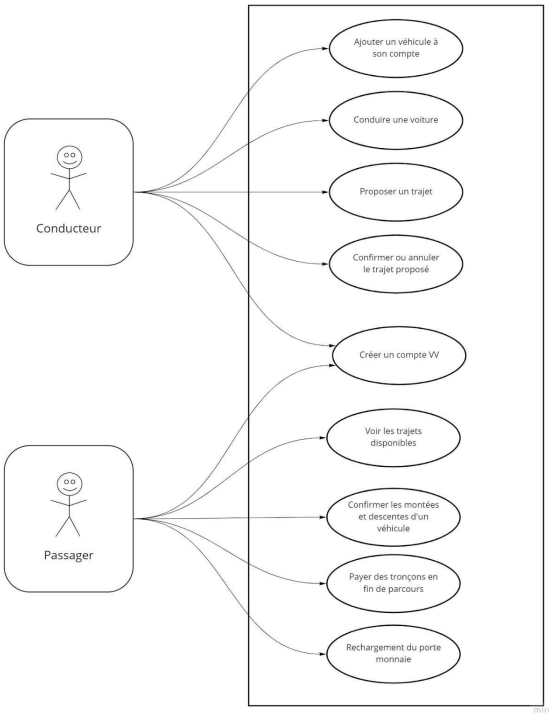
- la table **UTILISATEUR** : recense les utilisateurs de l'application comme son nom l'indique. Un utilisateur est identifié de manière unique par son email. Il est également identifié par son nom, prénom, sa ville de résidence, son mot de passe, et son porte monnaie.
- la table **TRAJET** : représente les trajets planifiés par les conducteurs. Un trajet est identifié par
    - sa clé primaire IDTRAJET
    - son lieu de départ
    - l'immatriculation du véhicule assigné au parcours (la clé étrangère qui identifie un véhicule de manière unique)
    - la date d'arrivée et de départ
    - et les indices de validation effectives du début et de la fin du trajet par les utilisateurs
- La table **TRONCON** : étroitement liée à la table trajet, elle répertorie les différentes étapes d'un trajet. Chaque tronçon est identifié par son numéro de séquence dans un trajet, et le couple (numéro de troncon, idTrajet) forme la clé

primaire de cette table. Les coordonnées GPS de départ et d'arrivée sont également dans cette table, ainsi que la distance parcourue, le temps estimé, le temps d'attente au départ du tronçon. Pour compléter le tout, des indices sont présents afin que l'utilisateur puisse confirmer sa montée dans le véhicule au début du tronçon (MONTEE_VALIDEE) et sa descente une fois à destination (DESCENTE_VALIDEE).

- La table **VEHICULE** : répertorie le parc de voitures enregistrées sur l'application. Les véhicules sont identifiés par leur immatriculation, leur modèle, leur puissance fiscale, la place dans le véhicule ainsi que l'énergie utilisée.
- La table **ENERGIE_PRIX** : associe le coût des énergies à leur type.
- La table **EMPRUNTE** : comporte les utilisateurs prenant part à un trajet. Il est ainsi nécessaire d'associer un email à un idTrajet et à un tronçon particulier.

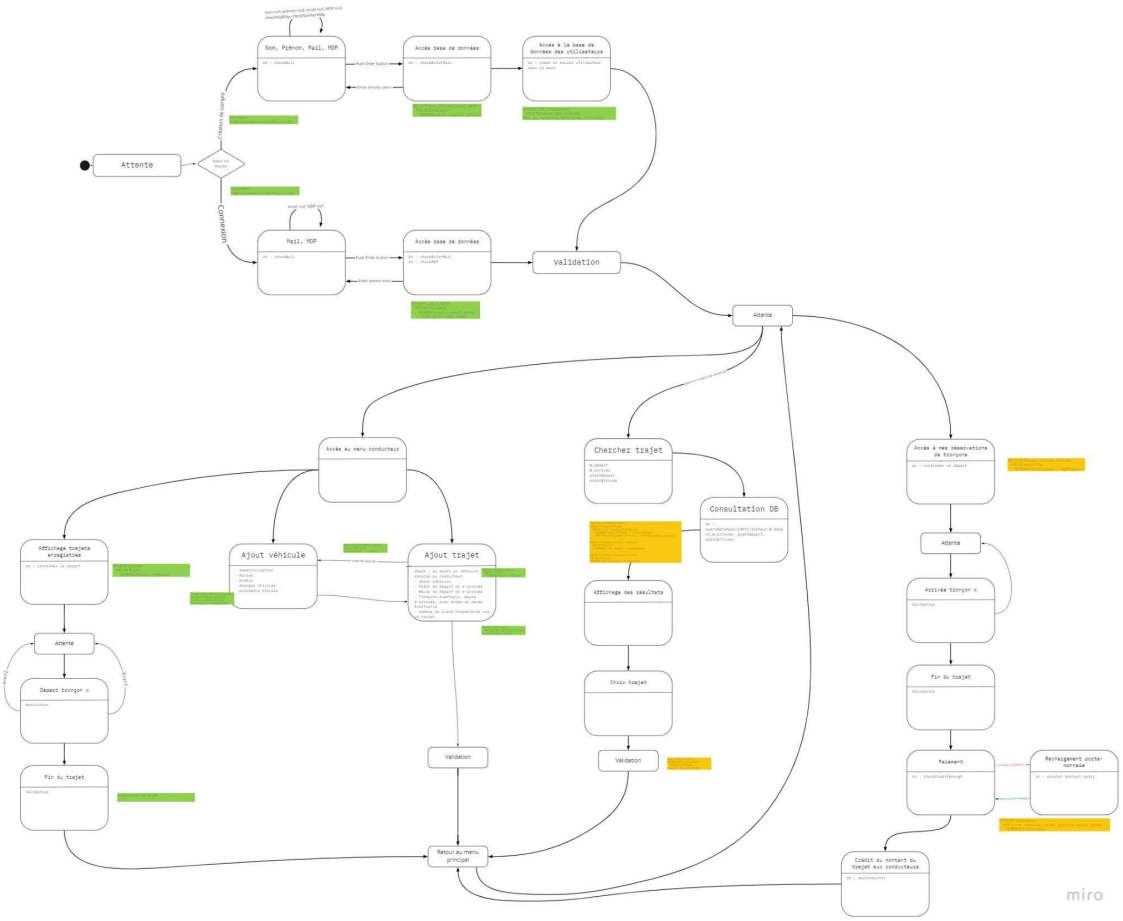
III.1)Analyse fonctionnelle et architecture de l'application

a) Les cas d'utilisation



Notre conception se base sur deux types d'utilisateurs, le Conducteur et le Passager. Ces utilisateurs auront des rôles différents. Cependant, par souci de simplicité, nous allons considérer qu'un utilisateur peut être à la fois Conducteur et Passager.

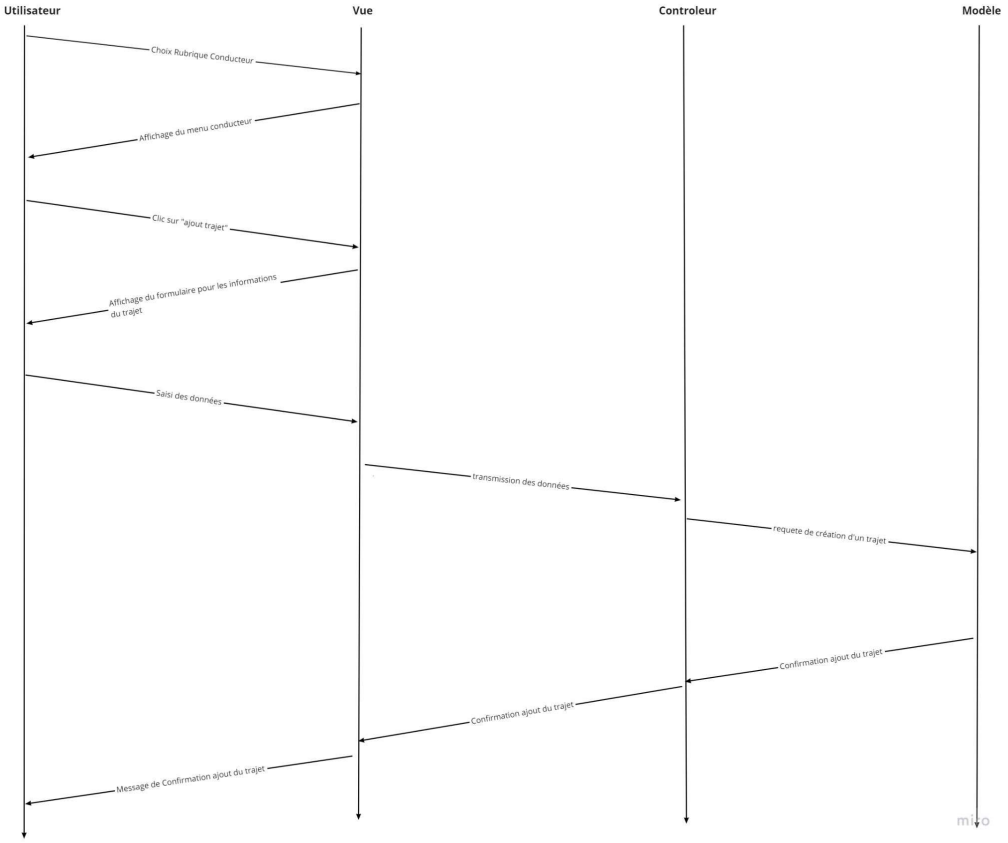
b) Diagramme d'état-transitions



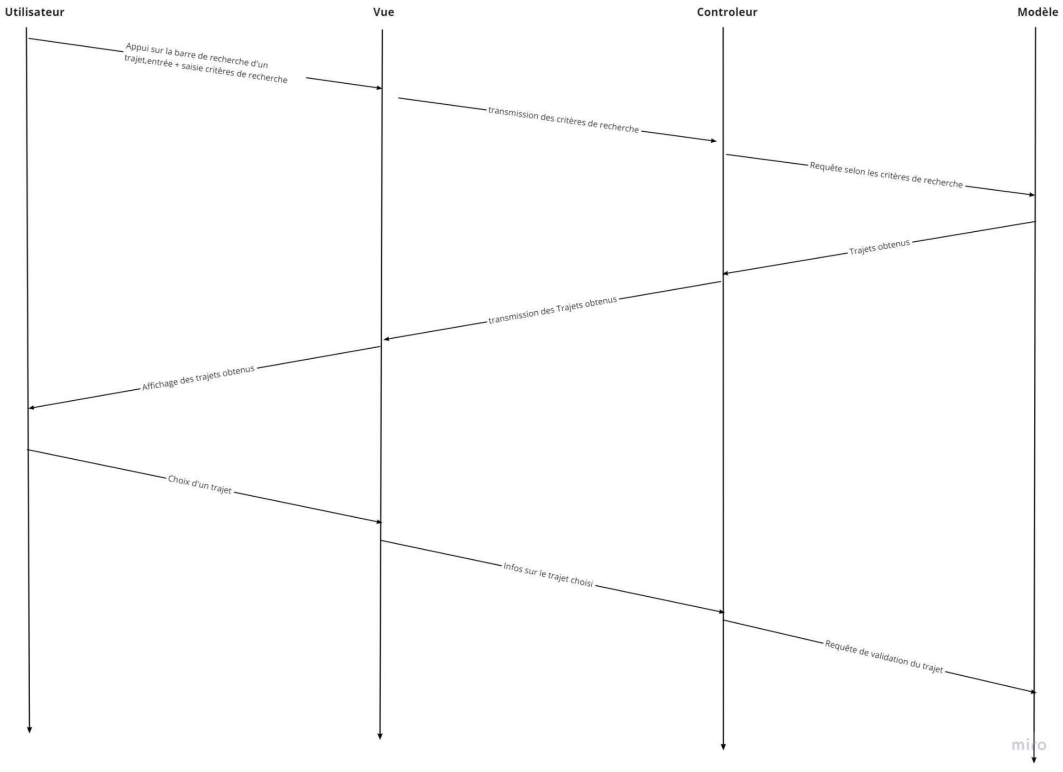
Ce diagramme présente les différentes étapes menant à la connexion d'un utilisateur, à l'ajout d'un véhicule et d'un trajet, à la recherche d'un trajet et aux tronçons déjà réservés par un passager.

c) Quelques diagrammes de séquences

Ces diagrammes présentent le cheminement de l'ajout d'un trajet ainsi que celui de la recherche d'un trajet à travers l'architecture MVC.



Recherche et sélection d'un trajet par l'utilisateur



III. Conception et utilisation de Verbiage Voiture

a) Compilation du projet

Pour compiler le projet, il faudra lancer Maven avec les commandes suivantes :

- mvn install pour installer les packages additionnels renseignés dans maven
- mvn compile pour compiler les fichiers du projet
- mvn exec:exec pour exécuter le projet

IV. Limites de l'Application Verbiage Voiture

Bien que les bases fondamentales du projet VerbiageVoiture aient été posées au cours du développement de l'application, il subsiste de nombreux manques. Il n'est notamment pas encore possible de payer un tronçon/trajet ou encore de …

V. Bilan du projet

Grâce à sa grande pluralité, ce projet nous a permis de développer une application en full stack. Nous avons rencontré quelques difficultés pour trouver comment mettre en place efficacement l'interface graphique.