POLYTECH TOURS
64 avenue Jean Portalis
37200 TOURS, FRANCE

BASE AÉRIENNE 705 TOURS
Rue Ancienne
37100 TOURS, FRANCE

Rapport de projet

Conception d'une application privée de type BlaBlaCar pour Covoiturage

2023-2024

Professeur encadrant : Jean-Charles Billaut

Client : Base aérienne 705 de Tours

Cheffe de Projet : Caroline Petit

Equipe : Arthur Crochemore, Mohamed Sid Brahim, Cyril Jacques, Jaafar Ghiffi et Adrien Amoroso

Sommaire :

Introduction :

Sujet

Le projet vise le développement d'une PWA pour le covoiturage. Cette PWA doit ressembler à des applications telles que BlaBlaCar ou OuestGo.

Plusieurs contraintes techniques ont été imposées ce qui a amené à l'utilisation de ces technologies :

- Front-end : Vue.Js (et Node.JS (21.6.0))
- Back-end : Laravel / Php (8.2.16) (et Composer (2.7.2))
- Base de données PostgreSQL (16.2)
- Limitation à 200 utilisateurs pour le déploiement initial

Contexte

La base aérienne de Tours cherche à promouvoir le covoiturage pour ses employés, pour leurs trajets base-domicile. Dans l'incapacité d'utiliser des applications commerciales telles que BlaBlaCar ou OuestGo, ils nous ont ainsi proposé de commencer le développement de ce projet.
L'objectif de ce projet était de concevoir le prototype solide de cette application de covoiturage privée pour qu'elle puisse être améliorée dans les années à venir.

Première phase :

Prise en main du sujet

Le première étape du projet a été de comprendre le sujet et de commencer à y réfléchir. N'ayant pas encore toutes les contraintes techniques du projet. Nous avons, donc au plus vite, planifié une première réunion avec nos clients pour recueillir leurs besoins et comprendre précisément chacune des fonctionnalités qu'ils souhaitaient intégrer à cette application. A la suite de cette réunion, un compte rendu a été rédigé par nos soins reprenant les points compris par l'équipe puis envoyé à nos clients afin de valider si nous avions bien saisi leurs besoins. Cette étape nous a permis de fixer une base solide avant de passer à la phase suivante, celle de la conception de l'application.

Précision de l'objectif

Nos clients souhaitaient donc une application de covoiturage destinée dans un premier temps aux membres de la base aérienne 705 de Tours mais qui sera amenée à s'étendre et à être utilisée par les membres d'autres bases militaires. Les clients recherchent dans cette application une utilisation simple et intuitive, où un utilisateur dispose des mêmes possibilités que sur une application de covoiturage standard c'est à dire, la réservation d'un covoiturage publié par un conducteur, la publication d'un covoiturage à l'attention de futurs passagers, la visualisation de ses trajets à la fois du point de vue conducteur et passagers.

Décisions techniques

En parallèle, une liste de contraintes en rapport aux technologies et à l'environnement de développement nous a été transmise. Nous précisons que notre client est la Base Aérienne 705 de Tours et donc que l'utilisation des technologies est très encadrée pour des raisons de protection de données sensibles. Nous avons tout de même eu des choix à faire sur certaines technologies. C'en est suivi une phase de découverte et de recherche sur différents langages notamment Vuejs, Bootstrap, Laravel et Symfony afin de faire des choix judicieux et réfléchis. Nous avons finalement choisi, comme énoncé précédemment, Vuejs pour le développement front et Laravel pour le back. Finalement, après avoir pris en main le sujet et assimilé les besoins de nos clients, nous avons pu commencer la phase de conception.

Deuxième Phase :

Conception

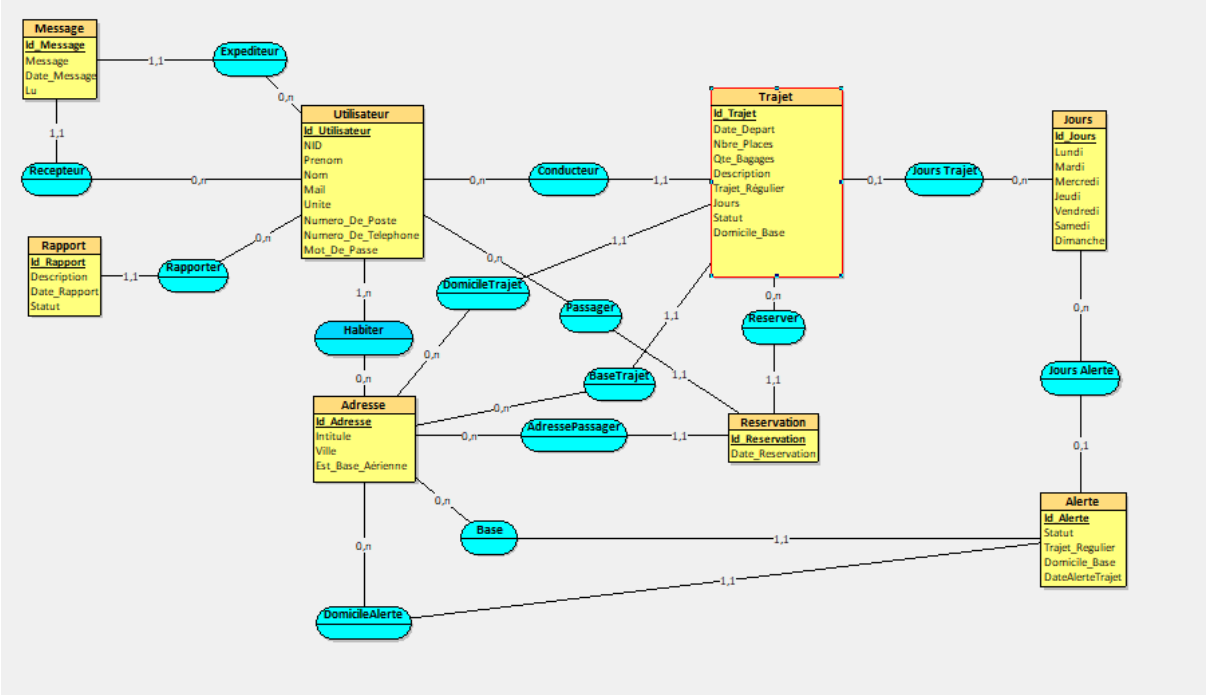
Nous avons consacré deux semaines à la définition des besoins et des exigences fonctionnelles. Un cahier des spécifications complet, clair et détaillé est indispensable pour un développement du code efficace et sans perte de temps causée par des incertitudes. Cela a donné lieu à un cahier des spécifications fonctionnelles que vous pouvez trouver ici : https://docs.google.com/document/d/16mo-7q7OLivZuXtgn2H9OZAAmHR6SNggWoWZn8zIPzI/edit

Nous avons établi un classement de priorités en 4 niveaux. Ce classement a été réalisé en se basant sur le principe MVP ou « Minimum Viable Product », c'est-à-dire que les fonctionnalités du niveau 4 sont celles qui constituent une version de l'application fonctionnelle, utilisable et élémentaire. Celles du niveau 3 et 2 regroupent les fonctionnalités importantes mais pas nécessaire au fonctionnement d'une application de covoiturage et enfin celles du niveau 1 sont les fonctionnalités optionnelles.

Ce classement des fonctionnalités peut être trouvé ici :
https://docs.google.com/spreadsheets/d/1lz2OO0Etshmt2Z_FzzSAEc16wP91GHICcSOCgfOAMXw/edit

Par la suite, nous avons réalisé des maquettes des interfaces de notre application de sorte à se projeter et aider notre patient à comprendre ce qu'on lui propose. Ces maquettes ont été réalisés avec l'outil collaboratif Figma et peuvent être trouvées ici :
https://www.figma.com/design/5bxxEzcQX61w47TbJjTQNU/Projet-Covoiturage?node-id=0-1&t=pfU2GybRvmEKtSui-0

Enfin, la dernière étape de cette phase a été de réaliser la structure des données de notre application. Pour ce faire, nous avons créé un MCD ainsi qu'un diagramme de classe. Le diagramme de classe peut être trouvée ici :
https://lucid.app/lucidchart/f431214f-b9ae-4d07-9f59-7bb5505e7277/edit?invitationId=inv_d8c15314-95e6-48c6-933f-9d14d152dba1&page=0_0



MCD de l'application de covoiturage

Troisième Phase :

Développement Back-End

Technologies et Développement des controllers

Nous avons utilisé Laravel pour le back-end. Laravel est un framework PHP offrant une structure MVC claire et des fonctionnalités avancées comme l'authentification et la gestion de la base de données. Ce framework permet de créer des migrations qui permettent de construire directement la structure de données (MCD) ainsi que des factories permettant de créer des données fictives pour réaliser toutes sortes de tests unitaires et fonctionnelles.

Développement Front-End

Technologies et Développement des interfaces utilisateur

Nous avons opté pour Vue.js pour le front-end, construisant des composants modulaires et réutilisables. Nous avons utilisé Vue Router pour une navigation fluide entre les pages.

La gestion des utilisateurs et des données GPS a été une problématique majeure, résolue par une validation manuelle des inscriptions et une gestion optimisée des adresses.

Intégration et fusion Front-End / Back-End

Initialement, nous avons utilisé Inertia pour la communication entre front-end et back-end, mais sur demande du client, nous avons basculé vers Axios. Axios est une bibliothèque qui permet d'envoyer des requêtes HTTP depuis le navigateur ou depuis le serveur (nodejs). Elle gère aussi les cas d'erreur et traite les réponses réussies. Les appels à l'API d'Axios pour envoyer des requêtes HTTP sont simples et son utilisation facilite l'échange de données entre l'application et le serveur.

Gestion de Projet :

Répartition des rôles et responsabilités

Pour faciliter et optimiser le développement de l'application, la cheffe d'équipe a fait le choix de former deux teams, une team consacré au développement front-end de l'application, c'est à dire la partie avec laquelle l'utilisateur interagit directement, et une team back-end qui gère le traitement et le stockage des données.

Team Front-End : Caroline, Jaafar, Arthur (responsable des merges)
Team Back-End : Mohamed, Adrien, Cyril (responsable des merges)

Communication

Nous sommes passés principalement par Discord pour la communication des informations en interne. Les documents importants et rédigés ont été concentrés sur un drive partagé. La totalité des heures prévues pour le projet s'est faite en présentiel pour favoriser la communication et la motivation des membres de l'équipe.

Organisation et gestion de projet

L'utilisation de méthodes de gestion de projet nous a servi à structurer et à organiser correctement notre charge de travail.
Nous avons notamment utilisé une méthode qui s'appelle Kanban. Elle permet de diviser les tâches en plusieurs catégories : à Faire, en cours et terminée ; nous donnant une vision globale du travail restant et du travail déjà réalisé. Il se présente souvent sous la forme d'un tableau divisé en longues colonnes.

Une bonne méthodologie de gestion de projet est indispensable pour avancer efficacement, particulièrement lors de la phase de développement du code.
Nous avons également géré le projet en mode Scrum. Chaque réunion d'équipe nous a permis de fixer de nouveaux objectifs clairs, aussi bien à court terme qu'à long terme, tout en mettant à jour notre tableau Kanban.

Pour réaliser ce tableau Kanban, nous nous sommes servi de Trello, avec les colonnes suivantes. C'est un outil très intuitif et simple à prendre en main, qui, avec quelques extensions, nous permet de classer par ordre de priorité les missions et donc d'adapter notre travail.
Nous sommes également passés par GitHub pour garantir un travail collaboratif efficace et contrôler correctement les différentes versions du code. Chaque team

était composée d'un seul responsable/maître des merges pour éviter au maximum les conflits de versions.

Échanges avec le client

Nous avons réalisé 3 réunions avec le client, la première pour la découverte du projet, la seconde pour la présentation et la validation du cahier des spécifications et enfin une dernière pour montrer aux clients l'évolution du développement du projet ainsi que pour recueillir leurs avis. Entre ces réunions, nous avons échangé avec le client par mail si nous avions des questions techniques ou pour organiser la prochaine réunion. Chaque échange avec le client nous permettait de s'adapter rapidement à leurs besoins pour toujours respecter leur vision du projet.

Passage de connaissances :

Ce que l'on a fait

Nous avons donc réalisé les fonctionnalités suivantes :

- Fonctionnalités de niveau 1 :
    - Inscription
    - Connexion
    - Déconnexion
    - Réservation de trajet
    - Publication de trajet
    - Visualisation d'un trajet passager et conducteur
    - Demande de réservation (accepter/refuser)
    - Recherche de trajet de niveau 1
    - Modification du profil
- Fonctionnalités de niveau 2 :
    - Visualisation d'un trajet avant réservation
    - Consultation du profil
    - Recherche de trajet de niveau 2

Ce qu'il reste à faire

L'inscription ne doit pas se faire dès qu'elle est validée par l'utilisateur, elle doit être envoyée sous forme de demande d'inscription à l'administrateur. C'est ce dernier qui pourra alors effectuer l'inscription. Cela permettra aussi de gérer l'enregistrement de l'adresse du domicile de l'utilisateur (ce qui n'est pas géré actuellement). Une table doit donc être ajoutée à la base de données pour contenir ces demandes.

La connexion et la déconnexion doivent être gérées de manière sécurisée. Aussi, la connexion doit fonctionner avec le NID. Il faut aussi gérer la gestion de mot de passe oublié.

Les demandes de réservation de trajet doivent être annulées automatiquement passé un certain délai (24 heures). De plus, il faut empêcher les demandes de réservations si le trajet est complet (possible si plusieurs personnes ouvrent les détails d'un trajet en même temps, et qu'ils réservent tous alors qu'il ne reste pas suffisamment de places libres).

La publication de trajet doit prendre en compte les points d'arrêts spécifiés.

Les interfaces de visualisation de trajets doivent être modifiées, notamment pour intégrer la description du trajet. De plus, un utilisateur en attente d'acceptation d'une demande de réservation ne doit pas pouvoir accéder aux informations confidentielles du trajet (numéro de téléphone du conducteur, adresses de domiciles, ...). Aussi, le statut régulier doit être pris en compte, pour les interfaces de détails.

La modification de profil doit, tout comme l'inscription (voir plus haut), aussi passer par une demande auprès de l'administrateur. L'adresse de domicile pourra ainsi être prise en compte.

La recherche de trajets réguliers doit prendre en compte les jours sélectionnés (soit en filtrant les trajets non correspondants, soit en triant par correspondance, ex : lundi, mardi correspond plus à lundi, mercredi que jeudi, vendredi, samedi).

En plus de ces modification à opérer sur les fonctionnalités déjà réalisées, il faut créer certaines interfaces :

- interfaces administrateur (accessibles depuis */admin*)
- interfaces de messageries
- interface d'acceptation / refus de réservation

Enfin, il faut adapter l'application pour la faire devenir PWA et gérer le rechargement des pages (F5) de sorte à ne plus avoir d'erreur 404.

Pour les fonctionnalités qui n'ont pas encore été réalisées, se fier au document suivant :
https://docs.google.com/document/d/16mo-7q7OLivZuXtgn2H9OZAAmHR6SNggWoWZn8zIPzI/edit

Doc utilisateur / technique :

Déploiement local

Pour pour pouvoir lancer l'application, vous devez configurer sur votre machine les outils suivants :

- Pour le backend, Php (8.2.16) et Composer (2.7.2) : https://www.youtube.com/watch?v=pS0U-PsXUlg

- Pour la base de données, PostgreSQL (16.2) : https://www.postgresql.org/download/

- Pour le frontend, Node (21.6.0) (et Vue.JS (5.0.8)) : https://www.youtube.com/watch?v=tsDGFUiNZog

Lors de la configuration de PostgreSQL, faites bien attention à définir votre username comme postgres et votre mot de passe comme password. Si vous n'avez pas cela, vous devrez modifier les champs DB_USERNAME et DB_PASSWORD de votre fichier .env.

Une fois fait, vous devez saisir les commandes suivantes depuis une console ouverte dans le dossier Application :

```
composer install
npm install
php artisan key:generate
```

Il faut aussi récupérer le projet sous ce répertoire git :

- Répertoire git : https://github.com/ArthurCrochemore/Application-Covoiturage

Vous devez ensuite créer une base de données covoiturage (depuis pgAdmin par exemple), ainsi qu'un schéma dans cette base de données nommé covoiturage également.

Vous pouvez ensuite automatiser la création des tables de cette base de donnée, ainsi que la génération de données fictives en saisissant ces 2 commandes depuis la console précédemment ouverte :

```
php artisan migrate
php artisan db:seed
```

Fonctionnement

Déjà, le fichier .env contient les informations de connexion à votre base de données. Le fichier vite.config.js contient les informations pour le lancement de l'application en local. Enfin, le dossier node_modules contient les librairies tierces utilisées par la partie vue.js et le dossier vendor contient celle nécessaire à la partie php.

Le dossier Application a été créé en tant que projet php, dans lequel vue.js a été installé.

Front-end

Les composants vue.js correspondant aux 3 parties de l'application (connexion, application et admin) se trouvent dans le dossier ressources/js.
App.vue correspond à la partie application, App_Admin.vue correspond à la partie administrateur et App_Connexion.vue correspond à la partie connexion.

Chacun de ces fichiers correspond à une SPA (single page app), ce qui signifie que seule certaine portion du fichier sont rechargées lors de la navigation. Ces parties correspondent aux balises vue-router.

Chacune des pages pouvant être chargées dans ces balises vue-router se trouvent dans le dossier Page et sont nommées avec le préfixe "Page". Les routes des vue-router vers ces fichiers Page sont indiqués dans les fichiers : routesApp.js pour l'application, routesAdmin.js pour la partie administrateur et routesConnexion.js pour la partie connexion.

Par exemple :

```
{ path: '/resultat-recherche', component: PageResultatsRecherche,
  props: route => ({
    idBase: route.query.idBase,
    idDomicile: route.query.idDomicile,
    booleenTrajetBaseDomicile: route.query.booleenTrajetBaseDomicile,
    typeTrajet: route.query.typeTrajet,
    jours: route.query.jours,
    heure: route.query.heure,
    date: route.query.date
  })
```

On peut voir que pour ouvrir le composant PageResultatRecherche, il faut indiquer le chemin /resultat-recherche, et que plusieurs paramètres peuvent être spécifié comme idBase ou idDomicile, on peut donc ouvrir cette page comme ceci :

```
router.push({
    path: '/resultat-recherche',
    query: {
        idBase: idBase.value ,
        idDomicile: idDomicile.value ,
        booleenTrajetBaseDomicile: trajetBaseDomicile.value,
        typeTrajet: typeTrajet.value,
        jours: [lundi, mardi, mercredi, jeudi, vendredi, samedi, dimanche],
        heure: heure.value,
        date: date.value
    }
});
```

Aussi, les fichiers routes et App sont lié dans les fichiers appMain.js ,pour l'application, appAdmin.js, pour la partie administrateur, et par appConnexion.js, pour la partie connexion.

Enfin, les images, fichiers .css et tout autres éléments spécifiés dans les fichiers .vue se trouvent, et doivent être mis dans le dossier public.

Back-end

Les différents fichiers .js gérant les différents parties de l'applications son chargées dans le back-end depuis le dossier ressources/views : welcomeApp.blade.php pour l'application, welcomeAdmin.blade.php pour la partie administrateur et welcomeConnexion.blade.php pour la partie connexion.

Les différentes routes du back-end (rien à voir avec vue-router) se trouvent dans le dossier routes. Le fichier par défaut est web.php, ce qui signifie que pour appeler une route de web.php, on fait comme ceci :

```
const response = await axios.get('/trajets-conducteur')
```

Tandis que pour les autres fichiers il faut spécifier le nom du fichiers, par exemple, comme ceci pour le fichier api.php :

```
const response = await axios.get('/api/adresses/base-aerienne')
```

Ces routes définissent les chemins vers les différents fichiers .php. Ainsi, il y est définit les vues, avec des chemins pour les fichier welcome [...] .blade.php, mais aussi les controllers php.

Ces controller peuvent être trouvés dans le dossier app/Http/Controllers. Il s'agit des fonctions appelables par le front pour récupérer (GET) ou modifier (POST) le contenu de la base de données.

Les modèles, fichiers qui contiennent les différentes structures de la base de données ainsi que certaines fonctions sur celles-ci sont définies dans le dossier

app/Http/Models. On y retrouve donc un fichier pour chaque table de la base de données.

En plus de ces modèles, on peut trouver les instructions pour la créations automatiques de la structure de la base de données dans database/migrations (`php artisan migrate`) ainsi que les instructions pour la création des données fictives dans database/seeders (`php artisan db:seed`).

Guide Utilisateur

Interface Login :
A l'ouverture de l'application, vous arrivez sur la page de connexion. D'ici, vous avez trois possibilités :

- Vous connecter si vous êtes déjà inscrit avec votre mail et votre mot de passe
- Vous inscrire
- Mot de passe oublié (non fonctionnel) pour modifier votre mot de passe si vous l'avez oublié

L'inscription requiert un mail, un NID, un mot de passe, un numéro d'unité, un numéro de poste, un prénom, un nom, une adresse postale et un numéro de téléphone.

Une fois validée, l'inscription doit être validée par un administrateur avant de pouvoir se connecter. Pour l'instant, elle se fait instantanément car la partie administrateur n'existe pas encore.

L'application :
Une fois connecté, vous arrivez sur l'interface de recherche. Vous avez une barre de navigation en bas pour pouvoir naviguer au travers des différentes sections de l'application :

- La loupe emmène sur l'interface de recherche de trajet, c'est ici qu'un utilisateur peut trouver puis réserver un trajet
- Le "plus" emmène vers l'interface de publication de trajet, c'est ici qu'un utilisateur peut publier un nouveau trajet, dans lequel il sera conducteur
- Le marqueur de carte emmène sur l'interface "vos trajets", ici, vous pouvez consulter tous les trajets que vous avez créé, ceux que vous avez réservé et qui attendent validation et ceux réservé qui ont été accepté
- La silhouette emmène vers l'interface profil qui permet de consulter ces informations saisies à l'inscription, les modifier, rapporter un bug et se déconnecter
- La phylactère emmène vers l'interface de message qui intégrera la messagerie instantanée, non fonctionnel pour le moment

L'application, interface de recherche :

Depuis la recherche, vous pouvez faire une recherche à travers différents paramètres :

- Un sens (domicile -> base ou base -> domicile), qui peut être inversé en cliquant dessus
- Une adresse de domicile, qui doit être saisie dans départ ou arrivée, en fonction du sens sélectionné. l'adresse doit correspondre à un élément proposé dans le menu déroulant (adresses domicile présente dans la base de données)
- Une adresse de base aérienne, qui doit être saisie dans l'autre champs
- Un type de trajet (régulier, qui se répète chaque semaine, ou ponctuel). Un bouton "switch" permet de basculer entre les deux :
    - Si régulier est choisi, vous pouvez choisir les jours de la semaine où vous comptez faire le trajet à partir des checkboxes qui apparaissent
    - Si ponctuel est choisi, vous devez sélectionner une date
- Une heure, de départ si le trajet part d'une base, d'arrivée sinon

A l'avenir, tous ses paramètres seront pris en compte, actuellement seule la date ou le caractère régulier sont pris en compte. Les trajets déjà complets ne sont pas affichés.

Une fois la recherche lancée, les trajets correspondant sont affichés les uns à la suite des autres. Un bouton à la fin de la liste permet de créer une alerte. Cela permettra (quand cela sera fonctionnel) d'être notifié lors de la publication de trajets correspondant aux paramètres saisies lors de la recherche.

Vous pouvez ouvrir un trajet en cliquant dessus. Vous arrivez alors sur une nouvelle interface qui affiche les détails du trajet. Vous pouvez alors envoyer une demande de réservation en cliquant sur le bouton "réserver".

L'application, interface de publication :

Depuis cette interface, vous pouvez définir les premiers paramètres du trajet à créer :

- Un sens (domicile -> base ou base -> domicile), qui peut être inversé en cliquant sur la double flèche qui se trouvent entre les 2 champs de départ et d'arrivé
- Une adresse de domicile, qui doit être saisie dans départ ou arrivée, en fonction du sens sélectionné. l'adresse doit correspondre à un élément proposé dans le menu déroulant (adresses domicile présente dans la base de données)
- Une adresse de base aérienne, qui doit être saisie dans l'autre champs
- Un type de trajet (régulier, qui se répète chaque semaine, ou ponctuel). Un bouton "switch" permet de basculer entre les deux :

    - Si régulier est choisi, vous pouvez choisir les jours de la semaine où vous comptez faire le trajet à partir des checkboxes qui apparaissent
    - Si ponctuel est choisi, vous devez sélectionner une date
- Une heure, de départ si le trajet part d'une base, d'arrivée sinon
- Une quantité de bagage possible, en fonction de la place que vous aurez
- Un nombre de passager maximum, correspondant au nombre de personnes que vous pouvez accueillir, qui doit être supérieur à 0
- Une description ou vous pouvez ajouter librement du détail pour votre trajet

Une fois tous ces champs remplis (description est facultatif), vous pouvez accéder à l'interface suivante en cliquant sur suivant.

Ici, vous pouvez définir des points d'arrêts, endroits ou vous acceptez de vous arrêter pour prendre des passagers (ce n'est pas encore pris en compte).

Une fois que tout est saisi, vous pouvez créer le trajet en cliquant sur le bouton vert "Créer", vous êtes alors amené sur "Vos trajets" et une notification indique que tout c'est bien passé.

<u>L'application, interface "vos trajets" :</u>
Ici s'affiche tout vos trajets dans trois catégories :

- Vos trajets conducteurs : ils correspondent aux trajets que vous avez créé. Vous pouvez le statut du trajet (Complet ou En ligne) qui indique si le trajet apparaît dans la recherche. Le statut est complet lorsque le nombre max de passagers a été atteint .Un Point vert apparaît dans leur coin droit si une demande de réservation attend votre validation. Passé 24 heures, les demandes de réservation sont automatiquement refusées si vous ne les avez pas encore validées (pas encore mis en place).
- Vos trajets passagers : ils correspondent aux trajets que vous avez réservés et qui sont soit en attente de validation du conducteur ou qui ont été acceptées par ce dernier.
- Vos propositions de trajet : ils correspondront aux trajets que vous proposez (trajets que vous publiez, mais dont vous ne voulez pas forcément être le conducteur, (pas encore mis en place))

En cliquant sur l'un de ces trajets vous pouvez consulter le détail.

Pour les trajets passagers, vous pouvez voir la date du trajet, le départ et l'arrivée, l'heure (de départ ou d'arrivée), le conducteur, les passagers, et le nombre de places encore disponibles. Le conducteur est identifié par son nom, prénom et unité. A l'avenir, son numéro de téléphone sera affiché une fois que l'on sera accepté.

Pour les trajets conducteurs, vous pouvez consulter les mêmes informations en plus des numéros de téléphones des passagers. Vous voyez aussi les demandes de réservations, indiquées par un point vert et un texte “Réservation en attente …” sur les passagers concernés. En cliquant sur ces passagers, vous arrivez sur une interfaces qui vous permet d’accepter ou de refuser la demande.

<u>L’application, interface profil :</u>
Ici, vous pouvez consulter vos renseignements personnels :

- prénom
- nom
- unité
- adresse de domicile
- mail
- numéro de téléphone

Vous pouvez modifier ces informations en cliquant sur le bouton “Modifier”, ainsi que votre mail et votre nid. Une fois les informations saisies, cliquez sur Valider pour envoyer une demande aux administrateurs pour opérer ces modifications, un message de succès s’affiche alors et vous êtes ramené à l'interface Profil.

Le bouton Déconnexion permet de se déconnecter, vous êtes alors ramené à l’interface de connexion.

Le bouton Rapporter un bug ouvre une interface qui permet de signaler un problème aux administrateurs. Saisissez votre message puis cliquez sur Envoyer.

<u>La partie administrateur :</u>
Bien que celle-ci ne soit pas encore créée, ces fichiers existent déjà et elle est donc accessible depuis le chemin /admin.

Remerciements

Au terme de ce projet, nous tenons à exprimer notre profonde reconnaissance à l'ensemble des membres de la Base Aérienne 705. Leur collaboration précieuse et leur confiance en notre équipe ont été des éléments déterminants dans la réussite de cette initiative.

Nous adressons un remerciement particulier à Monsieur Billaut, notre encadrant, pour son soutien indéfectible tout au long du projet. Sa présence bienveillante et ses conseils avisés nous ont permis de progresser et de surmonter les défis rencontrés.

En tant que cheffe de projet, je tiens à remercier chaleureusement mon équipe pour son investissement sans faille, sa motivation constante et son esprit de camaraderie.