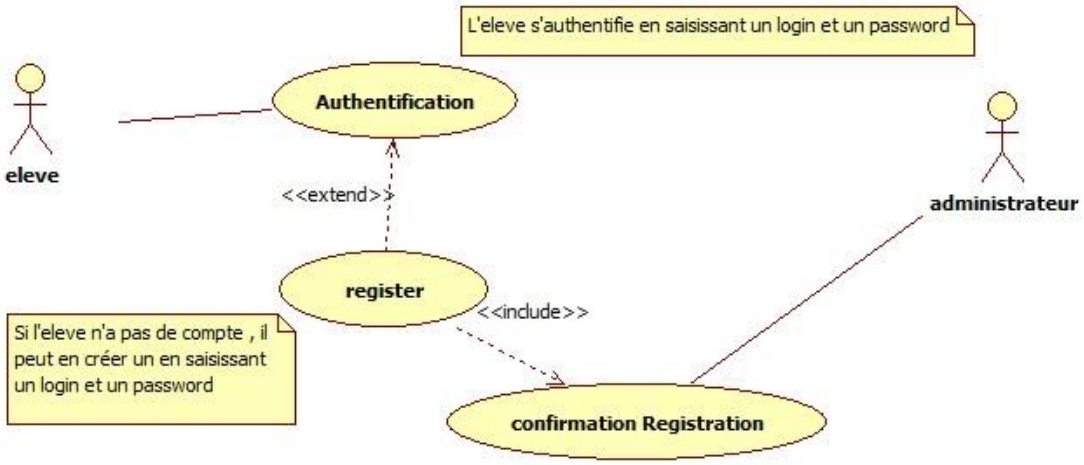
Application Mobile de Covoiturage

Objectif du document

Compte tenu des prix particulièrement élevés des carburants aujourd'hui, Le GRETA souhaite créer sa propre application de covoiturage pour ses élèves.
On vous demande dans un premier temps de créer une API REST.
Dans un second temps une application Front permettra aux élèves d' utiliser ce service.

Cette analyse faite à partir de la méthode UML va permettre d'identifier les objets relatifs aux différents cas d'utilisation, ainsi que le comportement statique et dynamique de chaque objet.

Cas d'utilisation Authentification



Description des interactions entre objets

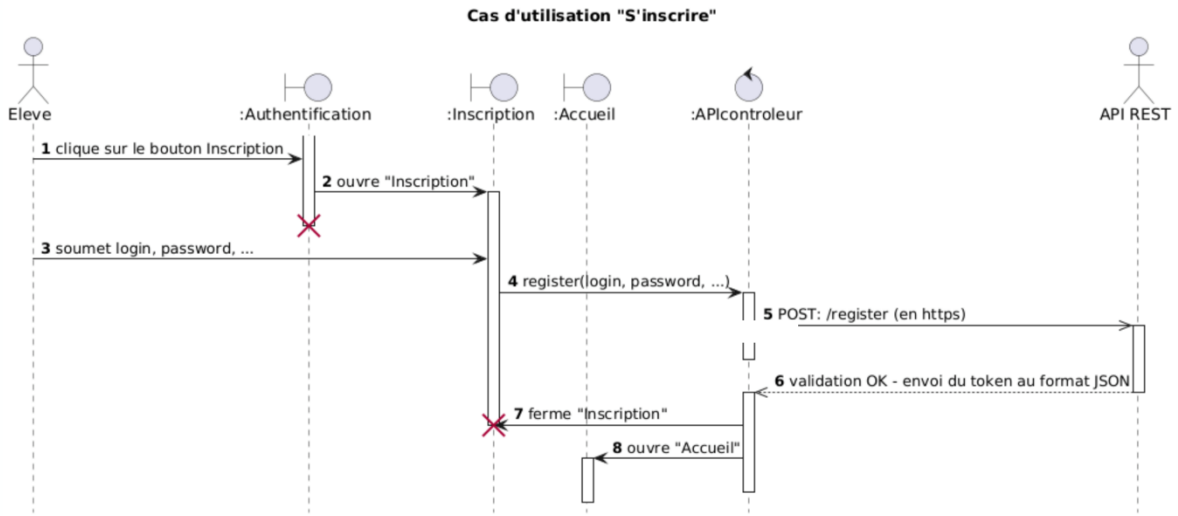
L'utilisateur saisit son email et son password sur l'application mobile.
Ses données sont envoyées sur une API REST via ma méthode POST..
Si le mot de passe et le login sont corrects, l'API REST renvoie **un token de session** au format JSON.
Dans le cas contraire, l'application renvoie au format JSON un **message d'erreur**.

L'administrateur n'a pas besoin de confirmer toutes les inscriptions.Mais il a la possiblité d'activer ou de désactiver des comptes utilisateurs.

Vue Authentification

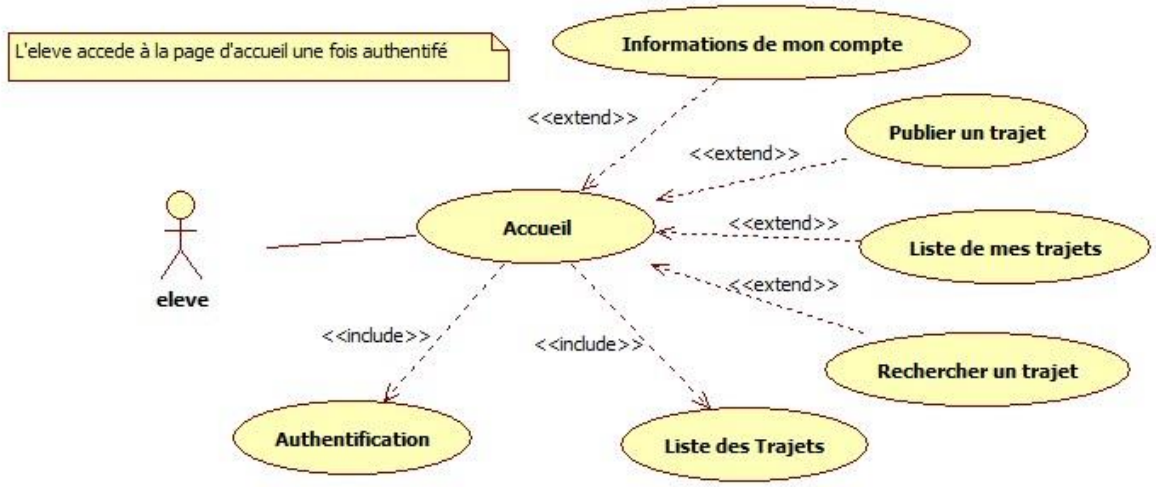
Description des interactions entre objets (Use case Register)

L'utilisateur doit pouvoir s'enregistrer en saisissant un login(email) et un mot de passe.



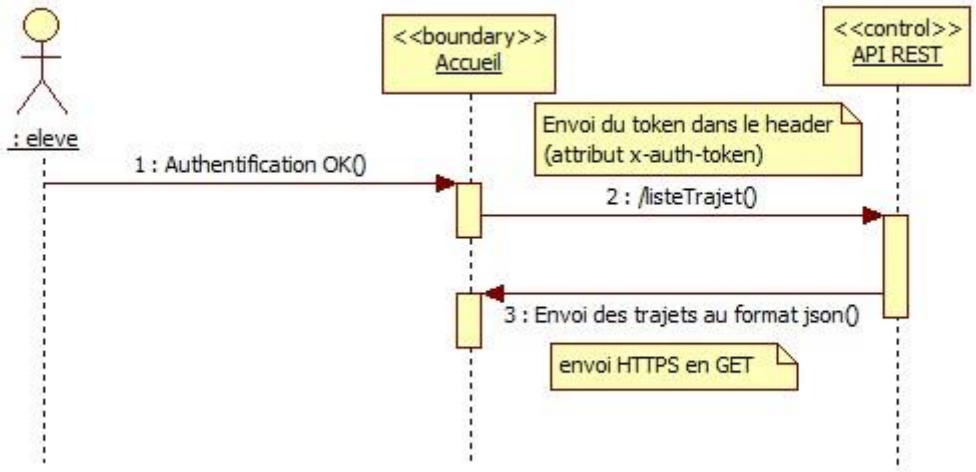
Le formulaire d'inscription doit lui proposer de confirmer son mot de passe Si les mots de passe ne sont pas identiques, l'API REST doit renvoyer un message d'erreur .Il faut utiliser la méthode POST pour envoyer les données à l'API.

Cas d'utilisation Accueil

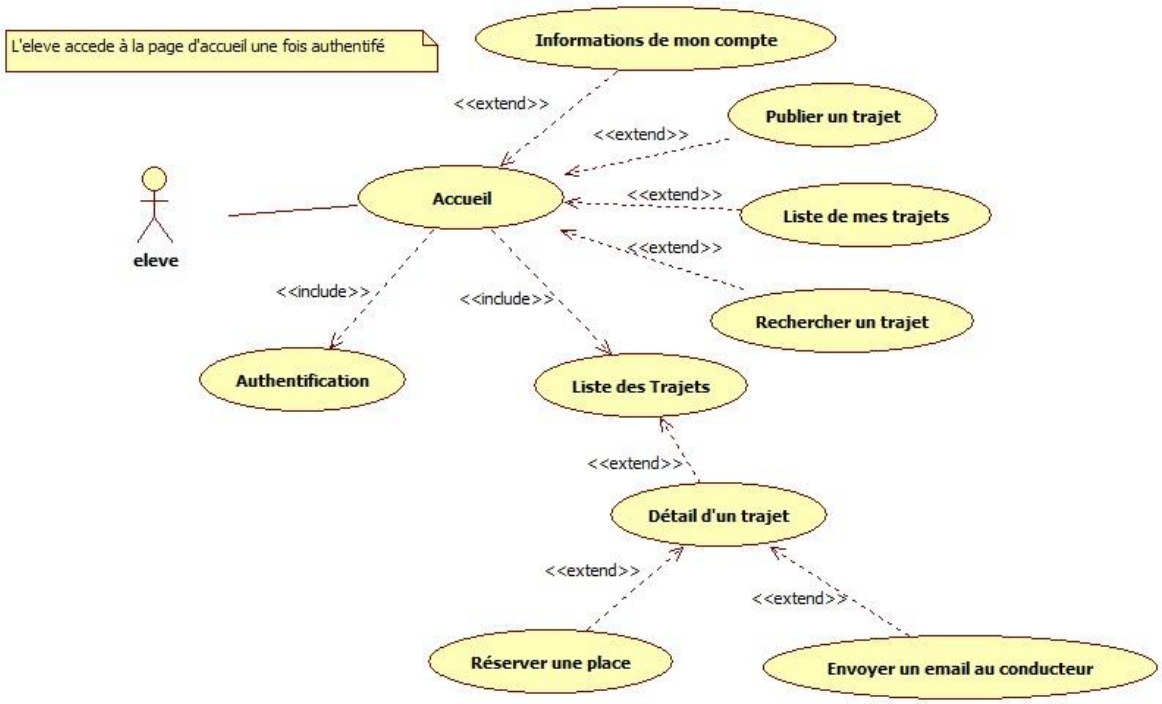


Description des interactions entre objets (Liste des Trajets)

L'utilisateur est authentifié car il a reçu un token lors de la phase d'authentification de l'API REST.
Désormais toutes les phases d'authentifications sont faites à partir de ce token, spécifié dans le **header http**, dans l'attribut **x-auth-token**.(cf cours).Ce token est **indispensable** pour récupérer les différentes routes de l'API REST.
Une fois authentifié, l'API REST retourne la liste de tous les trajets proposés par les conducteurs.

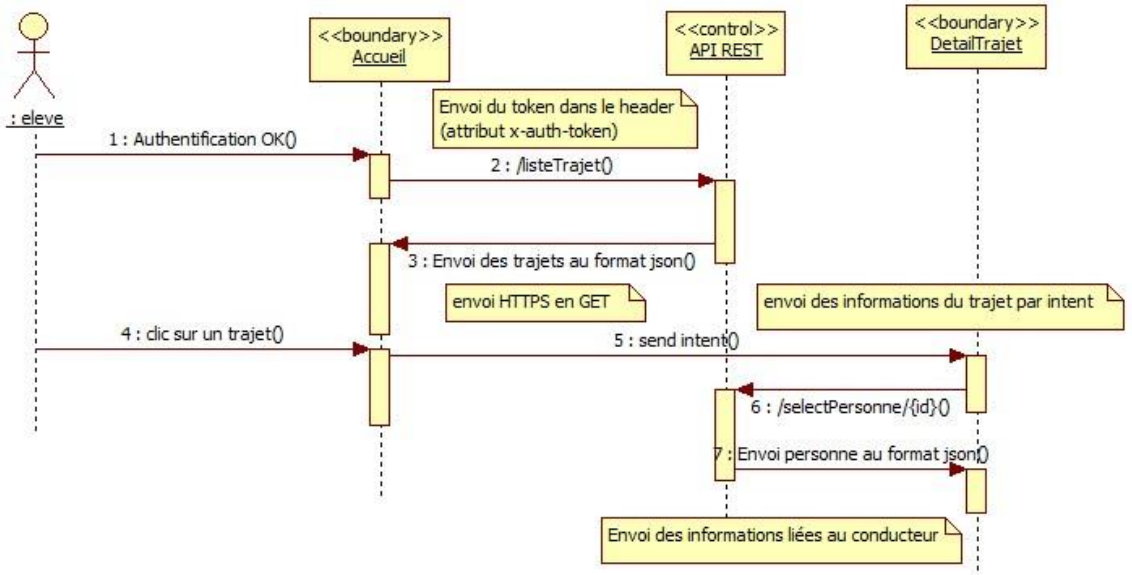


Cas d'utilisation Détail du trajet

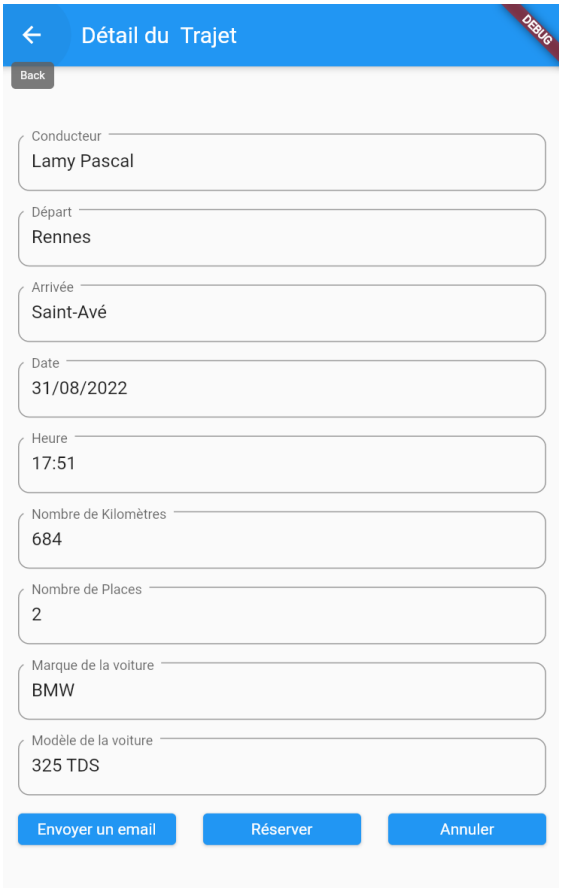


Description des interactions entre objets (Use case détail d'un trajet)

L'utilisateur sélectionne un trajet, pour en afficher les détails.
Le détail du trajet permet d'envoyer un email au conducteur, et de réserver une place pour le trajet.

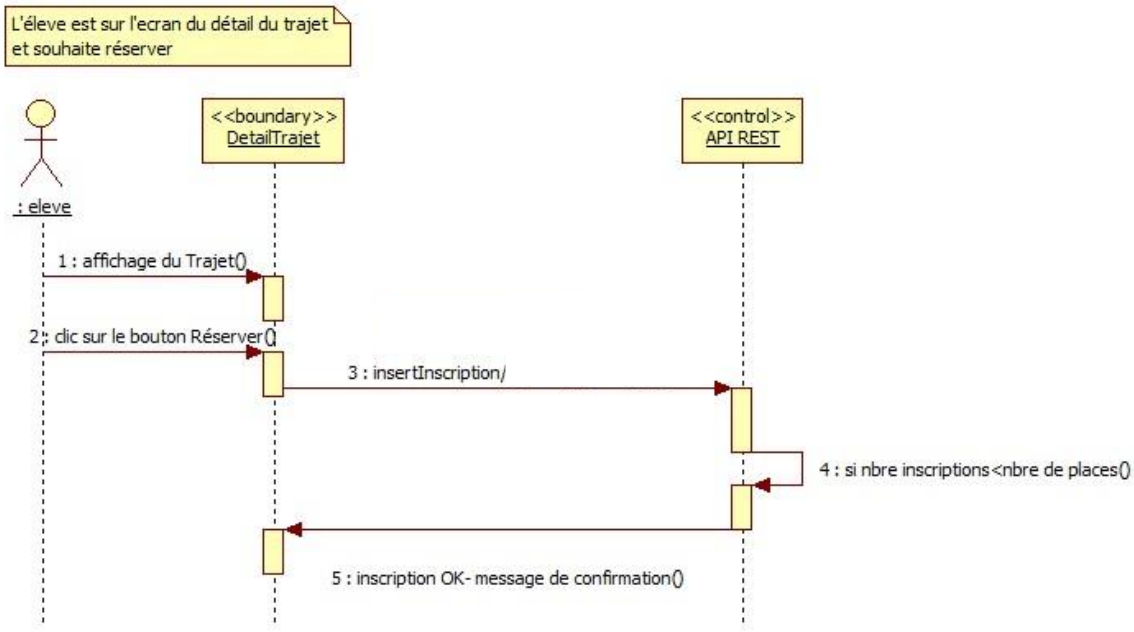


Vue Détail du trajet



Description des interactions entre objets (Use Case Réserver une place)

L'utilisateur clique sur Réserver pour réserver une place pour ce trajet.
Un contrôle est effectué par l'API REST pour vérifier si le nombre de places est suffisantes.
Dans le cas où il n'y a plus de place, l'application affiche un message **réservation impossible (plus de place)**, sinon **réservation effectuée**.

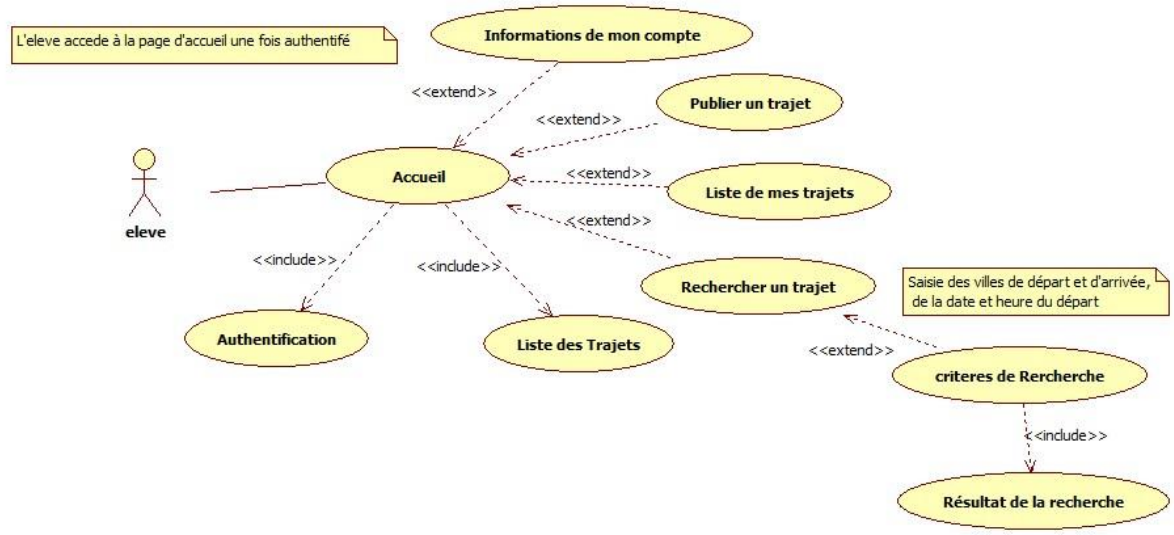


L'API retourne au format JSON le message **OK** indiquant que la réservation est prise en compte.
Dans le cas contraire l'API retourne **NOK**.

Envoi d'un courriel au conducteur

Lorsque le stagiaire qui a réservé son trajet clique sur le bouton **Envoyer un courriel**, l'application déclenche l'envoi d' un email **automatique** au conducteur du trajet, via l'API Brevo.
Cet email informe le conducteur de l'inscription du nouveau passager. Le mail contient les informations du passagers (nom, prenom,telephone,email).

Cas d'utilisation Rechercher un trajet

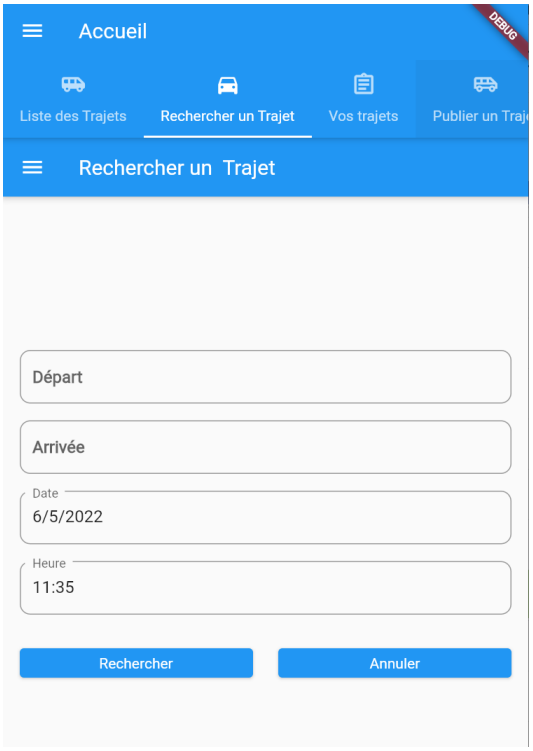


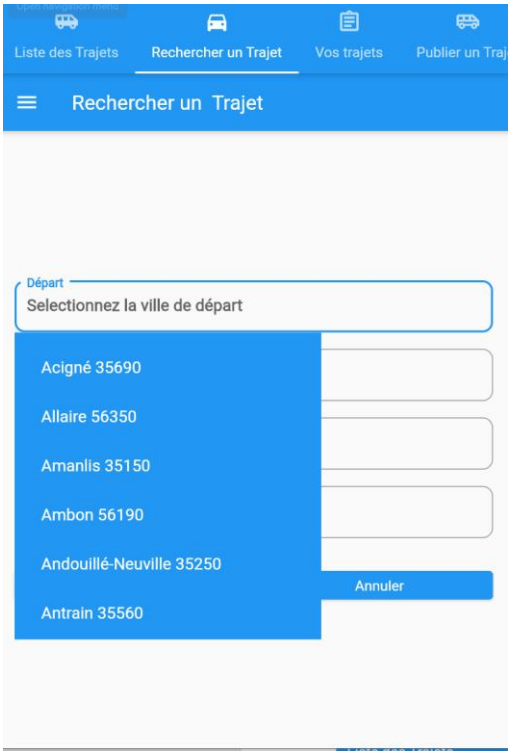
Description des interactions entre objets (Use case Rechercher un trajet)

L'utilisateur saisit dans les listes déroulantes la ville de départ ,et la ville d 'arrivée, puis il saisit la date de départ.

La ville de départ, la ville d'arrivée et la date sont envoyées à l'API REST.
L'API REST retourne les résultats de sa recherche au format JSON.
L'application doit pouvoir stocker les adresses et les villes.

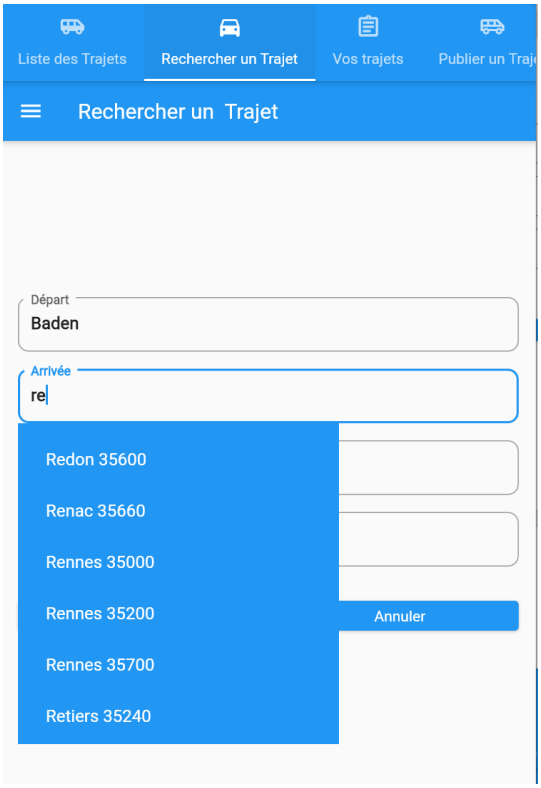
Vue Recherche

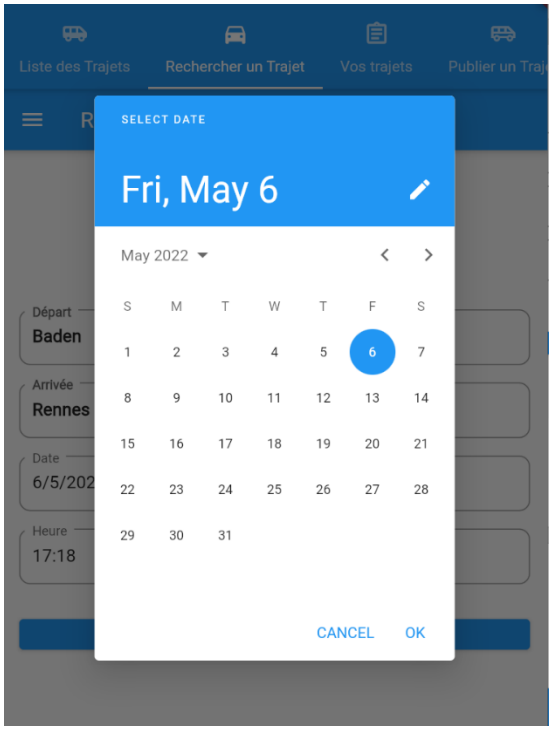


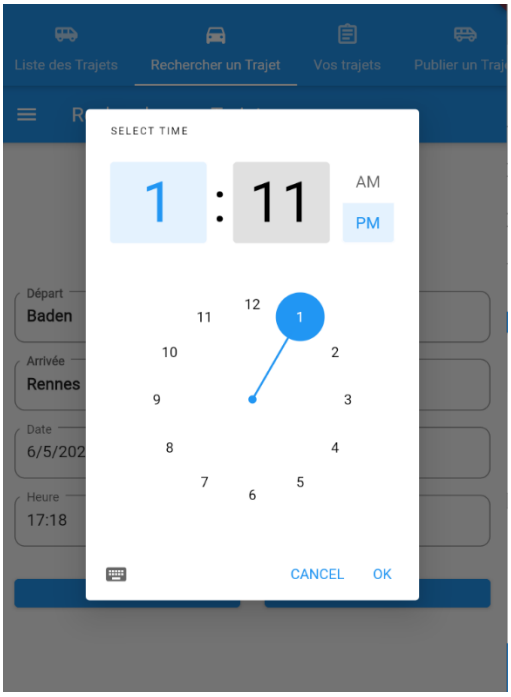


***La date** sera envoyée **au format suivant** : AAAA-MM-JJ h :m, ex : 2026-02-16 16 :18*

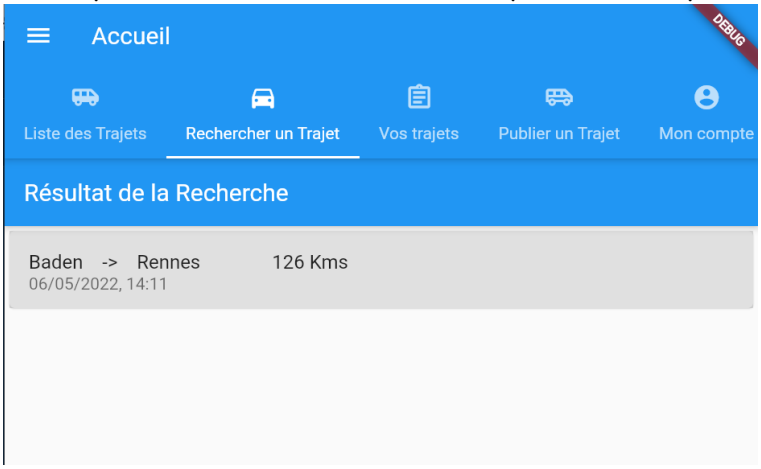
Vue Recherche Trajet -Saisie ville et date



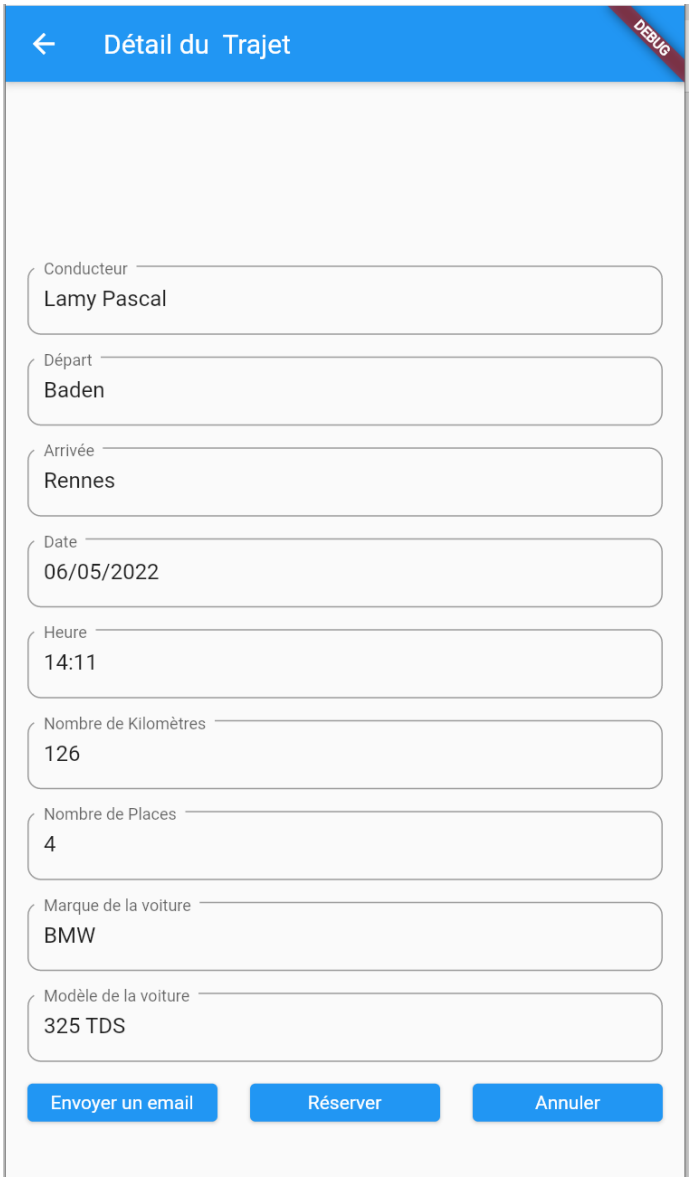




En cliquant sur le bouton rechercher de la vue , l'API REST doit renvoyer les trajets correspondant aux informations envoyées (ville depart , ville arrivée , date et heure)



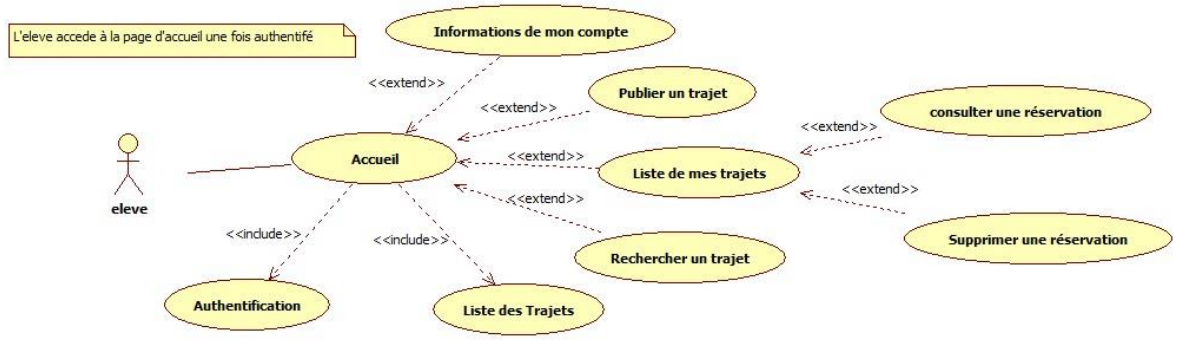
En cliquant sur le trajet, L'API REST RENVOIE le détail du trajet dont le nombre de places disponibles. Ainsi cela permet à l'utilisateur de réserver une place ou d'envoyer un email au conducteur via le front :



Lorsqu'un utilisateur souhaite créer ou se positionner sur un trajet en cliquant sur le bouton Reserver, il faut vérifier que son profil est complété.

Lorsqu'il clique sur le bouton Envoyer un email, l'application déclenche un mailto permettant d'envoyer un email à destination du chauffeur, via l'application de messagerie installée sur le smartphone.

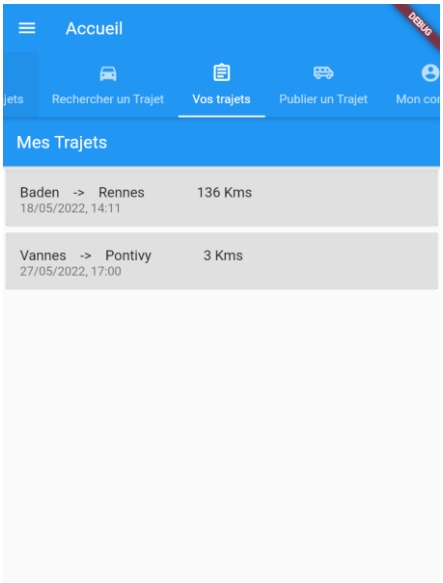
Cas d'utilisation Liste de mes trajets



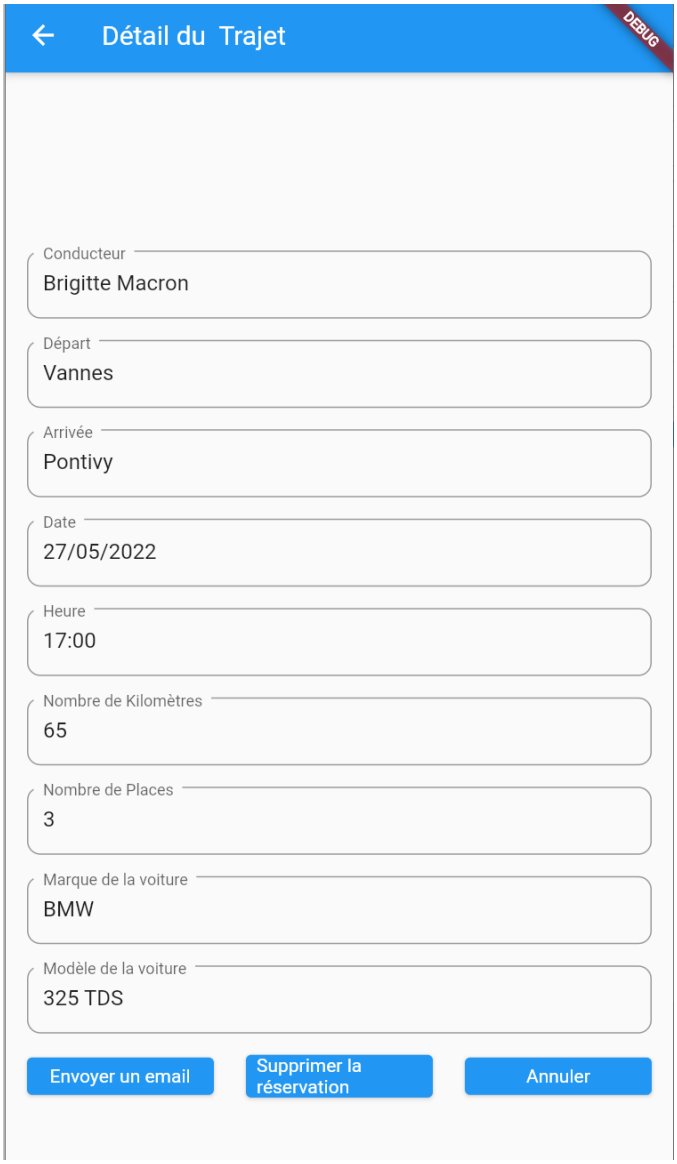
Description des interactions entre objets

L'utilisateur clique sur le bouton « Vos Trajets » afin d'accéder à ses réservations. La vue va récupérer la liste de ses réservations au format JSON à partir de l'Api REST, pour les afficher sous forme de liste.

Vue Mes Trajets



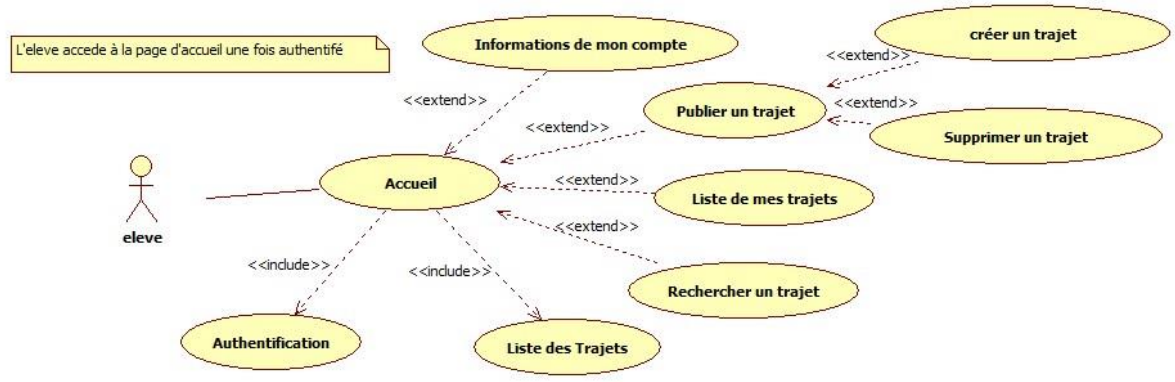
Quand l'utilisateur clique sur un trajet il retrouve le détail de sa réservation.



Description du Use Case Supprimer une réservation)

Quand L'utilisateur clique sur « supprimer une réservation » pour annuler sa réservation pour ce trajet, L'API retourne au format JSON le message **OK** indiquant que la réservation a été supprimée.
Dans le cas contraire l'API retourne **NOK**. L'application redirige l'utilisateur sur ses réservations.

Cas d'utilisation Publier un Trajet



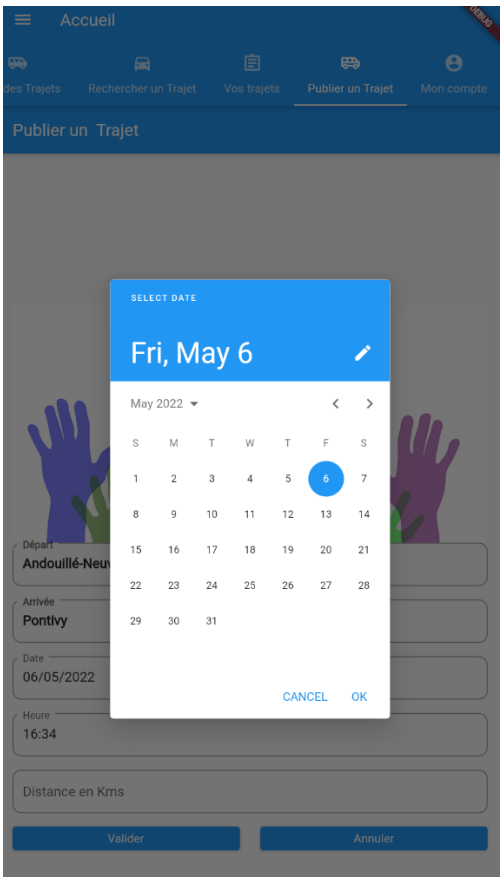
Description des interactions entre objets (Use case Publier un trajet)

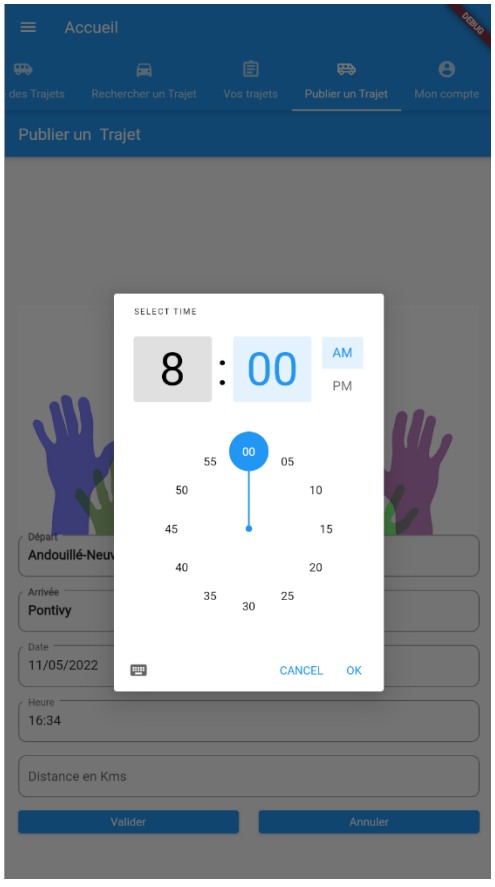
Quand le chauffeur publie un trajet , le front doit envoyer à l'API REST la ville de départ , la ville d'arrivée la date et la distance en kms .
***La date** sera envoyée **au format** (AAAA-MM-JJ h :m, ex : 2022-05-16 16 :18)*

Vue Publier Trajet

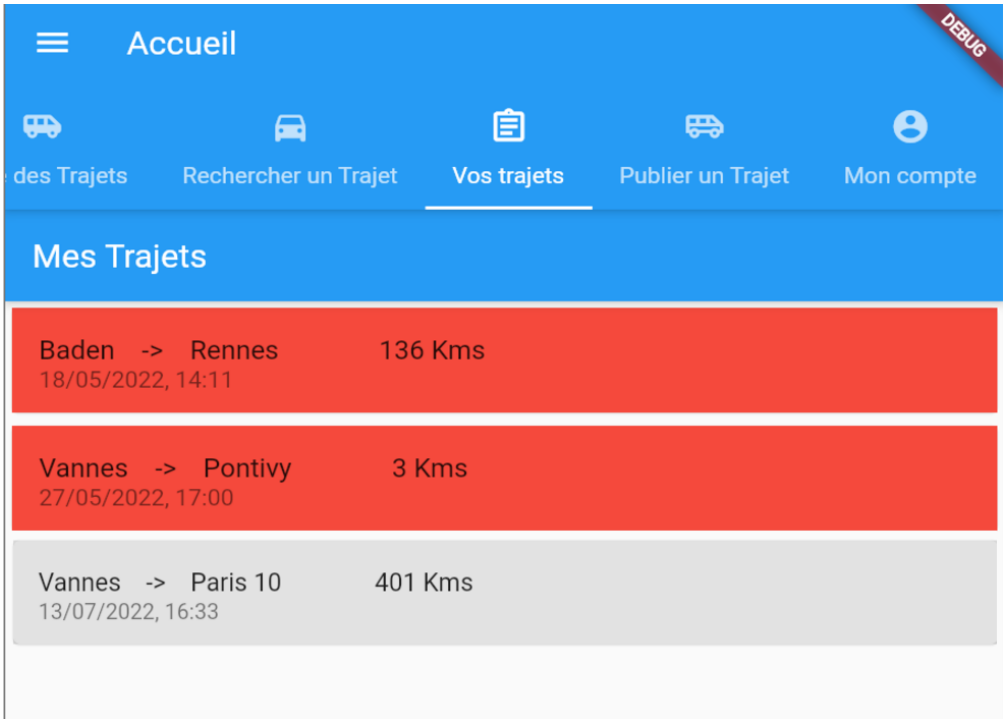

Accueil
des Trajets
Rechercher un Trajet
Vos trajets
Publier un Trajet
Mon compte
Publier un Trajet
Départ
Saisissez la ville de départ
Acigné 35690
Allaire 56350
Amanlis 35150
Ambon 56190
Andouillé-Neuville 35250
Antrain 35560
Annuler


Accueil
des Trajets
Rechercher un Trajet
Vos trajets
Publier un Trajet
Mon compte
Publier un Trajet
Départ
Andouillé-Neuville
Arrivée
Saisissez le lieu de destination
Acigné 35690
Allaire 56350
Amanlis 35150
Ambon 56190
Andouillé-Neuville 35250
Antrain 35560
Annuler


Accueil
Publier un Trajet
SELECT DATE
Fri, May 6
May 2022
CANCEL
OK
Départ
Arrivée
Pontivy
Date
06/05/2022
Heure
16:34
Distance en Kms
Valider
Annuler


Accueil
Publier un Trajet
SELECT TIME
8 : 00
AM
PM
CANCEL
OK
Départ
Arrivée
Pontivy
Date
11/05/2022
Heure
16:34
Distance en Kms
Valider
Annuler

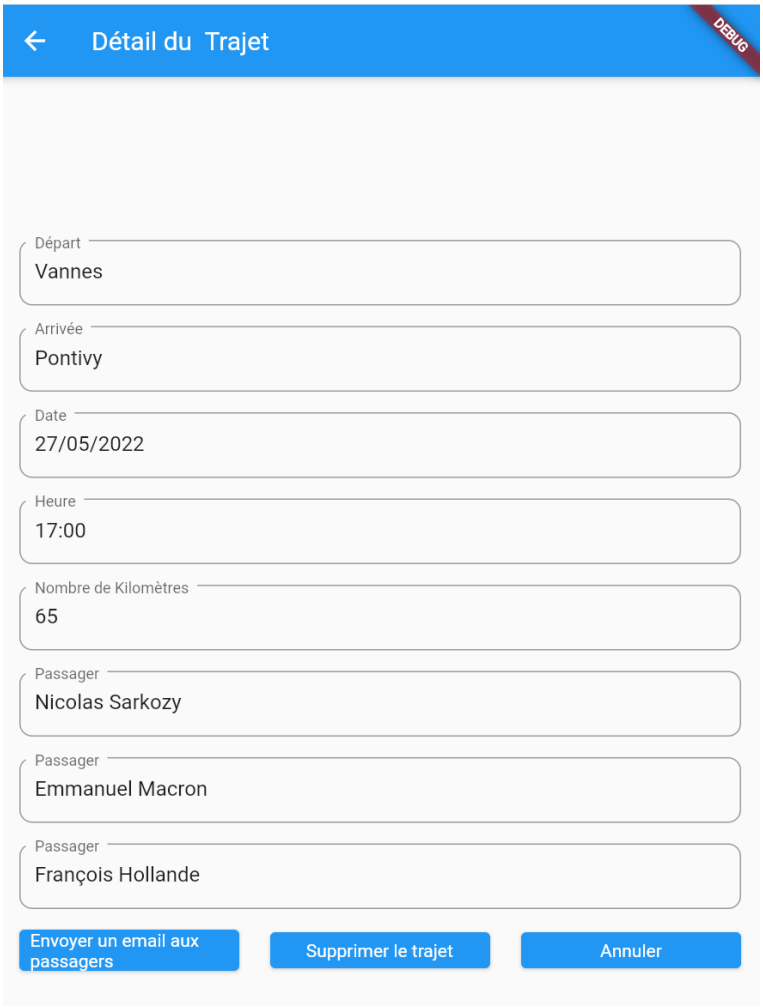
Lorsque l'utilisateur clique sur le bouton Valider, l'API REST renvoie ses **Trajets**.



En rouge sont affichés les trajets proposés par l'utilisateur (chauffeur).
En gris sont affichés ses réservations.

L'API REST doit renvoyer le détail du trajet, quand l'utilisateur clique sur un trajet. L'API doit retourner la ville de départ , la ville d'arrivée, la date et l'heure de départ , la distance en kilomètres entre les deux villes et les noms et prénoms des passagers.

Vue détail trajet conducteur

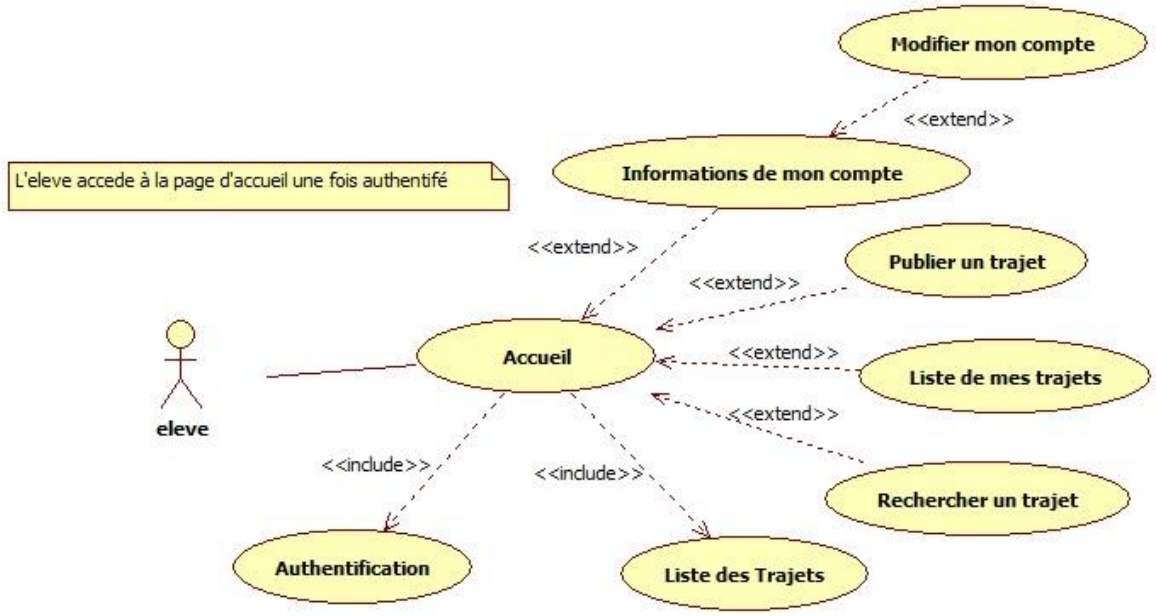


Seulement le conducteur peut envoyer un email à tous les passagers.
Le clic sur le bouton « envoyer un email aux passagers » déclenche l'envoi d'un email automatique via l'API Brevo, indiquant que **la réservation est annulée**.

En supprimant son trajet, **l'API REST supprime également toutes les réservations.**

Le clic sur le bouton « supprimer » déclenche la suppression du trajet et renvoie l'utilisateur à ses réservations. Un email automatique prévient le chauffeur de l'annulation du passager.

Cas d'utilisation Information sur mon compte



L'utilisateur a la possibilité de changer les informations de son compte.

Si l'utilisateur possède une voiture, une liste déroulante propose la liste des marques et deux champs textes permettent de saisir le modèle et le nombre de places.

En aucun cas l'utilisateur peut saisir une deuxième voiture.
l'application doit permettre uniquement 1 véhicule par utilisateur.

En outre, l'utilisateur peut changer son adresse son téléphone, son nom et son prénom.

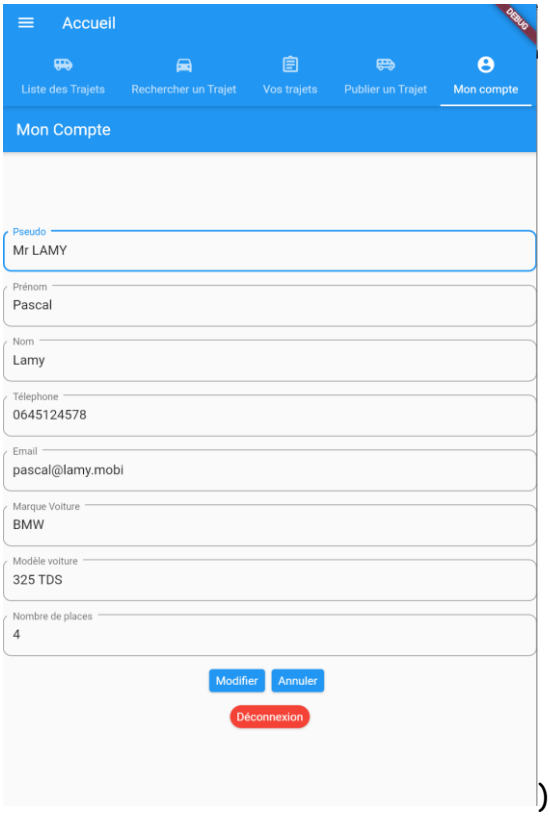
En aucun cas, **il peut changer son adresse email**

Un message s'affichera sur l'écran indiquant que la modification a bien été effectuée.

Le bouton Annuler renvoie l'utilisateur sur la liste des trajets.
Le bouton Déconnexion ferme l'application.

Conformément au RGPD, l'utilisateur a la possibilité de demander la suppression de ses données.
Vous devez prévoir la **suppression en cascade** des informations de l'utilisateur (profile et véhicule).

Vue Mon Compte



)

Travail à réaliser

Vous devez concevoir l'API REST permettant de répondre à l'ensemble des cas d'utilisation, à partir du Framework Symfony ou Spring,Laravel,NodeJS,Next.
Il sera necessaire de :

- concevoir la base de données sous **Mysql ou Postgres**.
  Il est important de définir les contraintes d'intégrités référentielles au niveau des relations des tables.

- planifier votre charge de travail ainsi qu'un rétro-planning sous Jira ou Notion, pour gérer au mieux la réalisation de ce projet.

- concevoir un dossier d'architecture technique expliquant le fonctionnement technique de votre API (MCD, MLD, diagrammes UML …).

- mettre en place des tests d'intégration sous Symfony ou Spring pour tester les différentes routes (voir annexe)
  Pour réaliser ces tests sous Symfony , utilisez la commande suivante :

  ```
  composer require --dev symfony/test-pack
  symfony console make:test
  ```

  Pour réaliser ces tests sous Spring, vous devez installer la dépendance suivante :

  ```
  <dependency>
     <groupId>org.springframework.boot</groupId>
     <artifactId>spring-boot-starter-test</artifactId>
     <scope>test</scope>
  </dependency>
  ```

  Vous trouverez l'utilisation de SpringBootTest sur ce lien :
  https://spring.io/guides/gs/testing-web

- Automatiser les tests, dans Gitlab ou Github.
- Mettre en place des tests d'intégration. Une fois la réussite des tests effectués via Gitlab, le déploiement de l'application doit s'effectuer automatiquement sur votre espace web O2Switch pour symfony.

  Concernant le déploiement sous Spring, vous devez générer un fichier war.

- Générer un rapport de test sur la qualité de votre code à partir de SonarQube.

Lors de la soutenance, les formateurs testeront l'application via Insomnia

 Vous devez solliciter les formateurs si vous avez des questions sur les spécifications techniques et fonctionnelles de l'application.

Liste des routes de l'API REST

Nom de la route	Champs	verbe
registrations		GET
drivers		GET
driver-passengers	{tripid}	GET
user-registrations	{personid}	GET
registration	{personid},{tripid}	POST
registration	{id}	DELETE
brand	{name}{model}	POST
brand	{id}	DELETE
brands		GET
persons		GET
person	{pseudo],{firstname},{lastname},{phone},{email},{car}	POST
person	{id}	DELETE
person	{id}	GET
person	{pseudo},{firstname},{lastname},{phone},{personid}	PUT
trip	{id}	GET
trips	{startingcity},{arrivalcity},{tripdate}	GET
trips	{kms},{personnid},{tripdate},{startingcity},{arrivalcity}	POST
trip	{id}	DELETE
register	{email},{password}	POST
login	{email},{password}	POST
city	{address},{city},{postcode}	POST
city	{id}	DELETE
trips		GET
postcodes		GET
car	{model},{seats},{brand},{carregistration}	POST
car	{model},{seats},{brand},{carregistration }	PUT
car	{model},{seats},{brand},{carregistration }	PATCH
car	{id}	DELETE
cars		GET